Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt

Nicolas T. Courtois

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

Abstract. A popular technique to construct stream ciphers is to use a linear sequence generator with a very large period and good statistical properties and a non-linear filter. There is abundant literature on how to use linear approximations of this non-linear function to attack the cipher, which is known as (fast) correlation attacks. In this paper we explore non-linear approximations, much less well known. We will reduce the cryptanalysis of a stream cipher to solving an overdefined system of multivariate equations.
At Eurocrypt 2000, Courtois, Klimov, Patarin and Shamir have introduced the XL algorithm for solving systems of overdefined multivariate quadratic equations over finite fields. The exact complexity of the XL algorithm remains an open problem. and some authors such as T.T.Moh have expressed serious doubts whether it actually works very well. However there is no doubt that such methods work very well for largely overdefined systems (much more equations than variables), and we confirm this by computer simulations. Luckily systems we obtain in cryptanalysis of stream ciphers are precisely very overdefined.
In this paper we will show how to break efficiently stream ciphers that are known to be immune to all the previously known attacks. For example, we will be able to break the stream cipher Toyocrypt submitted to the Japanese government Cryptrec call for cryptographic primitives, and one of only two candidates accepted to the second phase of Cryptrec evaluation process. Toyocrypt is a 128-bit stream cipher and at the time of submission it was claimed to resist to all known attacks on stream ciphers. Later, Mihaljevic and Imai have published a "guess-and-find" attack that shows that the effective key length in Toyocrypt is 96 bits. Still Toyocrypt may be easily modified to avoid this attack. In this paper we show a new, surprisingly efficient attack, that breaks both Toyocrypt and the modified versions. Our best attack on Toyocrypt takes 2^{92} CPU clocks for a 128-bit cipher. Moreover this type of attack has surprisingly small and loose requirements on the keystream needed, it works even knowing ONLY that the ciphertext is in English.

Key Words: Multivariate cryptography, overdefined systems of multivariate equations, MQ problem, XL algorithm, Gröbner bases, stream ciphers, nonlinear filtering, Toyocrypt-HR1, Toyocrypt-HS1, Cryptrec.

Acknowledgements: This paper has been written following the initial idea suggested by David Wagner.

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 problem is always (in

a sense) overdefined: the adversary is assumed to dispose of, for example, great many plaintext and cipher text pairs, message and signature pairs, etc. He usually disposes of 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 does not depend on how many times the scheme have been used.

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 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 link.

In this paper we will show that a large class of stream ciphers gives an overdefined system of multivariate equations of low degree. The fact that solving such overdefined systems of equations, is much easier than expected, has been demonstrated at Eurocrypt 2000 by Courtois, Klimov, Patarin and Shamir, as a development of an earlier linearization technique proposed by Shamir and Kipnis at Crypto 1999 [24].

The possibility to use multivariate polynomial equations in cryptanalysis was recently brought to public attention by Courtois and Pieprzyk [6]. This kind of attacks seems to give a complexity that grows very slowly with the parameters of the cipher, but with a huge constant. Such attacks became interesting only recently, because only recently some cryptosystems that claim to achieve the security as high as 2^{256} have been proposed. Unfortunately, these attacks are, to say the least, heuristic, it is very difficult to evaluate their complexity, and sometimes it is even impossible to verify if they actually work, because even for small examples they give huge complexities.

In this paper we will apply similar techniques to stream ciphers. Unlike in the work of Courtois and Pieprzyk, our systems of equations will be much more overdefined. We will show that in this case it is possible to predict the behaviour of the XL method with precision and confidence. This will be confirmed by compter simulations.

We will attack a large class of stream ciphers in which there is a linear part, producing a sequence with a large period, and a nonlinear part that produces the output, given the state of the linear part. This includes the very popular filter generator, in which the state of a single LFSR is transformed by a boolean function, and also not less popular combinatorial function generators, in which outputs of several LFSR are combined by a boolean function.

The security of such stream ciphers have been studied by many authors. In [12], 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 [12]. Moreover, several authors have developed improved fast correlation attacks.

In this paper we show that correlation immunity of order one is not sufficient, and show that it is really possible to use (at least in theory) any correlation of any order in an attack. We demonstrate that such attacks can be much faster than exhaustive search for

real stream ciphers, for example for Toyocrypt, that has been accepted to the second phase of the Japanese Cryptrec call for primitives. The paper is organized as follows: In Section 2 and in the Appendix we study the XL algorithm from [25] for solving multivariate quadratic equations, and extend it to equations of higher degree. In Section 3 we study a possibility to apply our results on XL to the problem of cryptanalysis of stream ciphers, the actual attack is given is Section 3.4. In Section 4 we discuss the opportunity to use bent functions in stream ciphers. Then in Section 5 we apply our attack on Toyocrypt stream cipher. In Section 7 we discuss various modifications of Toyocrypt and various extensions of the attack. Finally we present our conclusions.

2 The XL Algorithm

In the Appendix A.2 of this paper we describe a rather obvious extension of the XL algorithm proposed by Courtois; Klimov, Patarin and Shamir at Eurocrypt 2000 [25]. Instead of solving a system of m multivariate quadratic equations with n variables of degree $K = 2$ as in [25], we will consider higher degree equations, i.e. the general case $K \geq 2$: Let D be the parameter of the XL algorithm. Let l_i be the initial equations. 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 $k \leq D - K$, so that the total degree 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. **Solve:** Assume 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. **Repeat:** Simplify the equations and repeat the process

An extended analysis of the complexity of the XL algorithm and in the general case $K \geq 2$ is done in the Appendix. 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. Very little is known about the exact value of $Free$ for $D \geq 3$. In the paper that describes XL, the authors demonstrate that XL works with a series of computer simulations for $K = 2$ and over $GF(127)$ In Appendix B we will show that XL also works very well also for $K > 2$ and over $GF(2)$. Moreover we will be able to explain the origin of the linear dependencies that appear in the XL algorithm, and thus to predict the exact value $Free$ in XL. We will give a formula that allows to compute $Free$ (Conjecture B.3.1). This formula, though not rigourously proven to be exact, is always true in all simulations we have done, and in particular for values K and D that appear in the systems of equations, used in our later cryptographic attacks on stream ciphers. This will allow us to say that our applications of XL should **exactly** as predicted.

3 Application of XL to Stream Ciphers

In this part we will outline a general strategy to apply the XL attack to a general class of stream ciphers. Then we will apply it to some real stream ciphers.

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] for precise definitions. We consider regularly clocked stream ciphers, and also (indifferently) stream ciphers that are clocked in a known way.
For simplicity we restrict to binary stream ciphers in which the state and keystream are composed sequence of bits and that generate one bit at a time. We also restrict to the case when the "connection function" that computes the next state is linear over $GF(2)$. We call L this "connection function", and 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 function f used to combine the bits of the linear stage should obviously be non-linear, and this way of building stream ciphers is sometimes called "nonlinear filtering". The problem of cryptanalysis of a stream cipher can be described as follows. Let $(k_0, \dots k_{n-1})$ be the initial state, then the output of the cipher (i.e. the keystream) is given by:

$$\begin{cases} f \quad (k_0, \dots k_{n-1}) \\ f\left(L\ (k_0, \dots k_{n-1})\right) \\ f\left(L^2(k_0, \dots k_{n-1})\right) \\ \quad \vdots \end{cases}$$

The ciphers described above include the very popular filter generator, in which the state of a single LFSR[1] is transformed by a boolean function, and also not less popular scenario in which outputs of several LFSR are combined by a boolean function (combinatorial function generators).

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. the 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 was sufficiently long.
In our later attacks we will just assume that we have some m bits of the keystream .

3.3 Criteria on the Function f

Let f be the boolean function[2] that is used to combine the outputs of the linear part of the cipher (the entries of the function are for example some bits of the state of some

[1] 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 faster in practice and more secure in practice. A MLFSR is used in the Toyocrypt cipher that we study later.

[2] We restrict to the case when f is a single boolean function, however it is easy to see that the attack works in exactly the same way if there are several boolean functions in parallel.

LFSR's). There many design criteria known on boolean functions. Some of them are obvious to justify, for example a function should be balanced in order to avoid statistical attacks, some are not, no practical attacks are known when the function does not satisfy the criterion, and they are used rather to prevent some future attacks.
It is obvious that for stream ciphers such as described above, the function f should be non-linear. The abundant literature on fast correlation attacks implies also that it should be highly non-linear[3]. Similarly, f should have high order (i.e. high degree in its algebraic normal form), to prevent algebraic attacks. More generally, a "good" boolean function should not only be of high degree, but correlation immune at high order, as pointed ou in [4, 13]. This generalizes the high nonlinearity, which coincides with the correlation immunity at order 1. However up till now, no practical and non-trivial attacks were published, when a function is of high degree, but not higher-order correlation immune, is used in a stream cipher. In this paper we will design such a general attack based on the XL algorithm, and show that it can be successfully applied to Toyocrypt with a complexity much smaller than the exhaustive search.
Our attack will work in two cases:

1. When the boolean function f that combines the LFSR's bits has a low algebraic degree K,
2. and more importantly when f can be approximated[4] with good probability, by a function g that has a low algebraic degree K.

Thus we will assume that:

$$f(s_0, .., s_{n-1}) = g(s_0, .., s_{n-1}) \quad \text{with probability} \quad \geq 1 - \varepsilon \text{ and with } g \text{ of degree } K$$

Note: In the first case, when f has just a low algebraic degree, it is known that the system can be easily broken given $\binom{n}{K}$ keystream bits. A successful example of this attack is described for example in [2]. In this paper we show that, using XL, successful attacks can be mounted given much less keystream bits, and with much smaller complexities. More importantly we don't need that the function has a low algebraic degree (the second case above). 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

We will use the fact that at the time t after the start of the keystream generator, the state is a known linear combination of the key bits. If we know some m bits of the key stream, we have m equations as follows:

$$\begin{cases} b_{t_1} = f\left(L^{t_1}(k_0, \ldots k_{n-1})\right) \\ b_{t_2} = f\left(L^{t_2}(k_0, \ldots k_{n-1})\right) \\ \quad \vdots \\ b_{t_m} = f\left(L^{t_m}(k_0, \ldots k_{n-1})\right) \end{cases}$$

[3] But maybe not perfectly non-linear, see Section 4.

[4] If such a (sufficiently good) approximation exists, there are efficient algorithms to find it. This problem is also known in artificial intelligence as "learning polynomials in the presence of noise", and in the coding theory as "decoding Reed-Muller codes". See for example [4, 13, 10].

We recall that f, and all the L^{t_i} are public, and only the k_j are secret.
We assume that f agrees with a low degree polynomial g of degree K, on some noticeable fraction of inputs $(1-\varepsilon)$. Then each of the keystream bits will gives one known multivariate equation of degree K, with n variables $(k_0, .., k_{n-1})$, and being true with probability $(1-\varepsilon)$:

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

If we have 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 K with n variables. More generally, even if $(1-\varepsilon)^m < \frac{1}{2}$, the attack still work 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 A.4, multiplied by the number of tries necessary to succeed:

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

With μ being the ratio of linearly independent equations in XL. In all practical applications of this attack we will obtain $\mu = 1$. This is due to the fact that in our practical applications the systems are largely overdefined, and thus D is not too big. In this case, it turns out that most of the equations are linearly independent, and more precisely it is so until R exceeds T, after that we will have $Free = T - \epsilon$ with a small ϵ. For more details, see Conjecture B.3.1 and the simulations in the Appendix B that always confirm this conjecture.
The above attack requires about m keystream bits, out of which we chose m at each iteration of the attack.We also need to chose m that minimizes the complexity give above. In practice, since the XL algorithm complexity varies by thresholds depending on value of D, we will in fact chose D and determine a minimal m for which the attack works.

4 Non-linear Filtering using Bent Functions

As we have explained before, due to numerous known fast correlation attacks, cipher such as we described above (for example filter generators) should use a function f that is highly non-linear. From this, Meier and Staffelbach suggested at Eurocrypt'89 to use so called perfect non-linear functions, also 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 (not only stream ciphers) use such functions, or modified versions of such functions [5]. It is for example used in the stream cipher Toyocrypt we study later.
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

[5] In general the authors of [16] did not advocate to use pure bent functions, because is 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].

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 $f : GF(2)^n \to GF(2)$ is a bent function, still the augmented function $GF(2)^{n+\alpha} \to GF(2)^\alpha$ 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 will make any version of the attacks described by Anderson in [1] completely impractical. However, in a way, this paper can be seen as one of many possible extensions of the Anderson's approach, form LFSR's to arbitrary linear generators, i.e. with f and $f \circ L$ that do not have common input bits anymore. We will show that efficient attacks still exist.

Bent Function Used in Toyocrypt

The combining function f of Toyocrypt is built according to the following well known theorem from [22]:

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

$$f(x_1, x_2, x_3, \ldots, x_{2k}) \quad = \quad x_1x_2 + x_3x_4 + \ldots + x_{2k-1}x_{2k} \; + \; g(x_1, x_3, x_5, \ldots, x_{2k-1})$$

Remark: More precisely, as we will se below, the function of Toyocrypt is a Xor of s_{127} and a function build according to the above theorem. We must however 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 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 will not use this property of f, and design a different attack. This new attack can also break many variants of Toyocrypt with a function this is not all under the above form, and such that the "guess and find" attack of [18] will not apply.

5 Application of XL to the Cryptanalysis of Toyocrypt

In this section we will mount a general attack on Toyocrypt, that was, at the time of the design, believed to resist to all known attacks on block ciphers. We will follow the general attack framework described in Section 3.4, but will describe it in more details, to make sure to obtain an **exact** evaluation of the complexity of this attack, and not an approximation.

In Toyocrypt, we have one 128-bit LFSR, and thus $n = 128$. The boolean function is of the form:

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

$$+(...\text{a degree 17 monomial...}) + (...\text{a degree 63 monomial...})$$

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: the higher-order monomials are almost always zero.

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 $K = 2$ and $\varepsilon = 2^{-4}$. This approximation does not allow efficient attacks.

An Approximation of Degree $K = 4$

One can also see that if we put:

$$g'(s) = g(s) + s_{10}s_{23}s_{32}s_{42}.$$

Then $f(s) = g'(s)$ holds with probability very close to $1 - 2^{-17}$. We have $K = 4$ and we have approximatively $\varepsilon = 2^{-17}$ (the error is very small, order of 2^{-63}).

5.1 Our Higher Order Correlation Attack on Toyocrypt

As in Section 3.4, since the LFSR is entirely linear, from the 128-bit key k, we can express the state $s_0, .., s_{127}$ at time t as a set of 128 known linear polynomials in $k = (k_0, .., k_{127})$ which are $s_i = L^{t_i}(k)$.
We have $K = 4$ and $\varepsilon = 2^{-17}$. 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 key stream, 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 key stream bits, one can write from Toyocrypt $m = 1.3 \cdot 2^{16}$ equations of degree 4 and with 128 variables k_i. All these equations are simultaneously satisfied with probability of about 0.52. To this system of equations we apply generalized XL as described in Appendix A.2. We have $n = 128$ and let $D \in \mathbb{N}$. We will multiply each of 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, and since $D < 3K$, 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. See Section B.3. This will be 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 B.3.1, we expect that $Free = T - \epsilon$ with $\epsilon = 1$. Thus XL will be able to solve the system of equations[6] . The complexity of the attack is the complexity of solving a linear system $T \times T$ (we don't need to take more than T equations). Though the best known algorithm for this problem is asymptotically in $T^{2.376}$, see [5], the best practical algorithm we know is Strassen's algorithm, see [29]. The complexity of the Strassen's algorithm is about $7 \cdot T^{log_2 7}$. In practice it gives about $7 \cdot T^{log_2 7}/64$ CPU clocks, because the basic operations are multiplications and addition of bits modulo 2. All these operations can be implemented in a "bitslice" manner, thus transforming a 64 bit machine in a Single Instruction Multiple Data (SIMD) machine with 64 1-bit processors, see [3]. Thus the complexity of our attack is:

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

Thus we are able to break Toyocrypt faster than the exhaustive search of the key, using only about m bits of the keystream, i.e. about 9 kilobytes.

6 Improved Versions of the XL Higher Correlation Attack

The attack described above can be improved by exploring the tradeoff described in Section 3.4.

Exploring the Tradeoff

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 can compensate for the necessity to repeat the attack several times.
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 $D < 2K = 8$, following Section B.3 we expect to have $Free = Min(T, R) - \epsilon$ with $\epsilon = 1$, Thus we may say that for $D < 9$, and $R > 1.1 \cdot T$ the attack will certainly work. It gives the following condition on m:

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

Thus we put: $m = 1.1 \frac{\left(\sum_{i=0}^{D} \binom{n}{i}\right)}{\left(\sum_{i=0}^{D-4} \binom{n}{i}\right)}$ and the complexity of the whole attack is:

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

[6] The XL as described in [25] will work as long as $\epsilon < D + 1$, undoubtedly easily achieved by taking $(R - \binom{m}{2} - m))/T$ slightly higher than 1. Moreover in [6], authors show a version of XL (the so called "T' method") that works for much bigger values of ϵ.

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}

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

Iterating 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 will 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 $K' = 5$ that contain only $n' = 127$ variables will be $R - (T - T') = 129 \cdot 1.6 \cdot 2^{18} - \sum_{i=0}^{5} \binom{128}{i} + \sum_{i=0}^{5} \binom{127}{i} = 2^{25.37}$. 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 $K = 5$, $n' = 127$, $m' = R - (T - T') = 2^{25.37}$ 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 will be: $2^{4.62}\left(2^{75.5} + 2^{87.6}\right) = 2^{92.2}$ CPU clocks.

Thus the best attack is now in 2^{92}, it requires still 2^{65} bits of memory, and now only 51 kilobytes of keystream.

Comparison with other attacks

This 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).

Our attack is sometimes better, and sometimes worse than the Mihaljevic and Imai attack from [18]. On one side, for example, 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 in 2^{32}. On the other side, if only the key stream does not contain 32 consecutive bits, only our attack will work. Similarly, if only the keystream available is 2^{19}, then the Mihaljevic and Imai attack from [18] will require a pre-computation of about 2^{109}, exactly as in the generic tradeoff attack from [23].

7 Extensions, Generalizations, Combination with Other Attacks

Improved Elimination Methods. One should expect that a careful implementation of our attack might be much faster. For this one should use a more careful elimination algorithm, that will generate the equations in a specific order and will eliminate monomials progressively so that they are not generated anymore. It seems that such algorithms exist, see for example the Jean-Charles Faugère's Gröbner bases algorithm F5/2 [8, 9].
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 the in Toyocrypt we replace the bilinear part of f by a random quadratic form, such "guess-and-find" attacks as in [18] will not be 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 known keystream contains only sparsely distributed bits, and does not contain 32 consecutive bits, the "guess-and-find" attack from [18] does not work anymore, and our attack still works.
More Advanced Higher Order Correlation Attacks. One should not underestimate the power of the higher order correlation attacks. This is because the XL attack above can be combined with almost any kind of fast correlation attack, another higher order correlation attack, or other attacks such as Anderson's augmented function attacks from [1]. For example, let us assume that for Toyocrypt we know several linear approximations for all the s_i that appear in the term of degree 17 of f. Assume that all these approximations allow to predict when the corresponding $s_i = 1$ with a probability only vary slightly smaller than $1/2$. Even if this deviation from $1/2$ are very small, the product of all the 17 will be equal to 1 with a probability significantly smaller than 2^{-17}. Thus, in our attack described above, we will be able to select in the keystream m bits, for which $K = 4$ and ε is smaller. This will strictly decrease the complexity of our attack.

8 Conclusion

In this paper we studied higher order correlation attacks on stream ciphers. Our approach is to reduce their cryptanalysis to the problem of solving an overdefined system of multivariate equations. In order to solve such systems of equations, we studied an extension of the XL algorithm proposed at Eurocrypt 2000 for the case of quadratic equations [25]. 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 extend the XL algorithm 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. We do not have a proof that this prediction is always correct, but for largely overdefined systems there is no doubt that XL will work exactly as predicted.
From our results on XL, we presented a new attack on the Toyocrypt stream cipher submitted to the Japanese Cryptrec project. It is a 128-bit stream cipher, that at the time of submission of Toyocrypt was claimed to resist to all known attacks on stream ciphers. It is one of a few accepted to the final phase of Cryptrec. We reduced the problem

of recovering the secret key of the cipher to the problem of solving a largely overdefined system of multivariate equations of degree $K = 4$. In our best version of the XL-based higher-order correlation attack we have the parameter $D = 7$ and since $D < 2 * K$, from our theory, and the simulations done on XL with similar parameters, we expect that all the equations generated in XL will be linearly independent, i.e. there is no doubt that the attack will work exactly as described.
Our faster attack on Toyocrypt requires 2^{92} CPU clocks for a 128-bit cipher. This complexity can be achieved using only a 51 kilobytes of the keystream and 2^{65} bits of memory. Other tradeoffs are possible and our new attack is in many cases the best attack known on Toyocrypt.
We conclude that higher order correlation immunity, should be taken more seriously than previously thought, in the design of stream ciphers.

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. Eli Biham: *A Fast New DES Implementation in Software ;* FSE 4, 1997, Springer, CS 0891.
4. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions;* In Crypto'91, LNCS 576, Springer, pp. 86-100.
5. Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions"; J. Symbolic Computation (1990), 9, pp. 251-280.
6. Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations;* Preprint is available at `http://eprint.iacr.org/2002/044/`.
7. Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases (F_4),* Journal of Pure and Applied Algebra 139 (1999) pp. 61-88. See `www.elsevier.com/locate/jpaa`
8. Jean-Charles Faugère: *Computing Gröbner basis without reduction to 0,* technical report LIP6, in preparation, source: private communication. Also presented at the Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002.
9. Jean-Charles Faugère: Report on a successful attack of HFE Challege 1 with Gröbner bases algorithm F5/2, announcement that appeared in `sci.crypt` newsgroup on the internet in April 19th 2002.
10. 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.
11. Michael Garey, David Johnson: *Computers and Intractability, a guide to the theory of NP-completeness*, Freeman, p. 251.
12. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, Springer, pp. 173-188.
13. Jovan Dj. Golic: *Fast low order approximation of cryptographic functions,* Eurocrypt'96, LNCS 1070, Springer, pp. 268-282.
14. Willi Meier, Nicolas Courtois, Louis Goubin, Jean-Daniel Tacier: *Solving Underdefined Systems of Multivariate Quadratic Equations;* PKC 2002, LNCS 2274, Springer, pp. 211-227.
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 Oorshot, 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. Avialable 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;* in Eurocrypt'96, Springer Verlag, pp. 33-48.
21. Jacques Patarin, Louis Goubin, Nicolas Courtois, + papers of Eli Biham, Aviad Kipnis, T. T. Moh, et al.: *Asymmetric Cryptography with Multivariate Polynomials over a Small Finite Field*; known as 'orange script', compilation of different papers with added materials. Available from `Jacques.Patarin@louveciennes.sema.slb.com`.
22. O. S. Rothaus: *On "bent" functions;* Journal of Combinatorial Theory, Ser. A, Vol. 20, pp. 300-305, 1976.
23. Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers;* Asiacrypt 2000, LNCS 2248, Springer, pp. 1-13.
24. Adi Shamir, Aviad Kipnis: *Cryptanalysis of the HFE Public Key Cryptosystem*; In Advances in Cryptology, Proceedings of Crypto'99, Springer-Verlag, LNCS.
25. 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.
26. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator;* presented at SAC'2000, to appear in LNCS, Springer.
    The LILI-128 description submitted to Nessie can be found at `http://www.isrc.qut.edu.au/lili/`.
27. Markku-Juhani and Olavi Saarinen: *A Time-Memory Tradeoff Attack Against LILI-128;* Available at `http://eprint.iacr.org/2001/077/`.
28. Claude Elwood Shannon: *Communication theory of secrecy systems;* Bell System Technical Journal 28 (1949), see in patricular page 704.
29. Volker Strassen: *Gaussian Elimination is Not Optimal;* Numerische Mathematik, vol 13, pp 354-356, 1969.

A The Analysis of XL Algorithm

A.1 Common Conventions and Notations

In this paper we study solving systems of m multivariate equations with n variables over a small finite field $GF(q)$. the total degree of the equations is $K \geq 2$. We will use very similar notations that in [25]. The variables will be usually denoted by the x_i and belong to a small finite field $GF(q)$ with $q = 2$ unless otherwise stated. When $q = 2$ the equations contain no powers of x_i bigger than 1. We will assume that we want to solve the system for one particular output $b = (b_0, \dots b_{m-1})$, known in advance, and if $f_i(x_0, \dots x_{n-1})$ are the equations, we will systematically put $l_i(x_0, \dots x_{n-1}) \stackrel{def}{=} f_i(x_0, \dots x_{n-1}) - b_i$ so that the system to solve is:

$$\mathcal{A} : \begin{cases} l_0(x_0, \dots x_{n-1}) & = 0 \\ & \vdots \\ l_{m-1}(x_0, \dots x_{n-1}) & = 0 \end{cases}$$

In XL we will always assume that the system has one and unique solution.

The MS Problem and the MQ Problem

We assume that all the multivariate equations are of bounded degree $\leq K$, i.e. they can also include terms of lower degrees.
We call **the MS the problem of order K over $GF(q)$** the problem of finding one (but not necessarily all) solutions, to such system of m multivariate equations with n variables over $GF(q)$. MS stands for "Multivariate Solving" or "Multivariate System".
Following [25], we call MQ the MS problem of order 2. MQ stands for "Multivariate Quadratic" The MQ problem is NP-hard, see [11, 21]. Therefore the general MS problem is also NP-hard.

Manipulating the Equations

We will frequently refer to the equation $l_i(x_0, \dots x_{n-1}) = 0$ as simply the equation l_i. Because the right hand of all our equations is always 0, it is very useful to identify a multivariate polynomial and an equation that says it is equal to 0. Thus the equation $x_0 \cdot l_2(x_0, \dots x_{n-1}) = 0$ will sometimes be referred to as simply the equation $x_0 l_2$. We observe that each solution x that satisfies all the equations l_i, also does satisfy the equations such as $x_i l_j$ and in general any linear combination of products of the form $\prod_i x_i^{r_i} \cdot l_j$.
We say that the equations of the form $\prod_{j=1}^{k} x_{i_j} \cdot l_i = 0$, with all the i_j being pairwise different, are of type $x^k l$, and we call $x^k l$ the set of all these equations. For example the initial equations $\mathcal{A}$ are of type l.
We also denote by x^k the set of all terms of degree exactly k, $\prod_{j=1}^{k} x_{i_j}$. It is a slightly modified extension of the usual convention $x = (x_1, \dots, x_{n-1})$. We define $x^0 = \{1\}$.
Let $D \in \mathbb{N}$. We consider all the polynomials $\prod_j x_{i_j} \cdot l_i$ of total degree $\leq D$. Let $\mathcal{I}_D$ be the set of equations they span. $\mathcal{I}_D$ is the linear space generated by all the $x^k l$, $0 \leq k \leq D-K$. We have $\mathcal{I}_D \subset \mathcal{I}$, $\mathcal{I}$ being the ideal spanned by the l_i ($\mathcal{I}$ could be called $\mathcal{I}_\infty$).
We call $\mathcal{T}$ the set of monomials, including the constant monomial, that appear in all the equations of $\mathcal{I}_D$, $\mathcal{T} = \bigcup_{i=0}^{D} x^i$. We will call T the cardinal of $\mathcal{T}$.

A.2 The Basic Principle of XL

Let D be the parameter of XL algorithm. It is easy to extend the XL algorithm described in [25] for $K = 2$, to the general case $K \geq 2$:

Definition A.2.1 (The XL algorithm). Execute the following steps:

1. **Multiply:** Generate all the products $\prod_{j=1}^{k} x_{i_j} \cdot l_i \in \mathcal{I}_D$ with $k \leq D - K$.
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. **Solve:** Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm).
4. **Repeat:** Simplify the equations and repeat the process

A.3 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 $D - K$, so that the total degree of resulting equations is D. With the notations introduced above, this set of equations is called $\mathcal{I}_D$. Let R be the number of equations generated in $\mathcal{I}_D$ and T be the number of all monomials.
We have, (the first term is dominant):

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

It is likely that not all of these equations are linearly independent, and we denote by $Free$ the exact dimension of $\mathcal{I}_D$. We have $Free \leq R$. We also have necessarily $Free \leq T$. The basic principle of XL is the following: for some D we will have $R \geq T$. Then we expect that $Free \approx T$, as obviously it cannot be bigger than T. More precisely, following [25], when $Free \geq T - D$, it is possible by Gaussian elimination, to obtain one equation in only one variable, and XL will succeed.

The Saturation Problem in XL

The main problem in XL, is that, in general, not all the equations generated in XL are linearly independent. In the complexity evaluation of [25] the authors assume that "most of the equations are linearly independent". Obviously for XL algorithm to work it is not necessary that "most of the equations are linearly independent". It is sufficient that for some D, the number $Free$ of linearly independent equations satisfies $Free \geq T - D$. This is called the saturation problem. In [19], 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 T. T. Moh shows a very special example on which XL always fails, for any D [19]. This example is very interesting however we consider here random systems of quadratic (or degree K) equations, i.e. we look at the behaviour of XL on most of the systems, and not on some very special systems.

In Section 3 the author presents an argument that is apparently wrong. 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 T.T. Moh is inappropriate, not to say incorrect.
We will see in this paper that in many interesting cases, all or most of the equations are linearly independent in XL, and moreover we will be able to explain (and predict) exactly the number (and the origin) of all the linear dependencies.

A.4 Analysis of XL Extended to Solving Equations of Degree K

Let μ be the proportion of linearly independent equations generated in XL. If μ is not negligible (this is confirmed by our simulations, see Appendix B). We have

$$T = \sum_{\lambda=0}^{D} \binom{n}{\lambda}$$

$$R = m \left(\sum_{\lambda=0}^{D-K} \binom{n}{\lambda} \right).$$

If μ is the proportion of the equations that are linearly independent, $\mu = Free/R$, then XL algorithm will succeed if $R \times \mu \geq T$, i.e. when

$$m \left(\sum_{\lambda=0}^{D-K} \binom{n}{\lambda} \right) \times \mu \geq \sum_{\lambda=0}^{D} \binom{n}{\lambda}.$$

If we consider only the two main terms of the summation, this gives:

$$m \binom{n+1}{D-K} \times \mu \geq \binom{n+1}{D}$$

Therefore we get:

$$m \frac{(n-D+K+1) \cdot \ldots \cdot (n-D+2)}{D(D-1) \cdot \ldots \cdot (D-K+1)\mu}$$

Then assuming that $D \ll n$ we get:

$$D \geq \text{ about } \frac{n}{m^{1/K} \mu^{1/K}}.$$

Then the total complexity of the XL attack is about:

$$T^{\omega} \approx \binom{n}{n/(m\mu)^{1/K}}^{\omega}$$

with $\omega \leq 3$ being the exponent of the Gaussian reduction.
Asymptotically this is expected to be a good evaluation, at least when $m = \varepsilon n^K$ with $\varepsilon > 0$.

B About the Exact Dimension of $\mathcal{I}_D$ in XL

Let $Free$ be the dimension of $\mathcal{I}_D$, i.e. the maximum number of equations that are linearly independent in XL algorithm. In the paper that describes XL, the authors demonstrate that XL works with a series of computer simulations over $GF(127)$. In this section we will present some computer simulations on the XL algorithm over $GF(2)$. No such simulations has been published so far. Apparently we will be able to predict the exact value $Free$ obtained in these simulations. We will propose a formula for $Free$, that though not rigourously proven to work, is confirmed with good precision in all our experiments.
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 $\leq K$ (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 $\forall\ i \quad l_i(x_0, \ldots, x_{n-1}) = 0$.

B.1 The behaviour of XL for $K = 2$ and $D = 3$.

In general it is not possible that $Free = R$. One reason is that $Free$ cannot exceed T. We have therefore always

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

We have done various computer simulations with $D = 3$ and in our simulations, for $K = 2$ and $D = 3$, we have always[7] $Free = Min(T, R) - \epsilon$ with $\epsilon = 0, 1, 2$ or 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.

K	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 $K = 2$ 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 $Free \geq T - D$.

[7] Actually $Free$ is almost always the minimum of the two functions, around the point where the two graphics meet, we sometimes observed a "smooth" transition, and int his case we observe that $Free = Min(T, R) - \epsilon$ with $\epsilon = 0, 1, 2$ or 3. In our simulations the smooth transition is visible for $n = 10$, $m = 16$, $D = 3$.

B.2 The behaviour of XL for $K = 2$ and $D = 4$.

When $D = 4$ we do not have $Free = Min(T, R)$ anymore.

K	2	2	2	2	2	2	2	2	2	2
n	10	10	10	20	20	20	20	20	20	40
m	5	10	11	20	24	28	30	32	36	128
D	4	4	4	4	4	4	4	4	4	4
R	280	560	616	4220	5064	5908	6330	6752	7596	105088
T	386	386	386	6196	6196	6196	6196	6196	6196	102091
$Free$	265	385	385	4010	4764	5502	5865	6195	6195	96832

Figure 2: XL simulations with $K = 2$ and $D = 4$ (same notations as for Figure 1).
We see that for $D = 4$ most of the equations are linearly independent. We observed that for $K = 2$ 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 $R - \binom{m}{2} - m \leq T$, means 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 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 $l_1 = x_1x_3 + x_4$ 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: $l_1 = l_1x_1x_3 + l_1x_4$.

B.3 Tentative Conclusion on XL and More Simulations for $K \geq 2$ 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 B.3.1 (Behaviour of XL for $D < 3K$).

1. For $D = K..2K-1$ 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 $D = 2K..3K-1$ there are linear dependencies and we have $Free = Min\left(T, R - \left(\sum_{i=0}^{D-2K} \binom{n}{i}\right)\left(\binom{m}{2} + m\right)\right) - \epsilon$ with $\epsilon = 0, 1, 2$ or 3.
   The factor $\left(\binom{m}{2} + m\right)$ 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 $D > 2K$ there are other linear dependencies that are products of these by monomials in x_i of degree up to $D - 2K$, and to count these we have multiplied their number by a factor $\left(\sum_{i=0}^{D-2K} \binom{n}{i}\right)$.
3. It is also possible to anticipate what will happen for $D \geq 3K$. However, it is more complex, and in this paper we do not need to know this.

Up till now, this conjecture is pure guessing, as we did not yet consider systems with $K > 2$. Here is a series of simulations with such systems.

K	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 2: XL simulations with $K = 3$ (same notations as for Figure 1).

K	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 3: XL simulations with $K = 4$ (same notations as for Figure 1).
All these simulations confirm our Conjecture B.3.1.