Practical Algebraic Cryptanalysis for Dragon-Based Cryptosystems

Johannes Buchmann[1], Stanislav Bulygin[2], Jintai Ding[3], Wael Said Abd Elmageed Mohamed[1], and Fabian Werner[4]

[1] TU Darmstadt, FB Informatik Hochschulstrasse 10, 64289 Darmstadt, Germany
{buchmann,wael}@cdc.informatik.tu-darmstadt.de
[2] Center for Advanced Security Research Darmstadt (CASED)
Stanislav.Bulygin@cased.de
[3] Department of Mathematical Sciences, University of Cincinnati, Cincinnati OH 45220, USA
jintai.ding@uc.edu
[4] TU Darmstadt
fw@cccmz.de

Abstract. Recently, the Little Dragon Two and Poly-Dragon multivariate based public-key cryptosystems were proposed as efficient and secure schemes. In particular, the inventors of the two schemes claim that Little Dragon Two and Poly-Dragon resist algebraic cryptanalysis. In this paper, we show that MXL2, an algebraic attack method based on the XL algorithm and Ding's concept of Mutants, is able to break Little Dragon Two with keys of length up to 229 bits and Poly-Dragon with keys of length up to 299. This contradicts the security claim for the proposed schemes and demonstrates the strength of MXL2 and the Mutant concept. This strength is further supported by experiments that show that in attacks on both schemes the MXL2 algorithm outperforms the Magma's implementation of F4.

1 Introduction

The multivariate-based public-key cryptosystems (MPKCs) are public-key cryptosystems that are based on the problem of solving multivariate quadratic equations over finite fields. This problem is called "MQ-problem" and it is NP-complete [1]. Several MPKCs based on the MQ-problem have been proposed in the last two decades. An overview of MPKCs can be found in [2,3].

Recently, in the International Journal of Network Security & Its Applications (IJNSA), Singh et al. presented a new multivariate-based public-key encryption scheme which is called Little Dragon Two (LD2 for short) that is constructed using permutation polynomials over finite fields [4]. According to the authors, LD2 is as efficient as Patarin's Little Dragon [5], but secure against all the known attacks. Shortly after the publication [4] appeared, linearization equations were found by Lei Hu, which became known in the private communication with the first author.

S.-H. Heng, R.N. Wright, and B.-M. Goi (Eds.): CANS 2010, LNCS 6467, pp. 140–155, 2010.

Due to these linearization equations, the authors of the LD2 scheme presented another scheme called Poly-Dragon [6]. Poly-Dragon as well as LD2 is constructed using permutation polynomials over finite fields. It is considered as an improved version of Patarin's Big Dragon cryptosystem [5]. The Poly-Dragon scheme was also proposed as an efficient and secure scheme. In particular, the inventors of the Poly-Dragon scheme claim that Poly-Dragon resist algebraic cryptanalysis.

In this paper, we present an algebraic attack for the LD2 and Poly-Dragon schemes. We present experiments that show the weakness of these schemes by solving corresponding multivariate quadratic equation systems over $\mathbb{F}_2$ up to number of variables equal to 229 for LD2 and 299 for Poly-Dragon. In this attack, we use an improved implementation of the MXL2 algorithm. This algorithm is an improvement of the MutantXL [7] algorithm which was used to break the MQQ scheme in [8].

We analyzed the reason why MXL2 is able to break the two schemes efficiently. We use also two different versions of Magma's implementation of F4 to compare our results. For all the instances that we have, MXL2 outperforms Magma's F4 in terms of memory and time. We discuss linearization equations for both schemes and present a way of obtaining these linearization equations.

This paper is organized as follows. In Section 2 we give an overview of the LD2 scheme. Section 3 is a description for Poly-Dragon. We then briefly present the MXL2 algorithm in Section 4. Section 5 presents the experimental results. We discuss linearization equations in Section 6. Finally, we conclude the paper in Section 7.

2 LD2: Little Dragon Two Multivariate Public-Key Cryptosystem

The Little Dragon Two, LD2, multivariate public-key cryptosystem is a mixed type scheme that has a public key in which plaintext and ciphertext variables are "mixed" together. LD2 is a modified version of Patarin's Little Dragon cryptosystem and it is constructed using permutation polynomials over finite fields. In this section, we present an overview of the LD2 scheme. In multivariate public-key cryptosystems, the main security parameters are the number of equations and the number of variables. The authors of LD2 did not propose any such security parameters. For a more detailed explanation see [4].

Definition 1. *Let $\mathbb{F}_q$ be the finite field of $q = p^n$ elements where p is prime and n is a positive integer. A polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ is called a permutation polynomial in n variables over $\mathbb{F}_q$ if and only if one of the following equivalent conditions holds:*

1. *the function f is onto.*
2. *the function f is one-to-one.*
3. *$f(x) = a$ has a solution in $\mathbb{F}_q$ for each $a \in \mathbb{F}_q$.*
4. *$f(x) = a$ has a unique solution in $\mathbb{F}_q$ for each $a \in \mathbb{F}_q$.*

Simply speaking, this means that a polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ is a permutation polynomial over $\mathbb{F}_q$ if it induces a bijective map from $\mathbb{F}_q$ to itself.

Lemma 1. *Let $Tr(x)$ denotes the trace function on the field $\mathbb{F}_{2^n}$ i.e. $Tr : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is defined by $Tr(x) = x + x^2 + x^{2^2} + \cdots + x^{2^{n-1}}$. The polynomial $g(x) = (x^{2^{rk}} + x^{2^r} + \alpha)^\ell + x$ is a permutation polynomial of $\mathbb{F}_{2^n}$, when $Tr(\alpha) = 1$ and $\ell \cdot (2^{rk} + 2^r) = 1 \bmod 2^n - 1$.*

A proof of Lemma 1 is presented by the authors of [4]. A more detailed explanation for permutation polynomials and trace representation on finite fields can be found in [9].

Suppose that $X = (x_1, x_2, \ldots, x_n)$ denotes the plaintext variables and $Y = (y_1, y_2, \ldots, y_n)$ denotes the ciphertext variables. In the LD2 scheme, the public key equations are multivariate polynomials over $\mathbb{F}_2$ of the form:

$$\begin{cases} P_1(x_1, x_2, \ldots, x_n,\ y_1, y_2, \ldots, y_n) = 0, \\ P_2(x_1, x_2, \ldots, x_n,\ y_1, y_2, \ldots, y_n) = 0, \\ \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ P_\lambda(x_1, x_2, \ldots, x_n,\ y_1, y_2, \ldots, y_n) = 0, \end{cases}$$

where $P_1, P_2, \ldots, P_\lambda$ are polynomials of $\mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2$ of total degree two.

Due to the restrictions on r, k and ℓ placed by Lemma 1, there are a few choices for r, k and ℓ to produce a permutation polynomial $g(x) = (x^{2^{rk}} + x^{2^r} + \alpha)^\ell + x$ and to use $g(x)$ to design a public-key scheme with a quadratic public key. We can choose $r = 0, n = 2m - 1, k = m$ and $\ell = 2^m - 1$ for example, then $G(x) = (x^{2^m} + x + \alpha)^{2^m - 1} + x$ is a permutation polynomial. By choosing $r = 0, n = 2m - 1, k = m$ and $\ell = 2^m + 1$, the authors of [4] stated that it is not clear whether $G'(x) = (x^{2^m} + x + \alpha)^{2^m + 1} + x$ is a permutation polynomial or not while it can produce a quadratic public key.

Let S and T be two invertible affine transformations. Then the plaintext can be permuted to a ciphertext using the relation $G(S(x_1, \ldots, x_n)) = T(y_1, \ldots, y_n)$. Suppose $S(x_1, \ldots, x_n) = u$ and $T(y_1, \ldots, y_n) = v$. Therefore the relation between plaintext and ciphertext can be written as follows:

$$\begin{aligned} (u^{2^m} + u + \alpha)^{2^m - 1} + u &= v \\ (u^{2^m} + u + \alpha)^{2^m - 1} + (u + v) &= 0 \\ (u^{2^m} + u + \alpha)^{2^m} + (u + v)(u^{2^m} + u + \alpha) &= 0 \\ ((u^{2^m} + u) + \alpha)^{2^m} + u^{2^m + 1} + u^2 + u\alpha + vu^{2^m} + vu + v\alpha &= 0 \\ (u^{2^m} + u)^{2^m} + \alpha^{2^m} + u^{2^m + 1} + u^2 + u\alpha + vu^{2^m} + vu + v\alpha &= 0 \\ \vdots & \\ u^{2^m} + \alpha^{2^m} + u^{2^m + 1} + u\alpha + vu^{2^m} + vu + v\alpha &= 0 \\ u^{2^m + 1} + u^{2^m} v + uv + u\alpha + u^{2^m} + v\alpha + \alpha^{2^m} &= 0 \end{aligned} \tag{1}$$

It is known that the extension field $\mathbb{F}_{2^n}$ can be viewed as a vector space over $\mathbb{F}_2$. Let $\beta = \{\beta_1, \beta_2, \ldots, \beta_n, \}$ be a normal basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ for some $\beta \in \mathbb{F}_{2^n}$.

Therefore any $z \in \mathbb{F}_{2^n}$ can be expressed as $z = \sum_{i=1}^{n} z_i \beta_i$, where $z \in \mathbb{F}_2$. By substituting $u = S(x_1, \ldots, x_n)$ and $v = T(y_1, \ldots, y_n)$, Equation (1) can be represented as n quadratic polynomial equations of the form:

$$\sum a_{ij} x_i x_j + \sum b_{ij} x_i y_j + \sum c_k y_k + \sum d_k x_k + e_l = 0 \tag{2}$$

where the coefficients $a_{ij}, b_{ij}, c_k, d_k, e_l \in \mathbb{F}_2$.

In Equation (2), the terms of the form $\sum x_i x_j + \sum x_k + c_1$ are obtained from u^{2^m+1}, the terms of the form $\sum x_i y_j + \sum x_k + \sum y_k + c_2$ are obtained from $u^{2^m} v + uv$, the terms of the form $\sum x_i + c_3$ is obtained from $u\alpha + u^{2^m}$ and $v\alpha$ gives the terms of the form $\sum y_i + c_4$, , where c_1, c_2, c_3 and c_4 are constants.

The secret parameters are the finite field element α and the two invertible affine transformations S and T. A plaintext $X = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$ is encrypted by applying Algorithm 1. Decryption is accomplished by using Algorithm 2.

A discussion for the security and the efficiency of the proposed scheme is presented in [4]. As a conclusion, the authors claimed that they present an efficient and secure multivariate public key cryptosystem that can be used for both encryption and signatures.

Algorithm 1. Encryption

Inputs
  A plaintext message $X = (x_1, x_2, \ldots, x_n)$ of length n.
Output
  A ciphertext message $Y = (y_1, y_2, \ldots, y_n)$ of length n.
Begin
Substitute the plaintext $(x_1, x_2, \ldots, x_n)$ in the public key.
Get n linear equations in the ciphertext variables $(y_1, y_2, \ldots, y_n)$.
Solve these linear equations by Gaussian elimination method to obtain the correct ciphertext $Y = (y_1, y_2, \ldots, y_n)$.
End

Algorithm 2. Decryption

Inputs
  A ciphertext message $Y = (y_1, y_2, \ldots, y_n)$ of length n and the secret parameters (S, T, α).
Output
  A plaintext message $X = (x_1, x_2, \ldots, x_n)$ of length n.
Begin
Let $v = T(y_1, y_2, \ldots, y_n)$.
Let $z_1 = \alpha + 1 + v + v^{2^m}$.
Let $z_2 = z_1^{2^m - 1}$.
Let $z_3 = v + 1 + z_2$.
Let $X_1 = S^{-1}(v + 1)$.
Let $X_2 = S^{-1}(z_3)$.
Return (X_1, X_2), Either X_1 or X_2 is the required secret message.
End

3 Poly-Dragon Multivariate Public-Key Cryptosystem

The Poly-Dragon multivariate public-key cryptosystem is a mixed type scheme that has a public key of total degree three, two in plaintext and one in ciphertext. Poly-Dragon is based on permutation polynomials and is supposed to be as efficient as Patarin's Big Dragon [5]. As well as LD2, Poly-Dragon did not have a proposed security parameters. In this section, we introduce an overview of the Poly-Dragon scheme. See [6] for more details.

Definition 2. *Let $\mathbb{F}_q$ be the finite field of of characteristic p. A polynomial of the form*

$$L(x) = \sum_i \alpha_i x^{p^i}$$

with coefficients in an extension field $\mathbb{F}_q$ of $\mathbb{F}_p$ is called a p-polynomial over $\mathbb{F}_q$.

Simply speaking, a polynomial over $\mathbb{F}_q$ is said to be a p-polynomial over $\mathbb{F}_q$ if each of its terms has a degree equal to a power of p. A p-polynomial is also called linearized polynomial because for all $\beta, \gamma \in \mathbb{F}_q$ and $a \in \mathbb{F}_p$ it satisfies the following properties: $L(\beta + \gamma) = L(\beta) + L(\gamma)$ and $L(a\beta) = aL(\beta)$. In [9], it is proved that $L(x)$ is a permutation polynomial of $\mathbb{F}_q$ if and only if the only root of $L(x)$ in $\mathbb{F}_q$ is 0.

Proposition 1. *Let $L_\beta(x) = \sum_{i=0}^{n-1} \beta_i x^{2^i} \in \mathbb{F}_{2^n}$ be a p-polynomial defined with n an odd positive integer and $\beta = (\beta_1, \beta_2, \ldots, \beta_n) \in \mathbb{F}_{2^n}$ such that the weight of β is even and that 0 and 1 are the only roots of $L_\beta(x)$. Then*

$$f(x) = (L_\beta(x) + \gamma)^\ell + Tr(x)$$

is a permutation polynomial of $\mathbb{F}_{2^n}$, where l is any positive integer with $(2^{k_1} + 2^{k_2}) \cdot \ell \equiv 1 \pmod{2^n - 1}, \gamma \in \mathbb{F}_{2^n}$ with $Tr(\gamma) = 1$ and k_1 , k_2 are non negative integers such that $gcd(2^{k_1} + 2^{k_2}, 2^n - 1) = 1$.

Proposition 2. *The polynomial $g(x) = (x^{2^{k2^r}} + x^{2^r} + \alpha)^\ell + x$ is a permutation polynomial of $\mathbb{F}_{2^n}$ if $Tr(\alpha) = 1$ and $(2^{k2^r} + 2^r) \cdot \ell \equiv 1 \pmod{2^n - 1}$.*

The two permutation polynomials $g(x) = (x^{2^{k2^r}} + x^{2^r} + \alpha)^\ell + x$ and $f(x) = (L_\beta(x) + \gamma)^\ell + Tr(x)$ from Proposition 1 and Proposition 2 are used in Poly-Dragon public-key cryptosystem. The permutation polynomials in which ℓ is of the form $2^m - 1$ and $r = 0, n = 2m - 1, k = m, k_2 = m$ and $k_1 = 0$ are used to generate the public key. Therefore, for key generation $G(x) = (x^{2^m} + x + \alpha)^{2^m - 1} + x$ and $F(x) = (L_\beta(x) + \gamma)^{2^m - 1} + Tr(x)$ are used where α, β, γ are secret.

The relation between plaintext $X = (x_1, x_2, \ldots, x_n)$ and ciphertext $Y = (y_1, y_2, \ldots, y_n)$ can be written as $G(S(x_1, x_2, \ldots, x_n)) = F(T(y_1, y_2, \ldots, y_n))$,

where S and T are two invertible affine transformations. This relation can be written as $(u^{2^m}+u+\alpha)^{2^m-1}+u=(L_\beta(v)+\gamma)^{2^m-1}+Tr(v)$ such that $S(x_1,x_2,\ldots,x_n)=u$ and $T(y_1,y_2,\ldots,y_n)=v$. Multiplying by $(u^{2^m}+u+\alpha)\times(L_\beta(v)+\gamma)$, which is a nonzero in the field $\mathbb{F}_{2^n}$, we obtain:

$$\begin{aligned}&(u^{2^m}+u+\alpha)^{2^m}(L_\beta(v)+\gamma)+u(u^{2^m+u+\alpha})(L_\beta(v)+\gamma)\\&+(u^{2^m}+u+\alpha)(L_\beta(v)+\gamma)^{2^m}+Tr(v)(u^{2^m+u+\alpha})(L_\beta(v)+\gamma)=0\end{aligned}\quad(3)$$

The extension field $\mathbb{F}_{2^n}$ can be viewed as a vector space over $\mathbb{F}_2$. Then we can identify $\mathbb{F}_{2^n}$ with $\mathbb{F}_2^n$. Let $Tr(v)=\zeta_y\in\{0,1\}$ and by substituting $u=S(x_1,x_2,\ldots,x_n)$ and $v=T(y_1,y_2,\ldots,y_n)$, in Equation (3), we obtain n nonlinear polynomials equations of degree three of the form:

$$\sum a_{ijk}x_ix_jy_k+\sum b_{ij}x_ix_j+\sum(c_{ij}+\zeta_y)x_iy_j+\sum(d_k+\zeta_y)y_k+\sum(e_k+\zeta_y)x_k+f_l,\quad(4)$$

where $a_{ijk},b_{ij},c_{ij},d_k,e_k,f_l\in\mathbb{F}_2$.

The secrete parameters are the finite field elements α, β, γ and the two invertible affine transformations S and T. A plaintext $X=(x_1,x_2,\ldots,x_n)\in\mathbb{F}_2^n$ is encrypted by applying Algorithm 3. Decryption is accomplished by using Algorithm 4.

In [6], the authors stated a proof for the validity of the generated plaintext by the decryption algorithm. A discussion for the security and the efficiency of the proposed scheme is also presented. As a conclusion, the authors claimed that they presented an efficient and secure multivariate public key cryptosystem that can be used for encryption as well as for signature.

Algorithm 3. Encryption

1: **Inputs**
2: A plaintext message $X=(x_1,x_2,\ldots,x_n)$ of length n.
3: **Output**
4: A ciphertext message pair $(Y^{'},Y^{''})$.
5: **Begin**
6: Substitute the plaintext variables $(x_1,x_2,\ldots,x_n)$ and $\zeta_y=0$ in the public key.
7: Get n linear equations in the ciphertext variables $(y_1,y_2,\ldots,y_n)$.
8: Solve these linear equations by Gaussian elimination method to obtain the ciphertext variables $Y^{'}=(y_1,y_2,\ldots,y_n)$.
9: Substitute the plaintext $(x_1,x_2,\ldots,x_n)$ and $\zeta_y=1$ in the public key.
10: Get n linear equations in the ciphertext $(y_1,y_2,\ldots,y_n)$.
11: Solve these linear equations by Gaussian elimination method to obtain the ciphertext variables $Y^{''}=(y_1,y_2,\ldots,y_n)$.
12: Return the ordered pair $(Y^{'},Y^{''})$ as the required ciphertext.
13: **End**

Algorithm 4. Decryption

Inputs
  A ciphertext message $(Y^{'}, Y^{''})$ and the secrete parameters $(S, T, \alpha, \beta, \gamma)$.
Output
  A plaintext message $X = (x_1, x_2, \ldots, x_n)$ of length n.
Begin
Let $v_1 = T(Y^{'})$ and $v_2 = T(Y^{''})$.
Let $z_1 = L_\beta(v_1) + \gamma$ and $z_2 = L_\beta(v_2) + \gamma$.
Let $\bar{z_3} = z_1^{2^m-1}$ and $\bar{z_4} = z_2^{2^m-1}$.
Let $z_3 = \bar{z_3} + Tr(v_1)$ and $z_4 = \bar{z_4} + Tr(v_2)$.
Let $z_5 = z_3^{2^m} + z_3 + \alpha + 1$ and $z_6 = z_4^{2^m} + z_4 + \alpha + 1$.
Let $z_7 = z_5^{2^m-1}$ and $z_8 = z_6^{2^m-1}$.
Let $X_1 = S^{-1}(z_3 + 1)$.
Let $X_2 = S^{-1}(z_4 + 1)$.
Let $X_3 = S^{-1}(z_3 + z_7 + 1)$.
Let $X_4 = S^{-1}(z_4 + z_8 + 1)$.
Return (X_1, X_2, X_3, X_4), Either X_1, X_2, X_3 or X_4 is the required secret message.
End

4 MXL2: The MutantXL2 Algorithm

MXL2 [10] is an algorithm for solving systems of quadratic multivariate equations over $\mathbb{F}_2$ that was proposed at *PQC*2008. It is a variant of MutantXL [7] which improves on the XL algorithm [11]. The MXL2 and MutantXL algorithms are similar in using the concept of Mutants that is introduced by Ding [12], while MXL2 uses two substantial improvements over $\mathbb{F}_2$. In this section, we present a brief overview of the Mutant strategy, the MXL2 algorithm, MXL2 improvements and MXL2 implementation.

The main idea for the Mutant strategy is to maximize the effect of lower-degree polynomials occurring during the linear algebra step for the linearized representation of the polynomial system. Throughout this section we will use $x := \{x_1, \ldots, x_n\}$ to be a set of n variables. We consider

$$R := \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$$

the Boolean polynomial ring in x with graded lexicographical ordering $<_{gradlex}$ on the monomials of R. We consider elements of R as polynomials over $\mathbb{F}_2$ where the degree of each term w.r.t any variable is 0 or 1.

Let $P := (p_1, \ldots, p_m)$ be a system of m quadratic polynomial equations in R. We are interested in finding a solution for $P(x) = 0$ that is based on creating further elements of the ideal generated by the polynomials of P. In this context, Mutants are defined as follows.

Definition 3. *Let I be the ideal generated by the finite set of polynomials P, $f \in I$. For any representation of f, $f := \sum_{p \in P} f_p p$, we define the level of this representation to be $max\{deg(f_p p) : p \in P, f_p \neq 0\}$. Let $Rep(f)$ be the set of all representations of f. Then the level of f with respect to P is defined to be the*

minimum of levels of all representations in $Rep(f)$. The polynomial f is called a Mutant with respect to P if its degree is less than its level.

From a practical point of view, the concept of Mutants can be applied to the linear algebra step in the matrix-based algorithms for solving systems of multivariate polynomial equations, for example F4 and XL. In [10,7,13,14], during the linear algebra step, the new polynomials that appear having a lower degree are mutants. By using these mutants, MutantXL as well as MXL2 can solve multivariate quadratic polynomial equations at a lower degree than the usual XL.

The MXL2 algorithm performs the following steps:

- *Initialization:* Set $P = \{p_1, \ldots, p_m\}$, $D = Max\{deg(p) : p \in P\}$, the elimination degree $ED = min\{deg(p) : p \in P\}$, the set of Mutants $M = \emptyset$.
- *Echelonize:* Consider each term in P as a new variable. Set $P = \tilde{P}$ where $\tilde{P}$ is the row echelon form of P.
- *Solve:* If there are univariate polynomials in P, then determine the values of the corresponding variables and substitute in P. If the system is solved then return solution and terminate. Otherwise, set $D = Max\{deg(p) : p \in P\}$ and $ED = min\{deg(p) : p \in P\}$.
- *ExtractMutants:* Add all new polynomials of degree less than D in P to M.
- *MultiplyMutants:* If $M \neq \emptyset$, then select the necessary number of Mutants that have degree $k = min\{deg(p) : p \in M\}$, multiply lower degree Mutants by all terms up to degree D, remove the multiplied Mutants from M, add the new polynomials obtained from the multiplication to P, set $ED = k + 1$ then go back to *Echelonize*.
- *Extend:* Add all the polynomials that are obtained by multiplying a subset of the degree D polynomials in P by all variables that are smaller than the leading variable of the partition leading variable, set $D = D + 1$, $ED = D$. Go to back to *Echelonize*.

As stated in [10], MXL2 has two important advantages over MutantXL. The first is the use of the necessary number of mutants and the second is extending the system only partially to higher degrees. The main idea for the first improvement is to add only a necessary number of mutants to the given system. This number is numerically computed. By using not all the emerged mutants, the efficiency is increased, for space efficiency only a few mutants are used and for the time efficiency the multiplications to generate higher degree polynomials do not have to be performed to all mutants.

The second improvement is the usage of the partial enlargement technique in which the polynomials at degree D are divided according to their so-called "leading variable". The leading variable of a polynomial $p \in R$ is the smallest variable in the leading term with respect to the order defined on the variables. Instead of enlarging all the partitions, only the non-empty partitions are multiplied by all the variables that are smaller than the leading variable. This is accomplished partition by partition. This partial enlargement technique gives also an improvement in time and space since the system can be solved using a

lower number of polynomials. Moreover, in some systems mutants may appear in the last step together with the solution of the system. These mutants are not fully utilized. Using partial enlargement technique enforces these mutants to appear before the system is solved.

As a result of using the two MXL2 improvements, MXL2 generates the same solution as if all Mutants would have been used and all partitioned would have been multiplied. MXL2 solves multivariate quadratic polynomial equations using a smaller number of polynomials than MutantXL.

The MXL2 algorithm has been implemented in C/C++ based on the latest version of M4RI package [15]. In this package, there exist three different algorithms for computing row echelon form. In this paper, we use the Method of Four Russians Inversion (M4RI) algorithm [16].

5 Experimental Results

In this section we present experimental results of the attack on LD2 and Poly-Dragon by using MXL2 and compare the performance of MXL2 with two versions of Magma's implementation of F4 namely V2.13-10 and V2.16-1. The reason for using these two versions is that when we used Magma's version (V2.16-1), we found that this version solves the LD2 and Poly-Dragon systems at degree 4 while MXL2 as well as Magma V2.13-10 solves at degree 3. In this context, it is not fair to use only this version (V2.16-1) in the comparison.

The main task for a cryptanalyst is to find a solution of the systems of equations that represent the LD2 and Poly-Dragon schemes. These systems were essentially implemented as described in [4] and [6] respectively. Magma version (2.16-1) has been used for the implementation. Due to the high number of variables, this direct approach is not very efficient but it is sufficient for modeling purposes. For real-life applications, there are more elegant ways to create the public key using specialized software and techniques like polynomial interpolation (see [17] for example).

All the experiments are done on a Sun X4440 server, with four "Quad-Core AMD Opteron^{TM} Processor 8356" CPUs and 128GB of main memory. Each CPU is running at 2.3 GHz. In these experiments we used only one out of the 16 cores.

We tried to solve different systems with the same number of variables. As a result of our experiments, we noticed that the complexity for different systems of LD2 and Poly-Dragon schemes with the same number of variable will be, essentially, the same. In this context, the results given in this section are for one particular instance for each system.

Table 1 presents the required steps of solving an LD2 instance of $n = 229$ using MXL2. In this table, for each step (Round) we present the elimination degree (ED), the matrix dimensions (Matrix), the rank of the matrix (Rank), the total number of mutants found (#Mutants), the number of linear mutants found (#LM) and the number of univariate polynomials found (#Uni).

Table 2 and Table 3 show results of the LD2 systems for the range 79-229 equations in 79-229 variables and results of the Poly-Dragon systems for the

Table 1. MXL2: Results for LD2-229Var

Round	ED	Matrix	Rank	#Mutants	#LM	#Uni
1	2	229 × 26336	229	0	0	0
2	3	457 × 1975812	457	0	0	0
3	3	52670 × 2001690	52669	686	228	2
4	2	915 × 26336	913	226	226	108
5	2	913 × 26336	805	118	118	0
6	2	14847 × 26336	7140	1	1	1
7	2	7140 × 26336	7021	118	118	118

Table 2. Performance of MXL2 versus F4 for Little-Dragon-Two

Sys	$F4_{v2.13}$			$F4_{v2.16}$			MXL2		
	D	Mem	$Time$	D	Mem	$Time$	D	Mem	$Time$
79	3	490	29	4	321	26	3	211	22
89	3	841	116	4	1600	203	3	346	40
99	3	1357	238	4	2769	411	3	545	73
109	3	2092	500	4	2046	331	3	844	122
119	3	3102	998	4	6842	1142	3	1251	217
129	3	4479	1827	4	11541	2529	3	2380	458
139	3	6280	3134	4	8750	1723	3	3387	742
149	3	8602	4586	4	23325	5795	3	3490	692
159	3	11547	7466	4	30178	7845	3	4545	1146
169	3	15191	11478	4	46381	18551	3	6315	1613
179	3	19738	17134	4	46060	17502	3	8298	2025
189	3	25234	28263	4	91793	54655	3	10697	2635
199	3	31848	11.24H	4	134159	1.47D	3	13772	1H
209	3	39800	16.36H	4	97834	13.19H	3	17431	1.32H
219	3	49,134	1.11D	4	184,516	2.92D	3	29,856	2.81H
229	3	60,261	1.56D	Ran out of memory			3	25,847	2.60H

range 79-299 equations in 79-299 variables respectively. The first column "Sys" denotes the number of variables and the number of equations for each system. The highest degree of the elements of the system that occurred during the computation is denoted by "D". The used memory in Megabytes and the execution time in seconds can be found in the columns represented by "Mem" and "Time" respectively except for bigger systems it is in hours (H) or days (D). In both tables we can see that MXL2 always outperforms both versions of Magma's F4 in terms of memory and time.

Table 4 shows the required rounds of solving an Poly-Dragon instance of $n = 259$ using MXL2. The columns are represented as the same as in Table 1. From Table 4 we can see that MXL2 can solve the Poly-Dragon instance with 259 variables in 7 rounds. In the first round of the algorithm, there was no dependency in the original 259 polynomials and no mutants were found. Therefore, MXL2 extended the system partially to generate new 258 cubic equations. In

Table 3. Performance of MXL2 versus F4 for Poly-Dragon

Sys	$F4_{v2.13}$			$F4_{v2.16}$			MXL2		
	D	Mem	$Time$	D	Mem	$Time$	D	Mem	$Time$
79	3	488	34	4	301	26	3	224	31
89	3	841	82	4	1519	153	3	285	39
99	3	1360	164	4	2883	320	3	454	71
109	3	2093	328	4	2039	241	3	674	117
119	3	3103	622	4	6873	972	3	1473	347
129	3	4475	1194	4	11542	1899	3	1573	312
139	3	6277	2113	4	8701	1238	3	2253	451
149	3	8606	3686	4	24105	5397	3	3151	659
159	3	11546	6645	4	30177	6734	3	4318	960
169	3	15195	10451	4	47787	14800	3	5812	1449
179	3	19741	15801	4	46064	14122	3	7698	1907
189	3	25262	25386	4	91720	41805	3	14154	3782
199	3	31852	40618	4	134144	124278	3	12944	3633
209	3	39813	64753	4	97898	69144	3	16472	6730
219	3	49129	85635	Ran out of memory			3	20736	8165
229	3	60231	1.83D	Ran out of memory			3	36617	4.13H
239	3	73006	2.36D	Ran out of memory			3	31922	3.62H
249	3	87,908	3.42D	Ran out of memory			3	39,098	4.34H
259	3	105,012	4.15D	Ran out of memory			3	47,512	6.51H
...	.	.....	.....	.	.....	.....	.	.....	.....
299	Ran out of memory			Ran out of memory			3	95,317	11.28H

Table 4. MXL2: Results for Poly-Dragon-259Var

Round	ED	Matrix	Rank	#Mutants	#LM	#Uni
1	2	259 × 33671	259	0	0	0
2	3	517 × 2862727	517	0	0	0
3	3	67340 × 2895880	67339	776	258	2
4	2	1035 × 33671	792	256	256	138
5	2	1033 × 33671	895	118	118	0
6	2	14937 × 33671	7140	1	1	1
7	2	7140 × 33671	7021	118	118	118

the second round, after applying the *Echelonize* step to the extended system (517 equations), all the equations were independent and there were no mutants found. The MXL2 extended the system again by applying *Extend* step to generate 66823 new cubic equations. By echelonizing the resulting extended system (67340 equations), we obtained a system of rank 67339, 518 quadratic mutants, 258 linear mutants in which 2 equations are univariate. After simplifying with the two univariate polynomials and modifying the elimination degree to two, we obtain a quadratic system of (1035 equations). Then third round is finished. In the fourth round, echelonizing the system of 1035 equations at degree two,

Table 5. F4: Results for Poly-Dragon-259Var

Step	SD	Matrix	#Pairs
1	2	259 × 33671	251
2	3	67349 × 2895880	4694
3	2	34446 × 33671	777
4	3	2835604 × 2832190	20423

yielded a system of rank 1033 and 256 linear mutants, 138 out of them are univariate. Substituting with the 138 univariate equations and eliminating, we obtained a system of rank 895 and 118 linear mutants in round 5. The necessary number of mutants that are required at this round is 250. Therefore, all the 118 linear mutants are multiplied by variables using *MultiplyMutants*.In round 6, we obtained 1 linear mutant which is also univariate polynomial from eliminating the extended system of total 14937 equations and rank of 7140. In round 7, after substituting with the univariate equation, we started with 7140 equations and the elimination degree is the same, 2. We obtained a rank of 7021 and the rest 118 univariate equations after applying the *Echelonize* step.

Table 5 shows the required steps of solving the same Poly-Dragon instance as in Table 4 using F4 version (V2.13.10). In each step, we show the step degree (SD), the matrix dimensions (Matrix) and the number of pairs (#Pairs).

Experimentally, as far as we noticed, the main new feature of Magma's F4 version (V2.16-1) is that if there exist some predetermined number of linear polynomials, at certain degree, the program is interrupted and then a new phase is started with extended basis by adding the linear polynomials to the original

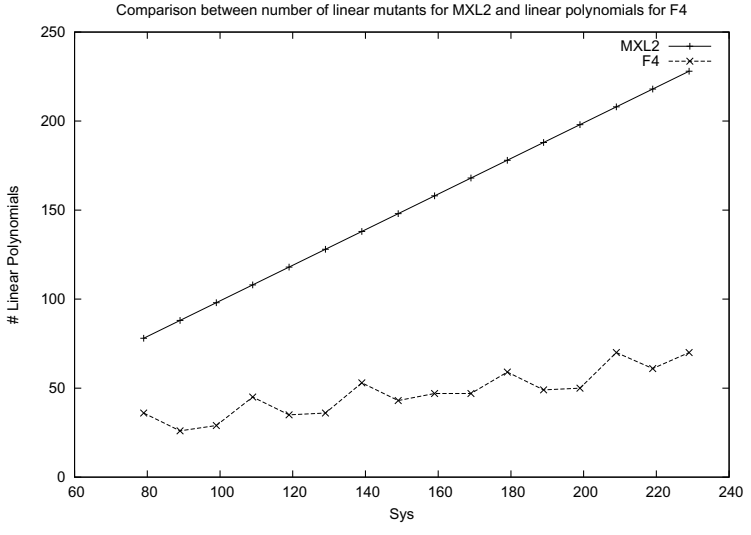

Fig. 1. LD2:Number of linear Mutants for MXL2 and linear polynomials for $F4_{v2.16-1}$

ones then compute a Gröbner basis to the extended new system. Figure 1 shows a comparison between the number of linear mutants that are generated at degree 3 and the number of linear polynomials generated by F4 (V2.16-1) at degree 3 for each LD2 system. These mutants as well as linear polynomials generated by F4 show that there is an algebraic hidden structure in the LD2 scheme that distinguishes it from a random system.

These predetermined number of linear polynomials at which Magma's F4 version (2.16-1) interrupts the first phase of computation is not enough to finish computing a Gröbner basis at the same degree at which these linear polynomials appear. The usage of the necessary number of mutants improvement for MXL2 could help new Magma's F4 to recover its defect.

6 Linearization Equations for the Dragons

The authors of [4] and [6] point out in [6] that both LD2 and Poly-Dragon possess high order linearization equations (second order in the case of LD2). Recall that a (high-order) linearization equation in plaintext variables X and ciphertext variables Y is a polynomial $F(X, Y)$, linear in the X-variables. If one is able to obtain such polynomials relating inputs and outputs of an MPKC, an attack can be undertaken by plugging in known ciphertexts and solving linear systems in the plaintext variables. Using notation of Sections 2 and 3, Poly-Dragon possesses linearization equations of the form

$$(z + u + 1)(z^{2^m} + z + \alpha + 1) + (z^{2^m} + z + \alpha + 1)^{2^m} = 0, \tag{5}$$

where $z = (L_\beta(v)+\gamma)^{2^m-1}+Tr(v)$. For the case of LD2 the equations are simply

$$(v + u + 1)(v^{2^m} + v + \alpha + 1) + (v^{2^m} + v + \alpha + 1)^{2^m} = 0. \tag{6}$$

Since equations (6) are only of degree 2 in ciphertext for LD2, a linearization attack is very feasible. The authors of [6] claim that the degree of (5) in ciphertext variables is too high to be used in a practical linearization attack.

It is interesting to note that even existence of equations (6) and (5) for LD2 and Poly-Dragon respectively does not explain the nice behavior of the direct attack using MXL2. Namely, the fact that we are able to solve at degree 3. This is unlike the classical attack on Matsumoto-Imai scheme. There it is possible to obtain linearization equations that are also linear in the ciphertext variables. Considering that ciphertext variables depend explicitly on the plaintext variables via quadratic equations, it is then clear why one is able to solve at degree 3 with XL and Gröbner basis techniques: one simply plugs in public key equations in the linearization equations and obtains that such degree-3 equations in the plaintext variables lie in the linear span of degree 3 XL-extension of the public key with the ciphertext variables fixed, see [18]. In the case of LD2 we have a mixed system, therefore there are no explicit quadratic relations between a ciphertext and a plaintext. Even if they existed, they would provide equations of degree 4 in the plaintext, which by no means explains solving at degree 3. Even "worse"

is the situation for the Poly-Dragon. This cryptosystem seems to be immune to linearization attacks, but is very susceptible to algebraic attacks.

There is a simple known way of obtaining linearization equations. One assumes that equations of the form $F(x_1, \ldots, x_n, y_1, \ldots, y_n) = \sum_i a_i x_i f_i(Y) + \sum_i b_i g_i(Y) + c$ with $Y = (y_1, \ldots, y_n)$ exist with certain requirements on degrees of f_i, g_i. Then by plugging in known plaintext/ciphertext pairs for $X = (x_1, \ldots, x_n)$ and Y one tries to find coefficients for the polynomials f_i, g_i as well as coefficients a_i, b_i, c. One should be lucky to get enough linearly independent equations relating the above coefficients. In practice this method works pretty well and has practical complexity polynomial in n. We present next a method of finding linearization equations based on Gröbner bases.

Proposition 3. *Let I be an ideal in the ring $\mathbb{F}_2[X, Y], X = (x_1, \ldots, x_n), Y = (y_1, \ldots, y_n)$. Let $D > 0$ be an integer. Let G be a Gröbner basis of I w.r.t weighted degree lexicographic ordering with $x_1 > \cdots > x_n > y_1 > \cdots > y_n$ with weights D for each X-variable and weight 1 for each Y-variable. If in I there exists a linearization equation $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = \sum_i a_i x_i f_i(Y) + \sum_i b_i g_i(Y) + c$ with $\deg(f_i) \leq D, \deg(g_i) \leq D \ \forall\ i$, then there is an element $g \in G$ which is also a linearization equation with leading monomial dividing the leading monomial of f: $lm(g)|lm(f)$.*

Proof. First of all, since G is a Gröbner basis of I and $f \in I$ there exists $g \in G$ such that $lm(g)|lm(f)$. From the form of f and the monomial ordering on $\mathbb{F}_2[X, Y]$ that we have chosen, we see that $\deg(g) = \deg(lm(g)) \leq \deg(lm(f)) = \deg(f) \leq 2D$, where by degree we mean weighted degree. If g has a monomial of the form $x_i x_j$ for some i and j, then weighted degree of this monomial is $2D$ and such monomial is larger than $lm(f)$ w.r.t weighted degree lexicographic ordering and so is larger than $lm(g)$, which yields a contradiction with the fact that $lm(f)$ is linear in the X-variables. Similarly one comes to a contradiction with other monomials that are non-linear in X-variables. This means that g is linear in the X-variables and so is a linearization equation in X.

So with the use of the above proposition we are able to get "basis" elements for the linearization equations in I. The method requires finding a Gröbner basis w.r.t a certain monomial ordering, therefore has higher complexity than the method described earlier. Still the latter method does not involve any probabilistic arguments and does not depend on a choice of plaintext/ciphertext pairs. It is a matter of future work to realize if such a method has practical implications on MPKCs.

7 Conclusion

We present an efficient algebraic cryptanalysis for the Little Dragon Two, LD2, and Poly-Dragon public-key cryptosystems. Both cryptosystems were proposed as efficient and secure schemes. In our attack we are able to break LD2 with key length up to 229 bits and Poly-Dragon with key length up to 299 bits using

both Magma's F4 and MXL2. In all experiments, MXL2 outperforms the used versions of Magma's F4. We realized that the last version of Magma's F4 is not so well suitable for solving LD2 and Poly-Dragon systems.

In MXL2 algebraic attack, the LD2 and Poly-Dragon schemes are solved at degree three which reflexes the weakness and contradicts the security claims for these two schemes. We expect that MXL2 can attack a system that represent Little Dragon Two up to 389 variables in less than one day using the same memory resources that we have, 128GB memory. We claim also that MXL2 can solve a systems that represent Poly-Dragon up to 339 variables in less than 20 hours.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
2. Ding, J., Gower, J.E., Schmidt, D.: Multivariate Public Key Cryptosystems (Advances in Information Security). Springer, New York (2006)
3. Ding, J., Yang, B.Y.: Multivariate Public Key Cryptography. In: Bernstein, D.J., et al. (eds.) Post Quantum Cryptography, pp. 193–234. Springer, Heidelberg (2008)
4. Singh, R.P., Saikia, A., Sarma, B.K.: Little Dragon Two: An Efficient Multivariate Public Key Cryptosystem. International Journal of Network Security and Its Applications (IJNSA) 2, 1–10 (2010)
5. Jacques, P.: Asymmetric Cryptography with a Hidden Monomial. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 45–60. Springer, Heidelberg (1996)
6. Singh, R.P., Saikia, A., Sarma, B.: Poly-Dragon: An efficient Multivariate Public Key Cryptosystem. Cryptology ePrint Archive, Report 2009/587 (2009), `http://eprint.iacr.org/`
7. Ding, J., Buchmann, J., Mohamed, M.S.E., Moahmed, W.S.A., Weinmann, R.P.: MutantXL. In: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China, pp. 16–22. LMIB (2008), `http://www.cdc.informatik.tu-darmstadt.de/reports/reports/MutantXL_Algorithm.pdf`
8. Mohamed, M.S., Ding, J., Buchmann, J., Werner, F.: Algebraic Attack on the MQQ Public Key Cryptosystem. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 392–401. Springer, Heidelberg (2009)
9. Lidl, R., Niederreiter, H.: Finite Fields, 2nd edn. Encyclopedia of Mathematics and its Applications, vol. 20. Cambridge University Press, Cambridge (1997)
10. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: MXL2: Solving Polynomial Equations over GF(2) Using an Improved Mutant Strategy. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)
11. Courtois, N.T., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
12. Ding, J.: Mutants and its Impact on Polynomial Solving Strategies and Algorithms. Privately distributed research note, University of Cincinnati and Technical University of Darmstadt (2006)

13. Ding, J., Cabarcas, D., Schmidt, D., Buchmann, J., Tohaneanu, S.: Mutant Gröbner Basis Algorithm. In: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China, pp. 23–32. LMIB (2008)
14. Mohamed, M.S.E., Cabarcas, D., Ding, J., Buchmann, J., Bulygin, S.: MXL3: An Efficient Algorithm for Computing Gröbner Bases of Zero-dimensional Ideals. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 87–100. Springer, Heidelberg (2010)
15. Albrecht, M., Bard, G.: The M4RI Library– Linear Algebra over GF(2) (2008), `http://m4ri.sagemath.org`
16. Bard, G.V.: Algebraic Cryptanalysis. Springer Publishing Company, Incorporated, Heidelberg (2009)
17. Wolf, C.: Efficient Public Key Generation for HFE and Variations. In: Dawson, E., Klemm, W. (eds.) Cryptographic Algorithms and their Uses, Queensland University of Technology, pp. 78–93 (2004)
18. Billet, O., Ding, J.: Overview of Cryptanalysis Techniques in Multivariate Public Key Cryptography. In: Sala, M., et al. (eds.) Gröbner Bases, Coding, and Cryptography, pp. 263–284. Springer, Heidelberg (2009)