High Saturation Complete Graph Approach for EC Point Decomposition and ECDL Problem

Nicolas T. Courtois

University College London, Computer Science, Room 6.18. Gower Street, WC1E 6BT, London, UK
n.courtois@ucl.ac.uk

Abstract. One of the key questions in contemporary applied cryptography is whether there exist an efficient algorithm for solving the discrete logarithm problem in elliptic curves. The primary approach for this problem is to try to solve a certain system of polynomial equations [46]. Current attempts try to solve them directly with existing software tools which does not work well due to their very loosely connected topology [45, 42] and illusory reliance on degree falls [37]. A deeper reflection on what makes systems of algebraic equations efficiently solvable is missing. In this paper we propose a new approach for solving this type of polynomial systems which is radically different than current approaches, cf. [45, 42]. We carefully engineer systems of equations with excessively dense topology obtained from a complete clique/biclique graphs and hypergraphs and unique special characteristics. We construct a sequence of systems of equations with a parameter K and argue that **asymptotically** when K grows the system of equations achieves a high level of saturation with $\lim_{K\to\infty} F/T = 1$ which allows to reduce the "regularity degree" and makes that polynomial equations over finite fields may become efficiently solvable.

Key Words: cryptanalysis, finite fields, elliptic curves, ECDL problem, index calculus, error correcting codes, linear codes, algebraic codes, codes on elliptic curves, Semaev polynomials, block ciphers, Simon cipher, NP-hard problems, MQ problem, overdefined systems of equations, phase transitions, XL algorithm, Gröbner bases, block ciphers, ElimLin.

Remark. This paper is work in progress, it will be be updated each time we have some new elements or have made some progress.

Table of Contents

1 A Short Explanation for Beginners

In order to explain in just a few words what we do in this paper, and what is all the fuss about, consider the following ultra simplified example. The dominant paradigm for solving the ECDL problem is to solve a certain system of polynomial equations using so called Semaev polynomials. We consider an ultra-simplified version of this problem with just two points. The attacker wants to solve the following problem, essentially nothing really more complex that:

$$P1 + P2 = Q$$

where $P1/P2$ satisfy some extra constraints, not every $P1$ is permitted. The standard method to do this which dominated in the literature is use Semaev polynomials, [46]. We bring a standard piece of software which converts our EC equations to polynomials. Then we bring another standard piece of software which **maybe** solves the equations with a so called Gröbner basis computation.

This wishful thinking approach has been a miserable failure [31, 37, 42]. One crucial problem is that the complexity of this process depends on a certain integer called "regularity degree" cf. [2] and would be solved in polynomial time if this integer was a small constant independent on input size n. Actually exactly the contrary happens, cf. [37], we hit the wall, for example some of the most recent attacks of this form have running times growing exponentially cf. [42].

The regularity degree however is NOT an absolute limitation in algebraic cryptanalysis, cf. [9]. The attacker needs to work on **reducing** this degree. And this is in fact very easy, not harder than in [9]. For example consider the following problem:

$$\begin{cases} P1 + P2 = Q \\ P1 + D = P1' \\ P2 + E = P2' \\ P1' + P2' = (Q + D + E) \end{cases}$$

Here Q, D, E and $Q + D + E$ are constants. Now we run our two pieces of software again. Oops, **the degree of regularity has decreased.** Maybe, or so we claim[1]. As simple as that. And we can iterate this simple expansion process at will. We do no longer have an obscure "plug and pray" approach in which some mysterious things such as degree falls might happen, but we **explicitly engineer** a new system of equations which is easier to solve. Specialists will say: we applied a "Linear ECC Code" expansion to reduce the "regularity degree".

We could stop here, claim that we have broken the EC discrete log and let the reader figure out all the little details. However there is a considerable amount of little details hence we wrote a longer paper. Moreover there is a very serious flaw in all the above methodology which basically **cannot work** as presented. Further reading will allow the reader to discover what the problem is.

[1] This is essentially because we have added 2 new variables and 3 new equations. Our system of polynomial equations will now be more **overdefined**, cf. [6]. And this is expected to make it "easier" to solve cf. [6, 7, 2, 1, 21, 5, 19, 8, 9]. The new constant $Q + D + E$ produced by the EC group law acts as new "non-trivial" information coming from an oracle.

Part I

Introduction

2 Introduction - Index Calculus and EC Crypto

At Crypto 1985 conference Victor Miller [40] have suggested that:

> It is extremely unlikely that an index calculus attack
> on the elliptic curve method will ever be able to work.
> V. Miller: "Use of elliptic curves in cryptography", Crypto'85, [40].

For the last 30 years this sort of impossibility, or a belief in impossibility, has been the basis on which the security of elliptic curve cryptosystems was based. More recently a number of papers in cryptography have attempted to nevertheless construct an index calculus for the EC discrete log problem. The first attempt in this direction is the 2004 paper by Semaev cf. [46]. The suggested method is to try to split a given point on an elliptic curve into a sum of several points lying in a certain sub-space. This is expected to be achieved through solving a system of polynomial equations derived from so called summation polynomials, cf. [46]. More recently several authors have suggested that this sort of method could eventually lead to sub-exponential algorithms at least for the binary elliptic curves [45, 43, 30, 33, 31]. Public confidence in the security of ECCs have suffered a sudden and unexpected blow in late 2015, when the NSA have all of the sudden decided that standard ECCs which everybody on this planet uses are NO longer secure, for reasons which frankly no researcher in cryptography can explain in a satisfactory way, cf. [35].

2.1 Current Algebraic Approaches

All current attempts to solve the ECDL problem operate by writing some systems of equations and trying to try to solve them directly (write and solve) with existing software tools such as Gröbner basis algorithms and sometimes other techniques such as SAT solvers. All these are very quite obscure software tools and nobody quite understands how, why and when they might work well. Not enough attention is paid to the question of what makes systems of algebraic equations efficiently solvable in the first place.

It is easy to see that all current attempts are deeply flawed on two accounts.

First of all, they do not work well due to their very loosely connected topology [45, 42], with a risk that Gröbner basis algorithms will never succeed to create equations which relate various variables in the system, and this probably explains the excessively poor scalability of results in all numerical examples published to date cf. for example the simulations in [42] which exhibit very fast exponential-like growth in the running times.

Secondly, we have this naive assumption that when degree falls occur, a certain system of equations might collapse and will be solvable, without too much additional effort. This sort of assumption definitely works in some cases and the system of equations is solved completely [4, 39, 37] but sometimes the process may run out of steam after making a lot of progress [8, 9, 5, 44]. Moreover this assumption is simply wrong and it is simply not true in general, cf. [37].

2.2 Binary vs. Mod P Curves

Interestingly, in the original paper from 2004, a variant of this method was also proposed for the "ordinary" elliptic curves modulo p, which is the main form of elliptic curve cryptography which is used in the real-life systems, for example in TLS, SSH, bitcoin etc.

Even though there is no evidence that any of the attacks proposed for binary curves actually works [31, 37], recently researchers have tried to propose some innovations which make attacks for mod P curves more plausible.

A very recent paper by Kosters, Petit and Messeng [March 2016] proposes a new method to encode the problem of DL in elliptic curves designed specifically to work well for curves such that $P-1$ has many small factors which is the case for standard NIST curves for example P-224, cf. [42]. Another method based on EC isogenies designed to work for arbitrary P is also proposed and tentatively implemented.

2.3 Poor Results

Both methods suffer from a serious problem which is the same as in all works in this space published in the last few years: there is no evidence that the method works at all except on very small examples[2]. In fact recent works very clearly explain that [first] degree falls do NOT guarantee that system of equations becomes efficiently solvable [37] and all recent papers on this topic require this assumption to produce an attack on ECDL problem. In general it is quite interesting to see that all these works speculate about how to solve ECDL problem using a point-splitting step, however no point-splitting of any sort was ever demonstrated in practice for any realistic elliptic curve. On the contrary, there is a very substantial qualitative and quantitative gap between what kind of toy attacks can be demonstrated in practice with current tools and what would be needed to break any even remotely realistic form or elliptic curve cryptography.

2.4 What Is New

In this paper we propose a different approach, unlike any approach yet proposed in the literature. We postulate that in contrast with systems of equations over $\mathbb{F}_2$ already massively exploited in cryptanalysis [6, 7, 2, 1, 4, 21, 5, 19], one needs to work A LOT harder in order to produce a system of equations over a larger finite field which can be solved. Our approach is to generate a sequence of systems of equations which are increasingly more and more overdefined up to a point of "saturation" where they could have better[3] chances of becoming efficiently solvable.

[2] In fact from simulations in Sec. 4.2. of [42] we can deduce that the method does not work, as the time complexity seems to grow faster than 2^n.

[3] At least better than in any previous approach. It is possible to see that even low degree and very highly saturated equations over both $\mathbb{F}_{2^n}$ and $\mathbb{F}_p$ fields can still be very hard to solve, at least on contrived examples, for example when they have many solutions, and handling these multiple solutions is a major difficulty (here and elsewhere).

The crucial question we are going to study is first to demonstrate the existence of new polynomial equations which are new, remarkably simple, and NOT contained in the ideal or Gröbner basis generated by the Semaev polynomials in previous approaches and then show HOW they can be used to construct certain **very highly overdefined systems** of equations with unique strong characteristics of high connectivity, density and interesting asymptotic properties.

3 The Road Map

Before we design our new attack which allows to split a point on the elliptic curve, we are going to explain what kind of characteristics we would like our attack to have. Our main strategy can be described in terms of what we don't want to have and a list of goals for an alternative approach.

1. We do not want to write a certain system of equations and hope it is efficiently solvable which is a frequent and major difficulty in algebraic cryptanalysis [46, 33, 45, 42, 24, 22].
   We want to engineer/contruct systems of equations which are made to be efficiently solvable.
2. We don't want to deal with a question whether a certain system of equations can be solved by Gröbner bases F4, XL or FXL after expansion to a certain degree. This degree is typically called a "regularity degree", cf. [2, 1]. We don't want to worry how quickly this degree will grow when the parameter n grows[4].
3. We do not want obscurity, we want clarity.
4. We don't want degree falls just to happen, maybe accidentally.
   We want to explicitly design, generate, enumerate/count and analyse the degree falls.
5. We would like to construct a system of equations where equations are quadratic or cubic, and they remain quadratic at every stage, and they are solved at degree 2 or 3. Or that they are eventually solved at degree $2 + \varepsilon$ where ε is small.

This sort of strategy is not new. For example it is very clearly proposed in slides 82-86 of [19]: avoid expansion stage[5]. This approach can be seen as one of the several possible so called "Fast" algebraic attacks strategies, cf. slide 84 in [19] and slide 121 in [22].

[4] Many previous work have naively assumed that this degree is fixed, cf. for example [33, 34, 45], while in fact it increases with n.

[5] Such "expansion" stage is really the essence of all XL, mutant XL, F4, and other Gröbner basis methods, cf. slides 72-75 in [22] and [1, 3, 6, 2]. .

3.1 Additional Postulates

We further postulate that:

1. Experience shows that sometimes we can work with systems of equations which are larger than initially yet they can be overall easier to solve, cf. for example [8, 5, 19].
   One major way to achieve this is to add new variables, which is major paradigm shift compared to methods which are primarily used in this space, and a major general direction of work, cf. Section 3.5. In particular, a major family of techniques in this space are the [linear] **ECC Codes**, cf. Section 3.8 below.
2. Working at degree 2 makes it potentially a lot easier to understand if our system of equations is efficiently solvable or not.
   Either it generates enough linear equations with ElimLin [5] or maybe it can be solved by some more advanced method[6], which operates without increasing the degree, or maybe it is hard to solve.
3. Our complexity claims should be based on a number of facts which highly regular, highly predictable and possible to validate experimentally for a wide range of parameters.
4. It should not be possible to exhibit a counter-example of a quadratic or cubic system of equations with similar characteristics where our method fails.
5. We will try to be conservative in our estimations and we will try to make our lives harder in order to obtain solid and robust attacks.
   The approach of this paper is conservative and could possibly later be discovered to be an overkill, with respect to events such as additional degree falls to make such attack work easier than expected. We simply do not want to rely on events we don't understand.

3.2 On F/T Ratio, Equations Topology Density and Connectivity Criteria

In this paper we propose another major general paradigm or philosophy which allows to achieve the objectives above in a quite specific way.

1. We want to design systems of equations which are able to achieve certain characteristics in terms of density, topology, and connectivity where the equations are viewed in terms of graphs or hypegraphs in which various types of equations connect different types of variables.
   We want to have high connectivity not only in general or in average case, but for all non-linear monomials we use.
2. We want to obtain equations which are very largely overdefined [6].
   More precisely we would like to have a very large R/T ratio [21, 24, 22] or the F/T ratio, where F is the the number of linearly independent equations.

[6] It could be for example a T' method cf. [22] slides 95-101 which was introduced at Asiacrypt 2002 [24] which however works well primarily for systems of equations over $\mathbb{F}_2$.

We will frequently postulate that systems with the characteristics above plus some additional conditions are likely to become efficiently solvable. However frequently it will not yet sufficient to obtain equations which in fact ARE solvable. This is major difficulty in ECDL research which no one was yet bee able to solve in a satisfactory way. We will go back to this question in Section 6.7 and few more times elsewhere.

3.3 Postulates on Constraints / Factor Basis Question

Furthermore we will invent ourselves a lot more postulates to satisfy much later in this paper, in particular in Section VI we have two main postulates F1-F2 and 14 additional more detailed postulates.

3.4 The Tale of Two Parameters

In all our new methods described in this paper there are two parameters, n which is the size of the initial problem, and a parameter K which we will chose to be sufficiently large so that our system of equations becomes efficiently solvable at degree 2 or 3 and by a certain specific method.

Moreover if for a certain value of parameter K the attack does not work, the attacker can increase the value of this parameter. This parameter K should also offer some extra flexibility which allows to optimize the attacks and test different software solvers.

Subsequently we are going to analyse how large K needs to be when n grows and argue that when n is fixed and K increases the number of newly generated independent equations will grow asymptotically quite fast.

This will be **a lot faster** than the number of [new+old] variables, which will make our equations quite massively overdefined. In fact the number of equations R will grow either roughly at the same speed and in some cases even faster(!), than the number of newly introduced monomials [7] T. Thus we will be able to produce a very highly overdefined system of equations with high R/T ratio but also with high density/connectivity characteristics. We will argue that modulo some important and strictly necessary requirement to have a limited number of solutions, such systems are bound to easily solvable in polynomial time. Similarly we will then consider that in order to achieve a “saturation” or “phase transition” point it should be probably sufficient for K to grow polynomially in n.

Remark. It is probably NOT necessary for K to be particularly large. It appears that for most properties we consider in this paper are such that for sure they do not work at all for very small K, and they should work starting from a certain threshold, rather than K must be really large. At least we hope so.

[7] In this paper, by convention, we do not count the constant monomial in T.

3.5 On Two Major Philosophies In Solving Non-Linear Equations

There are two major philosophies in algebraic cryptanalysis and for the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Either we expand the number of variables.
2. Or we expand the number of monomials.

Both types of methods already existed and both philosophies worked quite well in their own (somewhat disjoint) space in algebraic cryptanalysis of DES [5]. Both have also been studied for solving systems of polynomial equations over finite finite fields at Eurocrypt 2000 [6]. we have re-linearization technique vs. XL algorithm, cf. [6]. At Eurocrypt 2000 it was concluded that re-linearization technique is highly redundant and that XL works better [6]. Then we discover that at higher degrees XL is also redundant[8].

Let us also recall what is the main working principle in both types of techniques: we make two values grow, yet one grows faster. We will see a lot of examples of this in this paper. It allows one to understand why both families of techniques may and will work.

1. When we add **new monomials**, we grow both the number of monomials T and the number of new equations R. For any system of equations with a certain R/T we can easily improve the R/T ratio by increasing the degree. Then R grows **faster** because there are several ways to obtain the same monomial, cf. slide 80 of [22].
2. When we add **new variables** we also grow both the number of monomials T and the number of new equations T. Here is also R can grow **faster** and sometimes even asymptotically faster than T, which is demonstrated in the present paper, cf. for example Thm. 12.3.1 page 41.

It is important to see that techniques of type 1. expand monomials are nowadays standard, well studied, fully automated by software and do not require a lot of attention. The second family has not been sufficiently studied. It gives the code breaker a **very considerable degree of freedom** which is actually a big a problem: it is not clear **how to even start** to design an attack based on this idea. One major family of techniques are the **ECC Codes**, cf. Section 3.8 below.

It is important to note that both approaches 1. and 2. can and should be combined. To put it simply, the second approach makes the first approach work better, equations become more overdefined and the so called degree of regularity is expected to decrease, cf. Section 1. In this paper we are going to propose several new techniques in this space and we postulate that just by applying

[8] It is in general difficult to remove ALL redundancies in linear dependencies in expanded equations. cf. [3, 24, 25, 8, 9, 7, 1]. Gröbner basis techniques are precisely about removing even more redundancies in XL, cf. [1, 2]. However redundancies are NOT necessarily a problem. Our experience shows that simpler or alternative approaches (decimation: erasing a subset of equations) are usually equally fast and may also use less memory [when equations are very sparse] than some advanced Gröbner basis techniques such as F5.

the EC code technique **the regularity degree can be reduced**[9] **down to a really low value such as 2 or 3**. The crucial question here is the study of methods to achieve this. We would like to achieve some sort of combinatorial explosion in the equations or/and degree falls generated, similar as in ElimLin algorithm. One methods to achieve starts with a construction where $R/T \to \infty$, similar to super-linear growth for linear variables observed in ElimLin algorithm, not a fiction, cf. later Fig. 1. This will in many interesting cases lead to $F/T \to 1$, where $F \leq R$ is the number of equations which are actually linearly independent and not redundant. In this paper we present several new non-trivial techniques to achieve this objective.

3.6 On Equivalence of Two Approaches/Philosophies

One of the key points in the aforementioned Eurocrypt 2000 paper is that the approach with adding extra variables can be reduced to the XL-like approach, cf. [6]. We expect that it is the same here and that we have either equivalence of the two approaches or a semi-efficient reduction in at least one direction. Here are two examples of general results which show that the two approaches are closely related.

Example 1 - Univariate Polynomials

Theorem 3.6.1. Informally, for one of the natural linear EC code with expansion factor being at least K and which we define later and use extensively in this paper, and **for every degree N** $\in \mathbb{N}$, there exists an equation of type: for every $K' > K_0 \in \mathcal{O}(N)$

$$\begin{aligned} & x1^N \text{AffineSum(up to } K' \text{ variables excluding } \; x1) = \\ & \sum (\; \text{monomials of degree up to 2 excluding } \; x1) \end{aligned}$$

The particular form of this equations becomes easier to understand if we remark that this equation becomes quadratic if we consider that $x1^N$ is a new variable. We refer to Thm. 16.0.2 for a more precise formulation and in Section 14.3 we give an example of how such equations can be generated by explicit closed formulas for specific ECC Codes we use.

Example 2 - Multivariate Polynomials

Another nice result in this direction is outlined in Section 15 for arbitrarily high degrees and studied in detail for cubic monomials in Section 11.2. The main result is quite strong: every monomial of any degree can be eliminated by a certain type of equation. More precisely for EVERY number of distinct variables in our linear ECC Code there exists a unique equation such that it contains this product and all the other monomials are of lower degree and their number is linear [10] in the degree. In particular we have MQ3 equations and Thm. 13.9.4 for eliminating arbitrary cubic monomials and Thm. 11.2.1 contains explicit formulas which allow one to do so.

[9] This however has a price, a lot more new variables and new equations are generated.

[10] Not exponential as the reader might expect and not even polynomial. It is linear, cf. Thm. 15.1.1 in Section 15.1.

3.7 Limits of The Equivalence

It is possible that the equivalence will work only in one direction, and the other direction will not be as efficient. Moreover even one direction is not going to be efficient in our opinion, or would require one to develop a lot of extra highly specialized code to integrate in existing Gröbner basis software.

The key point is that our work contains many very specialized equations and though these equations are in theory equivalent ot some degree falls or mutants which could appear in a Gröbner basis computation, we can generate them DIRECTLY, without lengthy computations with long polynomial equations, due to existence of specialized formulas [dedicated algebraic shortcuts]. Moreover these equations can greatly simplify arithmetic in the ring multivariate polynomials modulo the ideal generated by our equations. Essentially some monomials can be replaced directly by short sums of other simpler polynomials. An excellent example which illustrates this principle is again the Example 2 above, each monomial with 3 variables can be replaced by a simple quadratic polynomial which is computed using a direct explicit formula given in Section 11.2.

Remark. These dedicated algebraic shortcuts are not equivalent to precomputations in algebraic cryptanalysis, because these formulas are generic and exist in vast numbers of instantiations. Accordingly they do NOT require storage or access to pre-computed tables, in order to be used for eliminating various degree 3 monomials on the fly when required.

3.8 On ECC Codes and Related Literature

The philosophy of adding new variables and in particular THE particular way in which WE understand and implement this philosophy in this paper, can be described as, and can be explained in terms of certain types of **EC codes**.

Definition 3.8.1 (ECC Code).

We call an ECC Code any injective application

$$F : E(\mathbb{F}_p)^L \to E(\mathbb{F}_p)^K$$

which is defined for all except a small number of special EC points.

Remark: We should note that error correcting codes which are defined or constructed using elliptic curves are typically defined as subsets of $\mathbb{F}^K$ where $\mathbb{F}$ is a finite field, cf. for example page 11 in [36]. In this paper we find it more convenient to define ECC Codes as a subset of $\mathbb{E}^K$ where $\mathbb{E}$ is an elliptic curve, even though later we will just look at EC coordinates of these points in $\mathbb{F}^K$. We refer to [36] for additional literature pointers about error correcting codes and those which use elliptic curves.

ECC Code Decoding Problems vs. Our Point Decomposition Problem

Many problems which we study in this paper and related literature are closely related and can be seen as certain types of variants of traditional **decoding problems** for special types of ECC Codes which are **linear over the EC** but not linear over $\mathbb{F}_p$.

In fact this whole paper and many other papers on the same topic can **all** be seen as papers about decoding certain particular families of ECC Codes.

For example if we want to split a point in 2 $Q = P1 + P2$ with $P1, P2 \in$ a certain subset of $\mathbb{F}_p$, we can define an ECC Code as follows:

$$(x, y) \mapsto [\ (x, y),\ Q - (x, y)\ ]$$

And the goal of the attacker will be to find a small codeword of type

$$[\ (x, .),\ (x', .))\ ]$$

such that x and x' belong simultaneously to a certain subspace, for example such that, as proposed in early work of Semaev in [46], we have simultaneously:

$$x < P^{\frac{1}{2}}\ \ AND\ \ x' < P^{\frac{1}{2}}$$

Remark 1. In both cases, in error correcting codes and here, the goal can be potentially the same: find a codeword under the constraints above using some fast/efficient decoding algorithm, which will give a fast point splitting algorithm.

Remark 2. A key advantage here is that the attacker can freely chose the metric by which this decoding can be done. Solving this decoding problem could be done for more or less any[11] achievement metric, or any fixed subset where the target points are expected to belong. Any such subset, which could be called a "factor basis", is expected to lead to a certain type of an index calculus algorithm for the ECDL problem. We refer to [42] for explanations regarding on how the point splitting problem relates to the ECDL problem.

Remark 3. We refer to [17] to see that NOT all point splitting problems lead to efficient algorithms for solving the ECDL problem.

[11] For as long as it does not depend too much on particular points we want to split.

4 Example to Imitate - ElimLin Connection

There is precedent for the sort of attack we are looking for. It is the behavior of ElimLin algorithm in block cipher cryptanalysis [5, 22, 8, 9, 12]. It is an incredible landmark result in cryptanalysis showing how the "regularity degree" is not at all a limitation in algebraic cryptanalysis, or how it can be defeated by the attacker, and it can be reduced to the lowest possible value with very little effort.

ElimLin is a curious sort of attack, cf. slide 126 in [22]. It can be described informally in 2 simple steps:

1. Find linear equations in the linear span.
2. Eliminate some variables, and iterate (try 1. again).

ElimLin is a stand-alone attack which allows one to recover the secret key of many block ciphers [8, 9, 13, 23] and more recently in [26, 44, 27].

The main characteristic of ElimLin is that it quietly dissolves non-linear equations and generates linear equations. This algorithm basically makes progressively disappear the main and **the** only thing which makes cryptographic schemes not broken by simple linear algebra: non-linearity. It is not clear however why this works and how well the ElimLin attack scales for larger systems of equations. For example in recent 2015 work of Raddum we discover that (experimentally) ElimLin breaks up to 16 rounds of Simon cipher [44] however it is hard to know exactly what happens for 17 rounds.

4.1 A Surprising Phenomenon within ElimLin

Here is a simple observation which is is crucial for understanding why ElimLin algorithms does eventually work and is able to cryptanalyse many block ciphers.

Conjecture 4.1.1. Consider a system of multivariate equations derived from a block cipher written following one of the two basic strategies described in [5, 22]. Consider a simple known plaintext attack with K Plaintext/ciphertext (P/C) pairs. Consider a case such that the cipher is broken by ElimLin, cf. [8, 9, 13, 23, 44]. The number of new and linearly independent linear equations generated by ElimLin algorithm grows faster than linearly with K until it reaches a saturation stage where the cipher is broken by ElimLin.

This claim may seem to be too good to be true. Therefore we are going to illustrate it with a real-life examples extracted from the work of our students and published in [12].

4.2 On Asymptotic Behavior of ElimLin when K Grows

One (old) example which shows that the number of equations grows faster than linear as a function of the data complexity K in ElimLin can be found at slide 153 in [22] which example is from 2006-7 and originally comes from [13].

More examples can be easily obtained using a basic software setup which we use at UCL to run a hands-on student lab session on algebraic cryptanalysis of block ciphers [23], which is part of GA18 course on cryptanalysis taught at UCL. One example could be easily obtained for the CTC2 cipher, cf. [8, 9, 23]. A more "modern" example can be studied with the recent NSA block cipher Simon. We have used the equations generator for the Simon block cipher developed by Guangyan Song and UCL InfoSec M.Sc. student Ilyas Azeem, the complete source code of which is available at github, cf. [23, 10].

The ElimLin is executed using using one of our implementations of ElimLin [23, 10] which has the nice particularity to display on screen the number of linear equations generated at each stage/iteration of the algorithm (it is the only implementation we are aware of which displays it).

On the figure below we show the number of linear equations generated at stage 4 [counting from 0] of the ElimLin algorithm for 8 rounds of Simon block cipher. We should note that nothing remarkable happens at earlier stages 0,1,2,3, the growth is linear.

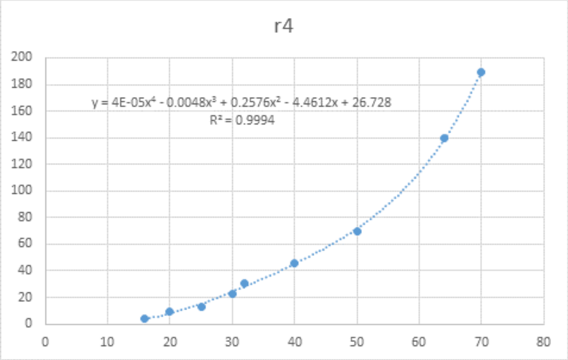

Fig. 1. Number of linearly independent equations generated at stage 4 of the ElimLin algorithm for 8 rounds of Simon 64/128 obtained with the exact software setup of [23].

This graph was generated by Iason Papapanagiotakis-Bousy, UCL Information Security M.Sc. student as part of his individual cryptanalysis project. Our work on this topic was presented at SECRYPT 2016 conference in July 2016, cf. [12]. On the picture we also provide the best polynomial approximation at degree 4 which minimizes the squared error and which was computed using Microsoft Excel. It is also noteworthy that with these techniques we have been able to produce extremely accurate predictions which give exact results up to 100 % of the time and which have R^2 close to 1.

4.3 The Unstoppable Force of An Asymptotic

Our conclusion is as follows. We observe that ElimLin breaks the cipher because the number of newly generated linear equations grows **asymptotically faster** than the number of variables. Eventually we obtain a sufficient number of linear equations which makes ElimLin compute all the variables and obtain the 128-bit secret key together with all the intermediate variables.

What's New, Growth vs. Penetration. This fact was known for at least a decade, though never yet publicly stated in this way as it seems to the author. It was previously observed in different cases where ElimLin was applied cf. slide 153 in [22] and [13] and [8, 9, 5, 23]. In the table at slide 153 in [22] we also observe that as the number of new linear equations grows, these equations penetrate more and more deeply inside the cipher for $2, 3, 5, \ldots$ rounds. We expect the same to happen for other block ciphers when ElimLin was applied. Some sort of converse is also true, it is possible to observe that for all ElimLin runs until a certain threshold K no single new linearly equation which involves the variables in the middle of the cipher is generated. The ElimLin attack really penetrates consecutive round of ciphers one by one from both sides in several clear-cut-threshold steps.

Can This Fail. We do not know any counter-example which would contradict this "super-linear" growth rule. For 8 rounds of Simon cipher we observed that the growth remains strictly linear for equations computed at stage 3, however some equations are only computed at stage 4 of the attack and this occurs quite early starting from $K = 1$. However Simon is a particularly simple cipher. For more complex ciphers, the ElimLin algorithm could have serious problems to enter this behavior or start working and exhibit any sort of non-trivial behavior. However when ElimLin starts to work for some cipher and produces new equations which depend on the plaintext or ciphertext data in a non-trivial way, and already have this "super-linear" character (quite early) it seems that nothing can stop it, we just need to increase K.

This except maybe if there isn't enough data available. A certain type of counter-example could occur for some ciphers with small blocks which would maybe not be able to allow K to be large enough for ElimLin to terminate.

Improvements. We stress that this powerful phenomenon of **combinatorial explosion** which eventually leads to ElimLin breaking the cipher occurs already for a basic straightforward application of ElimLin in a known-plaintext attack (KPA) scenario. Several methods to make ElimLin work better by using well-chosen P/C pairs. One method which has been used ever since ElimLin was invented [8, 9, 5] is to exploit a CPA (chosen-plaintext attacks) with plaintext which differ by extremely few bits, for example in a counter mode. More recently new methods have been proposed in the literature which are able to generate additional linear equations not automatically discovered by ElimLin [26] and [27]. Numerous researchers have demonstrated in their experiments that ElimLin combines very well with equations on bits which can be obtain from any of linear, differential, truncated-differential and cube-like attacks. Moreover

ElimLin benefits from such additional equations and generates typically yet more new equations.

Further Study of ElimLin. We see that ElimLin is a highly versatile approach susceptible of many advanced variants and optimizations. We provide the reader with a powerful Java tool which can inspect and analyse the equations found by ElimLin algorithm. It can be downloaded from [11]. For example this tool allows one to see that ElimLin discovers a lot more more equations than cube attacks: it generates numerous linear relations between sets of variables which are a lot less regular than in cube attacks .

4.4 Lessons Learned

Similarly as with ElimLin, we would like to design an attack on the ECDL problem with a parameter K, such that we can make our system of equations progressively "easier" to solve when we increase K.

4.5 Phase Transitions

It is known that many NP-hard problems are subject to "phase transition", with certain parameters that problem is hard, and then will rather abruptly transition from "hard" to "easy to solve". This what we observe with ElimLin here. We would like to be able to engineer such a "phase transition" for the ECDL problem in elliptic curves. This is not going to be an easy task.

5 Summation Polynomials in Elliptic Curves

Most existing works on this topic consider primarily and exclusively binary elliptic curves. We know only two exceptions to this rule: the old initial Semaev paper [46] and one of the most recent papers [42]. In this paper we consider exclusively ordinary elliptic curves modulo p such as used in real-life applications on the Internet, in payment systems such as bitcoin, etc.

5.1 Elliptic Curves Mod P

One of the most popular ways to define an elliptic curve group is to use the Weierstrass equation:

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

In addition in large characteristic > 3 it is typically assumed (without any loss of generality, cf. Section 3.1.1. in [32]) that we have $a_1 = a_3 = a_2 = 0$ and $A = a_4$ and $B = a_6$ so we get:

$$Y^2 = X^3 + AX + B$$

In contrast for typical curves in characteristic 2 we assume (following [45]) that $a_1 = 1$, $a_3 = 0$, $a_4 = 0$, and and $A = a_4$ and $B = a_6$ so we get:

$$Y^2 + XY = X^3 + AX^2 + B$$

5.2 Summation Polynomials

Now we consider a summation of 3 points on the elliptic curve:

$$(x_1, y_1) + (x_2, y_2) + (x_3, y_3) = \infty$$

We consider the case of characteristic > 3. Following [38] we have:

$$\begin{gathered} S_3(x_1, x_2, x_3) = \\ (x_1^2x_3^2 + x_2^2x_3^2 + x_1^2x_2^2) - 2(x_1^2x_2x_3 + x_1x_2^2x_3 + x_1x_2x_3^2) - d_2(x_1x_2x_3) + \\ -d_4(x_1x_3 + x_2x_3 + x_1x_2) - d_6(x_1 + x_2 + x_3) - d_8 \end{gathered}$$

where $d_2 = a_1^2 + 4a_2$ $d_4 = 2a_4 + a_1a_3$ $d_6 = a_3^2 + 4a_6$ $d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$ also following [38, 28].

Therefore when we assume that $a_1 = a_3 = a_2 = 0$ and $A = a_4$ and $B = a_6$ we get:

$$\begin{gathered} S_3(x_1, x_2, x_3) = \\ (x_1^2x_3^2 + x_2^2x_3^2 + x_1^2x_2^2) - 2(x_1^2x_2x_3 + x_1x_2^2x_3 + x_1x_2x_3^2) + \\ -2A(x_1x_3 + x_2x_3 + x_1x_2) + A^2 - 4B(x_1 + x_2 + x_3) \end{gathered}$$

Which following [45] can also be written as:

$$S_3(x_1, x_2, x_3) = (x_1 - x_2)^2x_3^2 - 2\left[(x_1 + x_2)(x_1x_2 + A) + 2B\right]x_3 + (x_1x_2 - A)^2 - 4B(x_1 + x_2)$$

5.3 Point Splitting Polynomials

In order to design an index calculus algorithm for the ECDL problem, we need to solve the problem of point splitting. Research research shows that splitting in 2 or splitting in 3 is easier than expected [17], at least for binary elliptic curves. It is easy to see that in order to obtain any improvement on Pollard's Rho complexity of $2^{n/2}$ we need to be able to split a point in at least 5.

Let $R = (R_X, R_Y)$ be the target point on the elliptic curve which we want to split in a form

$$R = P_1 + \dots P_i + \dots + P_M$$

with some M points P_i $i = 1..M$.

5.4 Point Splitting In Two Using S3

The most basic problem which we would like to be able to solve using the Semaev polynomials is the problem of splitting a point in two:

$$S_3(P_{1X}, P_{2X}, R_X)$$

where P_1 and P_2 should lie in a well-chosen subspace of size $2^{n/2}$.

5.5 General Point Splitting

We consider the problem of splitting in M points using the polynomial S_{M+1}:

$$S_{M+1}(P_{1X}, P_{2X}, \dots, P_{MX}, R_X)$$

where the P_i should lie in a well-chosen subspace of size $2^{n/M}$.

This is the initial Semaev approach from 2004 in [46].

5.6 Degree Considerations

It is possible to see that the degree of this polynomial S_{M+1} is 2^{M-1} in each variable [46] and the total degree is $(M+1)2^{M-1}$. We formalize this as follows:

Definition 5.6.1 (r-degree).
We say that a polynomial belongs to the set of polynomials of r-degree s if it is a sum of products of up to s powers $1..r$ of the individual variables.

According to this definition the Semaev polynomial S_{M+1} has 2^{M-1}-degree equal to $M + 1$.

5.7 Point Splitting In M Parts Using S3 Polynomials

More recently in 2015 a particular way to re-write this problem using only the simplest non-trivial summation polynomial S_3 and with many additional variables have been proposed by Semaev in 2015, cf. [45]. We recall this particular way to encode the problem of splitting the point on elliptic curve. We call x_i the x coordinate of point P_i in $\mathbb{F}_{2^n}$ and let u_i be $M-2$ auxiliary variables in $\mathbb{F}_{2^n}$.

$$\begin{cases} S_3(u_1, x_1, x_2) \\ S_3(u_1, u_2, x_3) \\ S_3(u_2, u_3, x_4) \\ \vdots \\ S_3(u_i, u_{i+1}, x_{i+2}) \\ \vdots \\ S_3(u_{M-3}, u_{M-2}, x_{M-1}) \\ S_3(u_{M-2}, x_M, R_X). \end{cases}$$

Fig. 2. A Serial-Connection-like Method for Point Splitting Proposed by Semev in 2015

We have $M-1$ equations in $\mathbb{F}_{2^n}$ where M is the number of points in our decomposition of R as a sum M elliptic curve points. We can call it a Semaev-serial system of equations as it effectively is **a serial connection**[12] of several systems of equations of type S_3 in a certain encoding (with a topology of straight line with connections only between consecutive components).

5.8 On Topology of Equations

Block ciphers are typically quite hard to break by algebraic attacks, and this sort of **block cipher or serial topology** of [45] is an example of how **not** to approach this problem. An example of alternative approach can be found in [34] which paper proposes to decompose S_M into several S3 equations using **a tree topology**, see Section 5 of [34]. We believe nevertheless that the serial approach in Fig. 2 and the tree one in [34] are better than the initial one from 2004 in [46], which leads to polynomials of very high degree. This point is also made in the abstract of [34]. Another approach which has this unfortunate serial topology is the very recent approach to constraints for certain elliptic curves mod P in [42]. In this paper we will propose an approach which at antipodes w.r.t. a serial connection topology. Our goal is to achieve some sort of **dense topology** where there is a lot interaction between different equations and different monomials. We believe that this is actually the only plausible way to achieve something which is efficiently solvable in finite fields of large size.

[12] This sort of topology for systems of equations is very common in block cipher cryptanalysis [5, 8, 14] and the hardness of solving these systems can be seen as a hardness to derive ANY sort of algebraic or statistical knowledge about the middle variables not easily accessible to the attacker.

Part II

High-Level General Point Splitting Methods

6 High Density Approach For General Point Splitting

In this section we propose our first method and approach which leads to **highly overdefined** systems of equations. Moreover we will also try to construct equations which have some sort of very dense[13] topology. Our construction is completely geeneral: we work for curves mod p and consider splitting in an arbitrary number of components. Later we will at some special cases such as splitting in two.

6.1 Our Main General Construction Ex_M^K

As in [45] we are going to add additional variables which is a frequently used trick in algebraic cryptanalysis. We are going to present a general approach for any M and ordinary elliptic curves mod p. We will generates a certain highly connected and highly overdefined system of equations which later will be called Ex_M^K.

We want to solve:

$$S_{M+1}(P_{1X}, P_{2X}, \ldots, P_{MX}, R_X) = 0$$

where the P_i should lie in a well-chosen subspace.

6.2 Adding Variables Using A Linear ECC Code

We are going to consider K constant points S_i which lie on the same elliptic curve In this paper we are going to assume that these points are random, and that their X coordinates are all distinct numbers mod p. Moreover the first point will be always assumed to be $S_0 = \infty$. In future works we are going to explore how to improve our attacks by using a well-chosen set of points S_i.

We are going to add new variables for the X coordinate of every possible $P_i + S_j$ which addition is done on the curve. Let

$$Z_{ij} = (P_i + S_j)_X$$

We have assumed $S_0 = \infty$ so that the original P_i is also one of the these variables with $\forall i P_i = Z_{i0}$.

New Equations Our new equations are going to be:

$$S_{M+1}(Z_{1i_1}, Z_{2i_2}, \ldots, Z_{Mi_M},\ (R+\sum_{j=1}^{M} S_{i_j})_X) = 0 \quad \forall i_j \in \{0, \ldots, K-1\} \qquad (Ex_M^K)$$

We call these expanded equations Ex_M^K.

[13] Informally, dense topology is the contrary of the serial or block cipher topology in [45, 42] Very dense could be defined as something like: "one equation shares many major non-linear monomials with a number of other equations as large as possible" where in our approach "as large as possible" means a number higher than a constant, for example $\mathcal{O}(K)$.

6.3 Comparison to ECC Codes

In our equations we have a large linear ECC Code with expansion factor of $K + K$ times defined as follows:

$$P \mapsto [\ P_1 + S_0,\ \ P_1 + S_1,\ \ldots\ P_1 + S_{K-1};\quad P_2 + S_0,\ \ P_2 + S_1,\ \ldots\ P_2 + S_{K-1}\ ]$$

and our equations are Semaev polynomials after substitution of one variable by specific constants computed on the EC. So they are not exactly just[14] Semaev equations.

6.4 Analysis of Equations $\boldsymbol{Ex_M^K}$

Here in each equation the last number is a constant which the attacker computes. This decreases the degree of the equations and it is THE place where the law of the elliptic curve interacts with the construction of our equations in a very substantial[15] way.

We have

$$R = K^M$$

equations of degree 2^{M-1} in each variable and one variable is a constant. Here according to Def. 5.6.1 each equation has 2^{M-1}-degree equal to M and total degree up to $M2^{M-1}$.

The total number of monomials which appear in these equations is approximately:

$$T = \mathcal{O}(K^M)$$

More precisely if we ignore products of $M - 1$ or less powers of variables, and only look at the dominating part with 2^{M-1}-degree equal to exactly M, it is about

$$T \approx 2^{M(M-1)} \cdot K^M/M!$$

This is a very largely overdefined system of equations, as $T \gg V$ where V is the number of variables which grows only linearly with K:

$$V = M \cdot K$$

We recall that M is a constant and K is allowed to grow in an arbitrary way.

[14] If the EC law was a black box group and the results of long point additions were essentially random rather than related to each other, not that we believe any such claim, we could consider that our equations are to a large degree random equations with some specific monomials, or at least locally they should look like random polynomials.

[15] This interaction satisfies our informal "dense topology" requirement in a certain different way than in other places and qualifies to be called a "densely connected" method. Different powers of the constants obtained form a complex EC point addition are used and their effect is diffused very substantially due to the fact that each monomial of 2^{M-1}-degree equal to M can be obtained from up to 2^{M-1} different monomials of 2^{M-1}-degree equal to $M + 1$ when one out of M variables is replaced by a constant in a Semaev polynomial S_M.

6.5 On Linear Dependencies

Linear dependencies are a major difficulty in algebraic cryptanalysis and a major topic of study in Gröbner basis theory. They make that many attacks do not work as well as expected, cf. [3, 24, 25, 8, 9, 7, 1] and many other.

At this stage we do NOT have this problem.

Theorem 6.5.1. 100% of equations in set Ex_M^K are **linearly independent** and we have

$$F = R = K^M.$$

Proof: Each of our equations contains a monomial

$$Z_{1i_1}^{2^{M-1}} \cdot Z_{2i_2}^{2^{M-1}} \cdot \ldots \cdot Z_{Mi_M}^{2^{M-1}}$$

which monomial appears exactly once, only in this equation and in no other equation in our set Ex_M^K .

6.6 On R/T and F/T Ratio

In algebraic cryptanalysis of block ciphers R/T ratio is frequently studied. Initially block ciphers cryptanalysts aim at a ratio of type say $R/T = 1/3$, or $1/4$ and as large as possible see slides 64,69-70 in [22] and [24, 25]. Then the expansion step such XL,XSL or other algorithm improves the ratio and makes it closer to 1, cf. slide 73 in [22]. In this paper we simply propose yet another **new** and original method to make this R/T ratio grow and which does not expand the degree of the equations compared to one single polynomial or type S_M. This degree remains constant at all times. Instead we expand the number of variables by adding new variables and new equations.

For now the original R/T ratio is:

$$R/T \approx M!/2^{M(M-1)}$$

And following Thm. 6.5.1:

$$F/T = R/T \approx M!/2^{M(M-1)}$$

This is a constant in any point splitting problem with M parts.

6.7 The Philosophy of Ex_M^K Approach

As in many algebraic attacks approaches we aim at obtaining degree falls and constructing an efficiently solvable overdefined system of algebraic equations. However we expand the equations NOT by increasing the degree but by increasing the number of variables[16] following a linear ECC Code expansion.

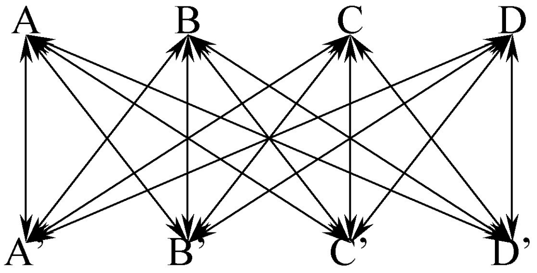

Fig. 3. Graph of connections between points on both sides, $M = 2, K = 4$

This serves our two main strategic goals:

1. We will obtain systems of equations which are **massively overdefined** with a very high R/T ratio which is substantially higher than in previous works on this topic, cf. Section 6.9 below.
2. We conserve a special structure which makes that only some types of monomials are used during the attack, however we try to make sure to obtain a very **densely connected topology**.

In this paper we will argue that the combination of these 2 properties can make systems of equations efficiently solvable even though it is clear that these conditions are NOT sufficient, see later Section and 12.8 and Part VI.

6.8 On [Complete] Graphs and HyperGraphs

When $M = 2$ we can study our equations in terms of graphs. If we connect in a graph each pair of variables for which we wrote an S3 equations with a third variable being a constant, we get a bi-clique graph, cf. Fig. 3. Later in this paper we will also add connections on each side, see Section 7 and later cf. Fig. 6 page 42 and we will obtain truly complete graphs which will somewhat lose their bi-partite structure (or it will be less visible).

In general for any $M \geq 2$ our approach can be studied in terms of [complete] hypergraphs in which hyper-edges can connect up to M vertices which correspond to distinct variables in our system of equations.

[16] This is again following one of so called "Fast" algebraic attacks strategies, cf. slide 84 in [19] and slide 121 in [22] and specifically avoiding expansion of the degree of the equations, cf. slides 82-86 of [19]. This can be seen as the dual approach of the degree expansion approach which has dominated this space since 2000 [6].

6.9 On F/T Ratio and Solvability

We claim that a ratio F/T being a constant makes our equations Ex_M^K potentially solvable by Gröbner basis algorithms. A constant ratio F/T should be compared to a much smaller ratio which is the case in ALL previously published papers on this topic [46, 45, 43, 42] and many other.

In general however it is known that **no amount of work on** R/T or F/T can solve algebraic equations **if they have a large number of solutions**, which is the case for our equations Ex_M^K. For algebraic equations it would be sufficient to guess some variables and decrease the number of solutions, cf. FXL algorithm [6, 7]. In the case of equations mod P this is a major difficulty and no satisfactory[17] solution to this problem was proposed until now[18]. In Part VI we consider the question what kind of properties are desirable for a solution to this problem to work well in the context of ECDL attacks and other similar problems we study, and we propose an new original way to solve this problem.

6.10 Sub-Exponential Attacks On EC Discrete Log Problem?

In spite of these difficulties let use assume that there exists[19] some approach which allows to reduce the number of solutions. Then probably with our equations Ex_M^K we can solve our point splitting problem:

Conjecture 6.10.1. Assume that the point splitting problem is encoded by using the same set of monomials as used in our equations Ex_M^K [this assumption is probably NOT realistic[19]]. Then one should be able to prove in Gröbner basis theory and the combined system of equations can be solved in sub-exponential time by F4 or similar Gröbner basis algorithm [1, 2].

Justification: For any system of equations with $R/T = 1/8$ we can easily improve the R/T ratio by increasing the degree. The basic mechanism is widely known: R grows and T grows but R grows faster because there are several ways to obtain the same monomial, cf. slide 80 of [22]. There are also questions of linear dependencies in expanded equations, cf. [3, 24, 25, 8, 9, 7, 1].

Here we need to design and analyse a custom version of this process adapted to the special structure of our monomials. We are going to multiply each monomial of 2^{M-1}-degree of up to M, by all possible powers T^r for $r = 1..2^{M-1}$ and for any variable T which does NOT appear in this monomial. Thus we avoid ever creating powers higher than 2^{M-1} and we move to equations of 2^{M-1}-degree of $M + 1$.

Most of the time a variable T does not appear in our monomial and we have $V = MK$ variables. The number of expanded equations is now about :

[17] Two major very recent attempts to solve this problem can be found in [42] however they do not have a "dense topology" we are trying to achieve in this paper.

[18] cf. Part VI for our proposed solution.

[19] All known solutions will require one to add yet more extra variables. Until recently the next best thing would be the method(s) of [42] which could maybe be combined with our methodology due to some equations the existence of which we show in Section 16. In late July 2016 we proposed a new original methodology and some concrete solutions to this problem which was added to this paper, cf. Part VI.

$$R^{expanded} \approx 2^{M-1} \cdot M \cdot K^{M+1}$$

and they are still of degree up to 2^{M-1} in each variable. We count the total number of monomials which appear in these equations. All monomials have 2^{M-1}-degree equal to at most $M+1$, se we have approximately:

$$T^{expanded} \approx 2^{(M+1)(M-1)} \cdot K^{M+1}/(M+1)! = \mathcal{O}(K^{M+1})$$

Initially we had $R/T \approx M!/2^{M(M-1)}$. Now we have:

$$\frac{R^{expanded}}{T^{expanded}} \approx \frac{2^{M-1} \cdot M \cdot (M+1)!}{2^{(M+1)(M-1)}} \approx M(M+1) \cdot (R/T)$$

We see that our ratio will improve roughly M^2 times (we neglected terms of lower r-degree). this at the price of increasing the 2^{M-1}-degree by 1. Eventually after a few steps we will achieve $R/T \approx 1$, though by the present method we do not claim that it will become exactly equal to 1.

Polynomial? Subexponential? - future work: We conjecture that a family of systems of equations which achieves $F/T \to 1$ and **have a unique**[20] **solution** is efficiently solvable in the worst case and that the solution can be computed in polynomial time.

This also for a peculiar set of monomials we have here in a similar way as in general, for general dense polynomials of a certain degree [1]. The situation is quite clear for equations over $\mathbb{F}_2$ with the so called T' method which was proposed at Asiacrypt 2002, cf. [22] slides 95-101 and [24].

For larger fields the result is more hazardous. We will leave this question for future research.

Remark: Later in this paper we will show how to achieve $F/T \approx 1$ **directly** without additional degree expansion [needed so far].

[20] Unhappily we don't achieve this here, this question is studied much later in Part VI and more precisely, in the case of the Ex_M^K construction for arbitrary M, we sketch some basic solution in Section 24.1 page 85. The solution will however in turn interfere with the F/T ratio and no final and mature solution with a very good F/T ratio is proposed yet.

7 High Density Approach for $M = 2$

In this section we are going to re-state approach of building a highly overdefined system of equations with dense topology for $M = 2$ in particular. We are also going to add more equations. Previous equations Ex_2^K formed a bi-clique graph, cf. Fig. 3 page 28. Now we will add connections internal to each side and we will obtain a complete graph, cf. Fig. 6 which figure we also reproduce on this page for better readability. The new equations will be called Dx_2^K.

8 Summary of Ex_2^K and New Equations Dx_2^K

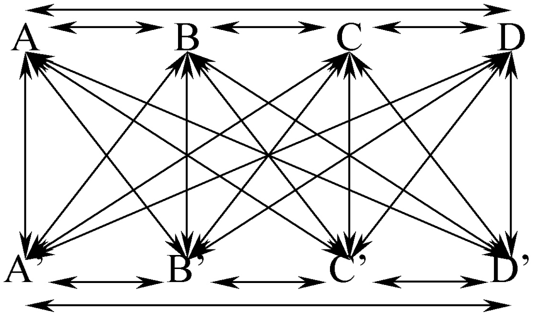

Fig. 4. Graph of connections in Ex_2^K and in new equations Dx_2^K

We are now going to re-state and summarize our how previous equations are created and we will introduce new additional equations. We want to solve:

$$S_3(P_{1X}, P_{2X}, R_X) = 0$$

where the P_i lie in some subspace of size $2^{n/M}$ which we do not yet specify.

Again we consider K constant [random] EC points S_i with distinct X coordinates and $S_0 = \infty$ and let:

$$Z_{ij} = (P_i + S_j)_X$$

We have $V = K + K$ variables. We consider two sets of equations where the first is the same as before for $M = 2$.

$$S_3(Z_{1i}, Z_{2j},\ (R + S_i + S_j)_X) = 0 \quad \forall i, j \in \{0, \ldots, K-1\} \qquad (Ex_2^K)$$

As before we have:

$$R^{Ex} = K^2.$$

Here are the additional equations. For any pair of variables Z_{ki} and Z_{kj} we code the fact that the difference on the EC of suitable points is a known constant:

$$S_3(Z_{ki}, Z_{kj}, (S_i - S_j)_X) = 0 \quad \forall k \in \{1, 2\} \forall i, j \in \{0, \ldots, K-1\} \qquad (Dx_2^K)$$

We have new equations of 2-degree 2. Initially we have

$$R^{Dx} = M\binom{K}{2} \approx K^2$$

such equations. New we count the monomials in these equations. First we count monomials of 2-degree 2 which come from the first set Ex^K_M. We have monomials of type TU, T^2U, TU^2, T^2U^2:

$$T^{D2x2} \approx 4\binom{K}{2} \approx 2K^2$$

Now we have a different disjoint set of 2-degree 2 monomials TT' or T^2T' etc from the second set Dx^K_M.

$$T^{E2x2} = 4 \cdot 2\binom{K}{2} \approx 4K^2$$

The number of affine monomials is $V + 1 = 2K + 1$. Overall the total number of monomials which appear in these equations is approximately:

$$T \approx 6K^2$$

Again we expect that:

Theorem 8.0.2. 100 % of equations in set Ex^K_M are linearly independent and we have

$$F = R = 2K^2.$$

Proof: Each of our Ex/Dx equations contains one monomial $Z^2_{1i} \cdot Z^2_{2j}$ or $Z^2_{1i} \cdot Z^2_{1j}$ or $Z^2_{2i} \cdot Z^2_{2j}$ which monomial appears exactly once, and appears in no other equation in our set $Dx^K_2 \cup Ex^K_2$.

In this system of equation we initially have

$$R/T \approx \frac{2}{6}$$

As in Section 6.10 we conjecture that this probably leads to a subexponential algorithm IF constraints can be coded without creating extra monomials [which again is rather unrealistic and we will study this question later].

9 On The Role of Semaev Polynomials

Until now the Semaev polynomials play a central role in our efforts to cryptanalyse the ECDL problem. We have just proposed a method to solve one single equation which is a Semaev polynomial by generating a vast number of other Semaev polynomials which leads to a highly overdefined system. In what follows wee are going to discover that there exist other **new polynomial equations** which are simpler the Semaev polynomials, have **lower degree** and lower complexity, and [as it will turn out] are NOT contained in the ideal a Gröbner basis generated by our Semaev polynomials used so far. In addition they can be generated in larger numbers with asymptotically interesting properties, so that for example their number grows asymptotically quite fast and some degree of saturation [as in ElimLin] becomes possible.

Until now with Semaev polynomials cf. Section 6.6 we could achieve by a variable change [replacing each power of a variable by a new variable] a system of pure quadratic equations [still based on Semaev polynomials] with $F/T = \mathcal{O}(1)$ which in practice is a fairly small constant. Our objective is a lot more ambitious, we want to construct a system of purely quadratic equations such that $F/T \rightarrow 1$.

Part III

Towards Better / Faster Point Splitting Methods - A Proof Of Concept For $M = 2$

10 Enhanced High Density Approach For $M = 2$

In this section we are going to prove some quite surprising results about our approach. Our current approach with $Dx_2^K \cup Ex_2^K$ can be summarized as follows: we have K variables of type 1 and K variables of type $P2$. Then for any pair either of type 11, 12 or 22 we are able to connect it by an S3 equation of 2-degree 2. We get a sort of complete graph: all connections are present. We are interested in triangles (cycles of length 3) in this graph. Without loss of generality each of them is of the form:

$$(P_1 + S_i)_X,\ (P_1 + S_j)_X,\ (P_2 + S_k)_X,$$

This triangle is in our system of equations coded as 3 equations:

$$\begin{cases} D(ij) & S_3(Z_{1i}, Z_{1j}, (S_i - S_j)_X) = 0 \\ E(ik) & S_3(Z_{1i}, Z_{2k},\ (R + S_i + S_k)_X) = 0 \\ E(jk) & S_3(Z_{1j}, Z_{2k},\ (R + S_j + S_k)_X) = 0 \end{cases}$$

These 3 equations interact in an interesting way. They are characterized by the fact by for any pair either their sum or a difference is known.

Now we have a surprising result:

Theorem 10.0.3. The attacker can essentially in constant time compute 1 additional linearly independent equation which use exactly the same 12 of 2-degree 2 monomials which appear in $D(ij), E(ik), E(jk)$ and no new monomial is needed. which is of total degree 2, i.e. no variable is squared. In fact it has only 3 non-linear monomials $Z_{1i}Z_{1j}, Z_{1i}Z_{2k}, Z_{1j}Z_{1k}$. It has 7 coefficients total which are polynomials in the inputs of the problem which are the fixe sums or differences between the 3 points, one for each side of the triangle in question.

Justification: This fact is a bit surprising because this equation is NOT[21] in the ideal generated by the 3 S3 equations which form the triangle. We will not obtain it from as degree falls in a Gröbner basis computation unless we add additional equations to it.

We thank our students Wei Shao and Huanyu Ma [UCL M.Sc. Information Security 2015-16 and Cryptanalysis GA18] for computing these equations by two different methods.

Definition 10.0.4 (MQ2(x,y,z)).
In what follows are going to call $MQ2(Z_{1i}, Z_{1j}, Z_{2k})$ this particular equation which is of total degree 2.

In Section 14.1 we show a realistic example of such an equation and in Section 11 we provide the general result and provide closed formulas for computing this MQ2 equation.

[21] We have verified this independently with three different software tools: SAGE, Maple and proprietary software.

11 MQ2 - A New Method of Generating Quadratic Equations

Now we are going to explain in which exact cases the equations of type MQ2 exist and can be generated.

Let $P \neq \infty$ be a point on an elliptic curve modulo a large prime p and let S_0 and $S_1 \neq S_0$, and $S_2 \neq S_0$ be three distinct constants, i.e. $S_1 \neq S_2$.

We consider a linear ECC Code defined as follows:

$$P \mapsto [\ P + S_0, \quad P + S_1, \quad P + S_2]$$

where additions are done on the curve. We call $ix1, ix2, ix3$ the x coordinates of these three points and we expand our constant to a [redundant] set of differences on the EC for which we also consider only x coordinates and ignore the y coordinates, which is another form of ECC Code which we use here because it symmetric greatly simplifies our equations and our result. By definition we have:

$$\begin{aligned}
(ix1, .) &= P + S_0 \\
(ix2, .) &= P + S_1 \\
(ix3, .) &= P + S_2 \\
(dx12, .) &= S_1 - S_0 \\
(dx13, .) &= S_2 - S_0 \\
(dx23, .) &= S_2 - S_1
\end{aligned}$$

Fig. 5. Notation for MQ2 and MQ3 Equations

We call each such configuration of three points with known differences dxij a **triangle** which involves three main variables $(P + S_i)_x$ which are the x coordinates for our EC codewords as defined above where P is a free variable which à priori not known.

Theorem 11.0.5. For every triangle with the assumptions above, there exist a polynomial in 6 variables ixj and $dxij$ and with only 3 non-linear [quadratic] monomials in the ixj which is true each time except when one point is degenerate, i.e. true always if all the three variables satisfy $P + S_i \neq \infty$.

For the bitcoin elliptic curve and any elliptic curve mod P where $A = 0$ and $B = 7$ this equation is exactly as follows.

```
1*(84*dx12*dx13+84*dx12*dx23+84*dx13*dx23)  +
ix1*(56*dx12+56*dx13+84*dx23+dx12^2*dx13^2-dx12^2*dx23^2-dx13^2*dx23^2+2*dx12*dx13*dx23^2)  +
ix2*(56*dx12+84*dx13+56*dx23-dx12^2*dx13^2+dx12^2*dx23^2-dx13^2*dx23^2+2*dx12*dx13^2*dx23)  +
ix3*(56*dx12+28*dx13+28*dx23+2*dx13^2*dx23^2-2*dx12*dx13*dx23^2-2*dx12*dx13^2*dx23)  +
ix1*ix2*(28-2*dx12*dx13^2-2*dx12*dx23^2+2*dx13*dx23^2+2*dx13^2*dx23+4*dx12*dx13*dx23)  +
ix1*ix3*(28+2*dx12*dx23^2-2*dx13*dx23^2-2*dx12^2*dx13+2*dx12^2*dx23+4*dx12*dx13*dx23)  +
ix2*ix3*(28+2*dx12*dx13^2+2*dx12^2*dx13-2*dx12^2*dx23-2*dx13^2*dx23+4*dx12*dx13*dx23) = 0
```

Furthermore, we have a completeness property: this equation is unique modulo the dependencies between that $dxij$ and it is entirely unique after substitution by fixed constants of $dxij$. Across all possible code words in the ECC Code defined above there is **exactly** one linearly independent equation involving 7 monomials $1, ixi, ixi * ixj$.

In general there exists one single unique equation of this form and for any other elliptic curve mod P in Weierstrass form, we just have a few more terms which depend on A, here $A = 0$ and the equation is simplified.

11.1 Beyond MQ2 - Related Results

We also have a similar equation with an 8-th monomial in the xi, which is the product of all the 3 variables. This second equation is called MQ3 and the corresponding formulas are given in below, cf. Thm. 11.2.1.

In addition in Thm. 15.0.1 and Thm. 15.1.1 Section 15 page 55 we generalize these basic equations to exhibit very similar unique equations of arbitrary degree d.

Similar results also exist if we square one variable, cf. for example 16.0.2 which equations are also unique equations with certain fixed set of monomials, and we refer to Section 14.3 and to Section 14.5 for specific examples of how such more general equations can be generated by some other explicit closed formulas for another specific ECC Code which is essentially the same except that we allow more monomials which can be seen as extending the EC Code by another polynomial-type code built on the top of it.

11.2 MQ3 - A Major Variant of MQ2

We also have:

Theorem 11.2.1. For every triangle with the assumptions above, there exist a second polynomial in 6 variables ixj and $dxij$ and with only 4 non-linear [3 quadratic and cubic] monomials in the ixj are used, again true all the time except when one point is degenerate, i.e. $\forall_i(P + S_i)_x \neq 0$. Again when $A = 0$ this equation is exactly:

```
-4B*(dx12*dx13+dx12*dx23+dx13*dx23)  +
-4B*(ix1+ix2+ix3)*(dx12+dx13+dx23)  +
ix1*ix2*(dx12*dx13^2+dx12*dx23^2-dx13*dx23^2-dx13^2*dx23)  +
ix1*ix3*(-dx12*dx23^2+dx13*dx23^2+dx12^2*dx13-dx12^2*dx23)  +
ix2*ix3*(-dx12*dx13^2-dx12^2*dx13+dx12^2*dx23+dx13^2*dx23)  +
ix1*ix2*ix3*(-dx12*dx13-dx12*dx23-dx13*dx23) = 0
```

Furthermore, we have this equation is also unique after substitution of constants of $dxij$, and across all possible code words in the ECC Code defined above there is **exactly one** linearly independent equation involving our 8 monomials.

This equation can be used to compute Eliminators: equations which allow to replace terms of degree 3 with lower degree terms, E is the number of Eliminators, cf. Section 13.9.

11.3 On Linear Dependencies, MQ2 vs. MQ3

It is easy to see that, unlike MQ2 equations most of which are typically linearly dependent in applications which use them, which we will see later in Thm. 12.3.1 and many subsequent results, the MQ3 equations are strictly linearly independent. We have the following result:

Theorem 11.3.1. Consider an arbitrary set of K points P_i on an elliptic curve mod P such that for every pair of points we know either their sum or their difference for the EC addition law (by convention we call such set an **ECC Code**[22] **expansion** of one point). Consider the x coordinates of the points and ignore the y coordinates. We call $D3^K$ the set of equations obtained from writing the equation MQ3 given by the formulas of Section 11.2 above for each triple of points.

We have exactly $\binom{K}{3}$ equations $D3^K$ and 100 % of equations in set $D3^K$ are linearly independent.

$$F = R = 2K^2.$$

Proof: From definition in Section 11.2 it is obvious that each cubic monomial in our $D3^K$ equations appears in exactly one equation, the equation written for that exact triple of variables.

12 Consequences of Using MQ2/MQ3 Equations

It is possible to see that with MQ2 equations we are going to obtain things which were not quite possible to have with standard Semaev polynomials we have used so far.

12.1 What's Special About the New Equations

We claim that our new equations MQ2 are **a lot more interesting** than any other equations we have seen so far, and anything which is contained in the ideal generated by the S3 polynomials. The key remark is that this new equation MQ2 is such that it depends simultaneously on all 3 sides of the triangle. This has very important consequences, basically the number of such equations which can be generated is overall going to be very substantially larger than just multiplying the number of Semaev equations by a constant.

In order to see that, we consider again our previous setup from Section 7 where we now added a lot new and simpler equations cf. Section 10.

12.2 How To Generate a Lot of Equations

The main difference between our MQ2 equation and the 3 initial S3 equations from which we started is that so far the number of (old) equations would grow quadratically with K. The initial 2-degree quadratic equations S3 exist for every pair i, j in the full system of equations $Dx_2^K \cup Ex_2^K$ with $K + K$ variables. At the same time the number of variables also grows quadratically.

Until now the only thing we could hope for and which we have achieved was something like $R/T \approx \frac{2}{6}$. This is already better than many previous approaches in the literature. However this not very satisfactory.

[22] As always in error-correcting codes, it is a linear code of a certain type where the linearity holds over the elliptic curve, cf. Section 3.8 for a definition we use which is slightly different than in the error correcting codes literature.

Unhappily current Gröbner basis theory does not provide any real guarantee that such systems of equations could be efficiently solvable. This remains a bold conjecture and the answer could be that it depends on the constant or the complexity could be a very fast growing sub-exponential implied by the progressive increase in R/T which cf. Section 6.10, which algorithm would be far from being practical to execute.

Things look **a lot better** with our set of pure unique quadratic equations $MQ2(Z_{1i}, Z_{1j}, Z_{2k})$.

12.3 On Combinatorial Explosion Due to MQ2 Equations

We continue the study the new MQ2 equations which can be written for the initial system of equations $Dx_2^K \cup Ex_2^K$ with $K+K$ variables.

Theorem 12.3.1. Our pure quadratic equations $MQ2(t,u,v)$ exist and are obtained in the same way which we detail in Section 11 **for any**[23] **triple of variables** t,u,v out of $K+K$ regardless on which side[23] they are. We call DEx_2^K the set of these equations for all possible $K+K$ variables $t,u,v \in \{Z_{1i},\ldots,Z_{2i}\}$.

The number of MQ2 equations in DEx_2^K grows as $\mathcal{O}(K^3)$ which is **a lot faster** than the number of monomials which remains $\mathcal{O}(K^2)$.

Proof:

We have:

$$R^{DEx} = \binom{2K}{3} \approx \frac{4}{3}K^3$$

The set of monomials is smaller than ever before, as we do not have any squares anymore for non-linear terms:

$$T^{DEx} \approx \binom{2K}{2} + 2K \approx 2K^2$$

Here we can easily achieve $R/T > 1$. We have **a super highly overdefined** system of equations.

[23] We can make them for any triangle on our graph in which each pair of variables out of $2K$ are connected by S3 polynomials because either their sum or their difference is known, cf. Fig. 6. In each case the quadratic equation MQ2 can be computed by the same formula cf. Section 11. This formula takes as input the sums or differences between each pair and requires only x coordinates and is indifferent whether these are sums or differences. It is also invariant if we replace the 3 points by their opposites on the curve. So that we can apply the same formula and get for example equations of type $MQ2(Z_{2i}, Z_{2j}, Z_{2k})$.

12.4 On Dense Topology and High Graph Connectivity in DEx_2^K Equations

There is no doubt either that the topology of this system of equations is **excessively dense**: each equation shares each non-linear monomial it has with $\mathcal{O}(K)$ other equations which connections go into and spread over about half of the whole set of equations. Our set of variables forms a complete graph with all pairs are connected and which contains a very large number of graph cliques of different sizes (complete subgraphs).

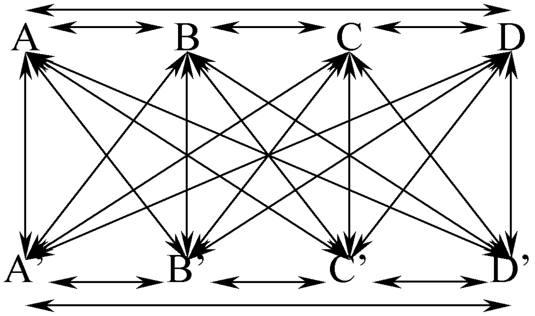

Fig. 6. Graph of connections between points on both sides, $K = 4$

Definition 12.4.1 (Triangles).
We recall that in this paper we will call "triangles" complete sub-graphs with 3 points in our graph which are also sets of 3 variables, cf. Section 11.

12.5 Linear Dependencies for MQ2 Equations in DEx_2^K

Let F is the number of linearly independent equations inside our $R^{DEx} = \mathcal{O}(K^3)$ equations, it is easy to see that a fast cubic growth of R cannot translate into a curve which grows as quickly as K^3. First, the equations we generate are somewhat linearly dependent. For example it is possible to see that if we consider a complete graph (a clique) with 4 points, we will obtain 4 MQ2 equations out of which only 3 will be linearly independent. This fact alone shows that as we generate our equations, some could be omitted or they will be redundant.

In addition there is also an upper limit. It is easy to see that at no moment we can have $F > T^{DEx} = \mathcal{O}(K^2)$. It is simply impossible to have more non-redundant equations than monomials. It is important to note however that our goal is not to obtain $F/T > 1$ which is impossible. We simply want to obtain $F/T \approx 1$, cf. slide 75 in [22]. Overall we want to monitor the following quantity:

$$\frac{2K^2 - F}{2K^2}$$

and see if this quantity decreases and approaches 0, which will mean that $\lim_{K\to\infty} F/T < 1$ or if it grows slower than quadratic, this will mean that we have $\lim_{K\to\infty} F/T = 1$.

We have done computer simulations for the bitcoin elliptic curve secp256k1 and asked Microsoft Excel for the best polynomial approximation (one which would maximize the squared error). This method gives gives a remarkably good accuracy and the approximation turns out to have integer coefficients:

$$F \approx 2K^2 - 3K + 1$$

Overall we have established that:

Theorem 12.5.1. The number F of linearly independent quadratic equations and the number of terms in DEx_2^K satisfy the following approximations:

$$F \approx 2K^2 - 3K \qquad T \approx 2K^2 + K \qquad \lim_{K\to\infty} F/T = 1.$$

At this moment we can only provide experimental evidence for this result, cf. Table 7 on page below. 45.

12.6 Comparison to ElimLin

We do NOT achieve an exact equivalent of what we previously achieved with ElimLin algorithm. In Fig. 1 page 17 we have really obtained a curve which grows as K^2 This even though in the long run the quantity must become linear, because we cannot obtain more than $T = \mathcal{O}(K)$. In Section 4.3 ElimLin has 3 distinct stages: no non-trivial equations initially or $\mathcal{O}(K^0)$ and overall the number of equations found grows linearly $R = \mathcal{O}(K^1)$ due to some trivial equations. Then for a long time equations are generated at a quadratic rate $\mathcal{O}(K^2)$ which is undoubtedly faster than linear, and ultimately there are dependencies and ElimLin reaches a phase transition. It seems that the growth inevitably tends to re-become linear for large K, but this is not quite correct. The phase transition is quite fast and we only have few discrete points from which it is hard to measure the growth rate in a sensible way.

With our DEx_2^K equations the situation is a bit different, possibly for fundamental reasons outlined in Section 6.9 and Section 12.8 below. A direct comparison would be therefore misleading. Here, our F does NOT grow faster than K^2, not even temporarily, or at least we have not observed it. However, overall we achieve our objective of “saturation” all the same. Our goal in both cases is just to make F approach T asymptotically, cf. slide 75 in [22]. In both cases we achieve our primary goal, with

$$\lim_{K\to\infty} F/T = 1.$$

12.7 Next Steps?

We are able to create a system of equations which encodes the problem of EC point splitting in two. It is massively overdefined and close to saturation. The density/connectivity here is in fact clearly[24] better than in ElimLin and therefore probably better than required. However we still have a serious problem.

12.8 Limitations Due to Large Number of Solutions

Is our system of equations DEx_2^K going necessarily tend to complete saturation[25] and become solvable in polynomial time? The answer is **no**, this would be too good to be true. Unhappily, as before in Section 6.10 we cannot hope to achieve complete saturation as long our system of equations has a large number of solutions. Moreover strictly speaking the answer is yes, if we don't specify any constants all systems of equations we study are easy to solve.

Now if we code additional constraints by some extra polynomial equations, the answer will be different. We need to work on encoding constraints into such a system of equations which again is an open problem which arguably has not yet been solved in a satisfactory[26] way.

[24] This is again because each equation shares each non-linear monomial with a large number of $\mathcal{O}(K)$ of other equations.

[25] Arguably we are very close to saturation, possibly as close as one could be, given the fact that our system has plenty of solutions, we cannot really expect to do much better, cf. Section 6.9.

[26] Until now extremely very few researchers have tried to do it for mod P curves, with two notable exceptions of early work on this topic [46] and again recently in [42]. In this paper in Section 16 we study some polynomial equations which could potentially be combined with the latter method. Finally we will propose a brand new original approach to this problem in Part VI.

13 Computer Simulations

In this section we do NOT yet address the problem of constraints. We test our "$M = 2$ + ECC Code" approach and describe additional expanded variants of it.

13.1 Computer Simulations At Degree 2

We have done a series of simulations with our equations.

The command line to run these simulations with our software under Windows 10 x64 command line prompt and for the 256-bit bitcoin elliptic curve is:

```
ec2decomp.exe 93931 256 K 1
```

This software which runs under windows command line can be found at [10]. There are two executables ec2decomp.exe and ax64.exe which need to be in the same directory, direct links are:
`http://www.nicolascourtois.com/software/ec2decomp.exe` and
`http://www.nicolascourtois.com/software/ax64.exe`.

We note that the results we report in computer simulations depend on the elliptic curve **slightly** for small curves sometimes we sometimes get different results. For large curves we have never observed any difference: the results seem not to depend at all on the curve choice[27].

DEx_2^K quadratic					
$2K$ value	4	8	16	32	64
$R = \#$eqs	4	48	448	3840	31744
$T = \#$mons-1	10	36	136	528	2080
F	3	21	105	465	1953
F/T	0.30	0.58	0.77	0.88	0.94
$T - F$	7	15	31	63	127

Fig. 7. Simulations on DEx_2^K equations at degree 2 for the curve secp256k1

Notation. Here E is the number of Eliminators, cf. later Section 13.9.

Remark: We have

$$F = 2K^2 - 3K + 1$$

cf. Section 13.8. Moreover the number of degree 2 monomials is exactly:

$$T = 2K^2 + K$$

[27] Yes MQ2 and every single equation and property we study in this paper also works for P-256, THE curve which almost everybody on this planet uses in their TLS or SSH communications, financial transactions, etc.

13.2 Faster Generation of Equations At Degree 2

An interesting question is that we have generated equations DEx_2^K in such a way that it is redundant system of equations: so far we have written $\mathcal{O}(K^3)$ equations which however have only $\mathcal{O}(K^2)$ monomials, and the rank is only at most $F \in \mathcal{O}(K^2)$ or more precisely $F = 2K^2 - 3K + 1$. An interesting question if there is an obvious way to write ONLY $\mathcal{O}(K^2)$ equations and still achieve the full rank of $F = 2K^2 - 3K + 1$. For example for $K = 2$ we could just ignore one equation out of 4 [selective erasure]. We have developed such a simple [heuristic] way to achieve the maximum rank, and it can be run as follows. It works in 100 % of cases we tested.

```
ec2decomp.exe 93931 256 K 1  /writeonlyMQLimitTriangles2
```

13.3 Completeness At Degree 2

An interesting question is whether MORE quadratic equations exist that those generated (implicit equations or relations, as apposed to the explicitly generated above by closed formulas cf. Section 11). The answer is no, we have already achieved the maximum possible rank.

implicit quadratic any set of $2K$ vars						
$2K$ value	4	8	16	32	64	128
$T =$ #mons-1	10	36	136	528	2080	8256
F	3	21	105	465	1953	
F/T	0.30	0.58	0.77	0.88	0.94	
$T - F$	7	15	31	63	127	

Fig. 8. Simulations on DEx_2^K POTENTIAL equations at degree 2.

13.4 Equations With Additional Cubic Monomials

It is possible to see that for every set of 3 variables there exist exactly one equation as in Section 11 which however also involves a product of 3 variables. We call this equation MQ3, cf. Thm. 11.2.1 in Section 11.2.

Adding this equation makes 2 equations per triangle, and in order to obtain this very basic cubic version we run:

```
ec2decomp.exe 93931 256 K 1 /sub23cubic
```

$DE^{2.3}x_2^K$ cubic			
$2K$ value	4	8	16
$R =$ #eqs	8	112	1120
$T =$ #mons-1	14	92	696
F	7	77	665
F/T	0.5	0.84	0.96
$T - F$	7	15	31

Fig. 9. Simulations on $DE^{2.3}x_2^K$ basic cubic equations for the curve secp256k1

13.5 Completeness At Degree 2.3

In our equations we allow only monomials which are products of up to three DISTINCT variables. In turns out that our equations are complete: no more equations exist with these monomials.

implicit cubic any set of $2K$			
$2K$ value	4	8	16
$T = \#$mons-1	14	92	696
F	7	77	665

Fig. 10. Simulations on $DE^3x_2^K$ POTENTIAL equations at degree 3.

13.6 Remark - No Squares

So far our equations do NOT contain squares nor multiples of squares, yet we could easily enhance our DEx_2^K equations to contain also squares and cubic terms which contain squares.

One method to this is for example by using explicit formulas of Section 14.3 or those in Section 14.5 in a different particular case [less general]. Such equations can also be generated by expansion of Semaev $S3$ polynomials after substitution. We have tested both methods and here we have more monomials and we do NO longer have new equations which are not consequences of the S3.

13.7 Equations At Full Degree Degree 3

This leads to a richer cubic version, or our second cubic version, in which more monomials are allowed, namely all monomials of type xi^2xj and squares of type xi^2xj. We will still not include monomials of type xi^3. In order to obtain this so called cubic version we run:

```
ec2decomp.exe 93931 256 K 1 /cubic
```

$DE^3x_2^K$ cubic				
$2K$ value	4	8	16	32
$R = \#$eqs	16	192	2240	
$T = \#$mons-1	34	156	952	
F	15	125	889	
F/T	0.50	0.80	0.93	0.98
$T - F$	15	31	63	127
E	4	56	560	4960

Fig. 11. Simulations on $DE^3x_2^K$ cubic equations for the curve secp256k1

13.8 Predicting the Outcomes

It is easy to see that for quadratic equations DEx_2^K the outcomes of all our experiments satisfy the following formula for every K:

$$T - F = 4K - 1$$

More precisely we have:

$$F = 2K^2 - 3K + 1 \quad AND \quad T = 2K^2 + K$$

Similarly for our equations $DE^{2.3}x_2^K$ with products of 3 distinct variables allowed we have:

$$T - F = 4K - 1$$

Then for our equations $DE^3x_2^K$ expanded at degree 3 we have:

$$T - F = 8K - 1$$

These formulas give exact results 100 % of the time, no exceptions are known and no exceptions are expected. It is possible to explain these formulas by a careful analysis of linear dependencies which are quite complex here.

We have obtained at last, for all the three major forms of equations:

$$\lim_{K\to\infty} F/T = 1$$

13.9 Emerging Order and Eliminators

Definition 13.9.1 (Deficit).
We call "deficit" the $T-F$ value, in one way all monomials "live" in a linear space of dimension $T-F$.

For example we look at the cubic variant, cf. Fig. 11. These figures show an interesting phenomenon: for $K \geq 4$ we have $T-F$ which is bigger than the number of monomials of degree 0, 1 and 2 combined. We have $T-F = 31 < 28+8+1$.

This means that we can hope that ALL cubic monomials could in theory be eliminated.

In order to study this question we need the following definitions:

Definition 13.9.2 (Degree Falls).
We call degree falls equations in a linear span which have lower degree that the maximum degree in our system of equations.

Definition 13.9.3 (Eliminators).
We call eliminators a subset of our equations which contain only one monomial of maximum degree [e.g. one unique cubic monomial], and all the other monomials are of lower degree.

We sometimes denote their number by letter E.

In our DEx_2^K equations we can write formulas to eliminate cubic monomials DIRECTLY:

Theorem 13.9.4. In $DE^3x_2^K$ we can generate eliminators which allow one to **eliminate by linear algebra ALL monomials of degree 3** out of those present and replace them by equations of degree up to 2.

Proof: For every triple of variables in Thm. 11.2.1 of Section 11.2 we provide direct formula for doing this which allow after substitution of sums/differences of points to obtain Eliminators as defined above.

These basic facts are meant to convince us that we generate systems of equations which are in a state **close to saturation, with abundant degree falls**, and therefore they are [possibly under additional technical conditions] very likely to be **efficiently solvable** by traditional techniques such as Gröbner bases.

Part IV

Higher Degree Generalizations and Other Extensions of MQ2/MQ3

14 On Generalizations of MQ2/MQ3

In this Part IV we work on a number of generalizations of our key result MQ2/MQ3, with higher powers, multivariate monomials of arbitrary degree, squares, specific choices of constants etc. These are additional results which can be omitted when reading this paper for the first time.

14.1 An Example of MQ2 Equation

We give here an example for the unique pure quadratic equation $MQ2(x1, x2, x3)$. This example was computed for the bitcoin elliptic curve secp256k1 and can be seen a computing a unique equation true with probability 1 each time 3 difference assumptions on the 3 variables are simultaneously true. It is meant to serve as a test vector for researchers who want to work on independent software implementations of our attacks.

```
(ix1,.)-(ix2,.)=(15171433492774190503748549945583436057844756659648149165133757042794824375060,
67640625960982344923913992057504513842322505602249806174879684644326606374127)
(ix1,.)+(ix3,.)=(81149267300640140886580502893228406835357736153786577582237861639949325504941,
7509570462874618777196481012071885381809826489931044740507479888910720480630)
(ix2,.)+(ix3,.)=(14246718372890754969447963568415333499450994232060143888351393993594149798440,
113966402045814283029837251687944051865335300380541156011318025391811599328184)

43506603594509476394840594689173534063402467003850248705173010684475860707425 +
-15850657958019531372385726206884354087638153848196298602398101011169790554998*(ix1) +
-39091418262156787157641480333648577894406905078535237269749418900009257959101*(ix2) +
-5252676441644254153640173010066443192133237233781349109351470098348832192777*(ix3) +
-22026423521232397769228695946140590336837141304498724299820260360851509468940*(ix1*ix2) +
-44538550602941308462669843272755909664628310323842486490667946261191910496060*(ix1*ix3) +
 1*(ix2*ix3) = 0
```

We also note that there is no terms of type v^2 in this equation even though such terms are quadratic. It has only 7 monomials.

14.2 Closed Formulas For MQ2 and MQ3 Equations

These formulas appear in Section 11 and Section 11.2.

14.3 MQ3^2 Equations with Squares

Similar to MQ2 and MQ3 equations, we also have:

```
A=0, B=7 =>
-308*(dx12*dx13+dx12*dx23+dx13*dx23)  +
(ix1+ix2+ix3)*(-224*dx12-224*dx13-224*dx23)  +
ix1^2*(3*dx12*dx13^2+3*dx12^2*dx13-dx12^2*dx23-dx13^2*dx23+2*dx12*dx13*dx23)  +
ix2^2*(3*dx12*dx23^2-dx13*dx23^2-dx12^2*dx13+3*dx12^2*dx23+2*dx12*dx13*dx23)  +
ix3^2*(-dx12*dx13^2-dx12*dx23^2+3*dx13*dx23^2+3*dx13^2*dx23+2*dx12*dx13*dx23)  +
ix1*ix2*(10*dx12*dx13^2+10*dx12*dx23^2-10*dx13*dx23^2-2*dx12^2*dx13
        -2*dx12^2*dx23-10*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix1*ix3*(-2*dx12*dx13^2-10*dx12*dx23^2+10*dx13*dx23^2+10*dx12^2*dx13
        -10*dx12^2*dx23-2*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix2*ix3*(-10*dx12*dx13^2-2*dx12*dx23^2-2*dx13*dx23^2-10*dx12^2*dx13
        +10*dx12^2*dx23+10*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix1*(ix2^2+ix3^2)*(-2*dx23^2+dx12*dx13-dx12*dx23-dx13*dx23)  +
ix2*(ix3^2+ix1^2)*(-2*dx13^2-dx12*dx13+dx12*dx23-dx13*dx23)  +
ix3*(ix1^2+ix2^2)*(-2*dx12^2-dx12*dx13-dx12*dx23+dx13*dx23)  = 0
```

Similar equations exist for general elliptic curves. We found that there are several such equations, some with higher degree/complexity in the dx, and which are possibly redundant. We show one which is unusually simple and elegant.

One special case of these equations can be found in Section 14.5. If we substitute the $dx1$, we obtain typically 7 linearly independent equations which number allows one to find special cases of equations which eliminate many monomials of high degree. For example out of 7 we can generate MQ2, one MQ3, and three will be examples for Thm. 16.0.4. All these cases uses substantially less cubic monomials [and MQ2 uses none].

14.4 Open Problems

An interesting question is how an attacker [in some complex attack] can select some well chosen constants in equations of Section 11, or other numerous classes of equations defined in this paper, e.g. D73 equations studied later, so that the equations can be simpler and can be easier to solve.

Another questions is whether for some well-chosen constraints MORE equations could exist which are NOT MQ2 nor MQ3 nor in the ideal generated by such equations. This is what happens in Section 14.5 below.

14.5 More Specific Equations With Squares - A Simple Case With 3 Points

Here is a special case of a triangle, where more than the usual equations MQ2 and MQ3 exist, and there exist an additional third equation with squares which is somewhat unusually simple and contains very few monomials. It is also an example that specific very interesting things can happen if we use well chosen constants in MQ2 equations defined in Section 11.

$$(P1) \mapsto \begin{array}{c} P1 - D \\ P1 \\ P1 + D \end{array}$$

Fig. 12. One Possible Regular Expansion of 1 Variable

We get the following remarkably simple equation:

```
+4B
+dx1^2*(2*ix2-ix1-ix3)
+ix2^2*(2*dx1-ix1-ix3)
+2*ix2*dx1*(ix1+ix3) = 0
```

Remark 1. Here there is a symmetry which makes that ix1 and ix3 always appear together with same coefficients and cannot be separated. There is also a symmetry between $dx1$ and $ix2$.

Remark 2. If we substitute the $dx1$, the existence of such an equation is due to Cor. 16.0.4.

15 Higher Degree Multivariate Eliminators and Generalizations of MQ2 and MQ3

We consider equations which contain only monomials of degrees $0, 1$ and d for every $d + 1$ points. We call these equations **A-d-H Equations** (mix of Affine and degree d Homogenous terms). They are a generalization of MQ2.

Theorem 15.0.1 (A-d-H Equation). For every d and for every $K \geq d + 1$ we consider the usual basic linear ECC Code expansion as follows:

$$P \mapsto [\ P + S_0, \quad P + S_1, \quad P + S_2, \ldots \quad P + S_{K-1}]$$

We assume that the s_i are K distinct fixed points on the ECC. There exist one **unique** equation [unique if constants are fixed] which involves monomials of degrees 0,1, and d and is true across all the points which belong to our code defined above provided that we have $\forall_i P + S_i \neq \infty$.

We give here one example of such equation we generated with $d = 4$.

```
-2911649*(1)+985083*(ix1)-63765*(ix2)-3621022*(ix3)-2932833*(ix4)-3392556*(ix5)
+2091767*(ix1*ix2*ix3*ix4)+1590098*(ix1*ix2*ix3*ix5)+303909*(ix1*ix2*ix4*ix5)
-2441608*(ix1*ix3*ix4*ix5)+1*(ix2*ix3*ix4*ix5) = 0
```

15.1 A Major Variant with 1 More Monomial - MQ3 Equivalent

Similarly, we also have another set of unique equations which contain monomials of degrees $0, 1$ and d and $d+1$, which generalize MQ3 equations for every $d+1$ points.

Theorem 15.1.1 (A-d+1-H Equation). For every d, for every $K \geq d+1$ and for the same ECC Code as in Thm. 15.0.1 above we have one **unique** equation [if constants are fixed] which involves monomials of degrees 0,1, d and $d+1$ true across all the codewords [again if we avoid singularities for all the K points].

Remark: Eliminators: These equations can be used to ELIMINATE any monomial of ANY degree with an arbitrary number of variables and replace it by relatively small number [up to $d+1$] of lower degree and affine monomials.

16 Equations with High Powers in One Variable

This section is meant to demonstrate the power of our technique which consists of adding new variables instead of expanding the degree of equations. The primary goal below is to show that ECC Codes do a job in some remote sense polynomial-algebraically equivalent to increasing the degree of polynomials with one[28] variable motivated by the idea that we need such high powers to encode constraints, specifically by methods such as recently proposed in [42].

For simplicity let $x1 = Z_{10}$ and $x2 = Z_{20}$ and $x3$ by ANY other variable different than $x1, x2$. We have:

Theorem 16.0.2. For every N if $2K \geq 2N-1$ there exists an equation of type:

$$x1^N \text{AffineSum}(2K-1\text{variables excluding } \ x1) = \sum(DEx_2^K \ \text{ monomials})$$

Where $\sum(DEx_2^K monomials)$ is a quadratic equation which belongs to DEx_2^K. discovered by theory, confirmed by simulations

This equations becomes quadratic if we consider that $x1^N$ is a new variable.

Moreover, with 2 more variables, it is possible to avoid x1 to appear inside the quadratic part. The variable x1 will then appear only once.

For example:

Corollary 16.0.3. If $2K \geq 7$ there exists an equation of type:

$$x1^4(1 + ax2 + bx3 + cx4 + dx5 + dx6) = \sum(DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 9$ variables, it is possible to avoid x1 inside the quadratic part.

Corollary 16.0.4. If $2K \geq 3$ there exists an equation of type:

$$x1^2(1 + ax2 + bx3) = \sum(DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 5$ variables, it is possible to avoid x1 inside the quadratic part.

16.1 Eliminating $x1^1 = x1$

The simplest case is as follows:

Corollary 16.1.1. If $2K \geq 1$ there exists an equation of type:

$$x1(1 + ax2 + bx3) = \sum(DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 3$ variables, it is possible to avoid x1 inside the quadratic part and we get

$$x1(1 + ax2 + bx3) = cx2x3 + dx2 + ex3 + f$$

The last equation is simply just what we called MQ2 elsewhere and the explicit formulas to write this equation are given in Section 11.

[28] The reader should also read Section 3.6 which shows the same thing for multivariate polynomials.

Part V

ECC Coding Methods With Three Points At Degree 3

17 Objectives for Part V and D73/D93 Equations

Our goal is to re-code specific forms of ECC Codes as systems of polynomial equations which should be as simple as possible. We start by observing that a certain remarkably simple polynomial equation exists.

17.1 D73 - A New Family of Cubic Polynomial Equations

The following result has been designed as a plausible replacement for arbitrary S3 equations in configurations with redundant "expanded" variables. We have:

Theorem 17.1.1 (D73 Theorem). We consider the following set of variables on EC, a special form of ECC Code with 3 inputs and 7 outputs for any Weierstrass elliptic curve modulo a large P.

$$(P1, P2, P3) \mapsto \begin{array}{ccc} P1 & P2 & P1+P2 \\ P1+P3 & P2+P3 & P1+P2+P3 \\ P3 & & \end{array}$$

Again we look only at x coordinates of the 7 points. We call $sx1 - sx123$ the x coordinates of the 7 points, with $sx1, sx2, sx12$ being the points the first line, with $sx13, sx23, sx123$ being the points the first line, and $sx3$ being the x coordinate for the last point $P3$. This is summarized on the picture below:

$$\begin{array}{ccc} sx1 & sx2 & sx12 \\ sx13 & sx23 & sx123 \\ sx3 & & \end{array}$$

If all the 7 points are distinct from the ECC neutral element ∞ we have:

```
sx1*sx2*(sx23-sx13) +sx1*sx3*(sx12-sx23) +sx2*sx3*(sx13-sx12)
+sx123[sx1*(sx13-sx12)+sx2*(sx12-sx23)+sx3*(sx23-sx13)] = 0
```

Remark. Our D73 equation is a homogenous polynomial of degree 3. It has extremely few terms. Yet another remarkable polynomial relation which we have discovered. Unlike all our previous equations such as MQ2/MQ3 and their generalizations it does NOT depend on the EC coefficients and works the same for all sorts of curves including NIST curves such as P-384. We challenge the reader to discover anything comparable in terms of elegance and simplicity for an ECC Code expansion with a similar expansion factor [29] and with 3 free variables.

[29] This equation is unusually simple. A comparison to MQ2 would probably be not fair, as MQ2 has fewer active EC points (only 1 degree of freedom, here we have 3 degrees of freedom!). In spite of this, the new equation D73 is actually still overall shorter and simpler than MQ2 before substitution, i.e. when we look at how MQ2 depends on the dx variables. Overall the MQ2 formulas are quite complex and MQ2 has polynomials of degree up to 5 in all 6 variables, which degree would become even higher if we wanted to replace these variables by a less redundant set of 2 variables. This main point in this paper is that having redundant variables is a **good idea** and it allows to greatly simply polynomial equations and effectively replace Semaev polynomials by some simpler and lower degree polynomials.

17.2 D93 - Another Family of Homogenous Cubic Equations

In this section we introduce another class of equations similar to D73. We have:

Theorem 17.2.1 (D93 Theorem). We consider the following set of variables on EC, a special form of ECC Code with 3 inputs and 9 outputs for any Weierstrass elliptic curve modulo a large P.

$$(P1, P2, P3) \mapsto \begin{array}{ccc} P1 - P3 & P2 - P3 & \\ P1 & P2 & P1 + P2 \\ P1 + P3 & P2 + P3 & P1 + P2 + P3 \\ P3 & & \end{array}$$

Again we look only at x coordinates of the 9 points. The variable numbering is as follows:

$$\begin{array}{lll} sx11 & sx12 & \\ sx21 & sx22 & sx212 \\ sx31 & sx32 & sx312 \\ sx43 & & \end{array}$$

If all the 9 points are distinct from the ECC neutral element ∞ we have:

```
+sx12*sx22*sx212 -sx12*sx22*sx31 -sx12*sx212*sx43 +sx12*sx31*sx43
-sx11*sx21*sx212 +sx11*sx21*sx32 +sx11*sx212*sx43 -sx11*sx32*sx43
-2*sx22*sx21*sx32 +2*sx22*sx21*sx31 +sx22*sx212*sx32 -2*sx22*sx212*sx312
+sx22*sx32*sx312 -sx22*sx31*sx43 +sx22*sx312*sx43 -sx21*sx212*sx31
+2*sx21*sx212*sx312 +sx21*sx32*sx43 -sx21*sx31*sx312 -sx21*sx312*sx43
-sx212*sx32*sx312 +sx212*sx31*sx312 = 0
```

Remark. Our D93 equation is also a homogenous polynomial of degree 3. Many other classes of such cubic homogenous equations with more than 9 variables exist. Many do NOT depend on the EC coefficients and work all the same for all sorts of Weierstrass elliptic curves modulo p. We omit them due to the lack of space.

18 Writing Equations For Complex Sets of Monomials

We are looking for methods for encoding computational problems on ECCs with more flexibility than until now, and with additional degrees of freedom, for example re-coding a problem described by S3 equations initially, following a general framework which could be as follows.

1. Encode a certain problem with equations on ECC.
2. Expand these equations with additional variables and by using some specific ECC Codes to show how new variables are related.
3. Generate a system of low degree equations which relate

There exist three basic methods to generate polynomial equations in such cases.

18.1 Three Basic Methods for Generating Equations

Imagine that we have some ECC Code which mandates some set of related elliptic curve points. In other terms we have some set of variables which are algebraically related, for example some specific relations hold between every pair of elliptic curve points. For example:

$$\begin{array}{ccc} P1 & P2 & -P1-P2 \\ P1+D & P2+D & -P1-P2+D \end{array}$$

This for every triple of points $P1, P2, D$ and avoiding singularities, no point should be a point at infinity. Furthermore we assume that some of these variables are known constants, some are unknowns. We want to write algebraic polynomial low degree relations mod P for the x coordinates of all these 6 points.

We want to algebra-ise this problem: write polynomial equations of low dedgree. We also preferably want to use ONLY quadratic or cubic equations, or only simple low degree equations. It should be noted that for some relatively low degree [13] low degree equations will be SUFFICIENT: their set of solutions will be EXACTLY identical to the set of solution, cf the notion of Describing Degree in [13]. There are many different ways to write such equations. Here are the three main methods which we use.

1. By interpolation, discovering equations by Gaussian reduction, as in our completeness studies in Section 13.3 and elsewhere.
2. By explicit formulas such MQ2,M3,D73,D93 equations, cf. Sections 17.1, 11, 11.2 etc. Such equations can sometimes be highly redundant. One simple method to handle is "random decimation[30]".
3. By expansion of a smaller set of more basic MQ2,M3,D73,D93 equations at higher degree and looking for degree falls.

Our experience shows that in general all these 3 methods work, or they always work starting from some threshold of degree or number of equations R generated, and this threshold can be determined by either theory or experimentation.

[30] For example if number of equations generated by these formulas grows asymptotically faster than the final rank, we simple generate less equations at random, and hope to achieve full rank.

19 The Semi-Invariant Set Method

As in Section 18 above we want to design and built some general-purpose ECC coding tools. Here we want to go one step further with an approach we call the **DS3 Coding**. The primary goal of DS3 Coding is to take an arbitrary ECC Code expansion with a certain imperfect semi-invariant property [new, not studied before] and write an overdefined system of equations of degree 3 with a good R/T ratio.

Here in Part V we do no longer work at degree 2 but we do work at degree 3, and the idea is again that the cryptanalyst should always remain at degree 3 maximum and avoid increasing the degree of equations which appear during the attack, this at the cost of adding additional variables.

19.1 Expansion Objectives

One key idea is to work with arbitrary S3 equations as a starting point, maybe for those example Section 5.7 even though this might seem very difficult and quite ambitious, and expand/augmment each 3 variables by adding extra variables, by a method which we do not specify at this stage. However we DO specify the **key objective** of such expansion below.

Definition 19.1.1 (Semi-Stability of ECC Codes).
We consider a family of ECC Codes with a parameter K. We say that our ECC Code Expansion is Semi-Stable by addition if for a pair of points selected at random in our code the probability that their sum on the EC is also present in our code is lower-bounded by a constant which does not depend on K.

19.2 Example of Semi-Stable Expansion

Here we give an example of ECC Code expansion which is Semi-Stable. The reader should not think this is THE example, it is just one example which is here to illustrate our concept. On Fig. 13 below we show a simple method of expanding an arbitrary point addition into a family of ECC Code expansions with a parameter K, where K is the number of lines on Fig. 13 below. Here the points $P1, P2$ are variables, D is a constant.

$$\begin{array}{rccc}
 & \vdots & \vdots & \vdots \\
 & P1-2D & P2-2D & P1+P2+2D \\
 & P1-D & P2-D & P1+P2+D \\
(P1,P2)\mapsto & P1 & P2 & P1+P2 \\
 & P1+D & P2+D & P1+P2+D \\
 & P1+2D & P2+2D & P1+P2+2D \\
 & \vdots & \vdots & \vdots
\end{array}$$

Fig. 13. Definition of RD3 Expansion with Parameter K, D is a known constant

Here if we select a point at random in the first 2 columns, with probability about 1/2 their sum is in the third column. Adding a point from the first column to a point from third column does not work, which is OK, as the probability that we want to make such an addition is constant and we are allowed to fail with a constant probability.

Remark. All our methods with semi-invariants can be improved if the ECC has a point D of a relatively small order. This will simply greatly improve many probabilities.

20 Converting Semi-Stable Expansions to Cubic Polynomial Equations Mod P

In the section we explain the general method on how the attacker can exploit the D73 type of equations. The method is a bit heuristic, or it requires a slightly stronger notion than Semi-Stability defined above. We recall that an expansion is Semi-Stable by EC addition if for a pair of points selected at random the probability that their sum is already present in our code at least a equal to a certain constant.

Definition 20.0.1 (3-Way Semi-Stability of ECC Codes).
We consider any family of ECC Codes with a parameter K which are semi-stable for EC point addition, cf. Def. 19.1.1 page 64.

We say that our coding method is 3-Way Semi-Stable if for a triple of points P1,P2,P3 chosen at random, the probability that the four points $P1+P2, P1+P3, P2+P3$ and $P1+P2+P3$ are simultaneously inside our initial expansion, is also lower-bounded by a fixed constant, independent on K.

Definition 20.0.2 (DS3 Coding for Semi-Stable ECC Codes).
For any family of ECC Codes with a parameter K which are 3-Way Semi-Stable for EC point addition, cf. Def. 20.0.1 above, for each triple of points P1,P2,P3 in the initial ECC Code, if the four points $P1 + P2, P1 + P3, P2 + P3$ and $P1+P2+P3$ are simultaneously inside our ECC Code expansion we write one cubic equation as specified in side Thm. 17.1.1 for each such set of 7 variables.

It is then easy to see that:

Theorem 20.0.3 (DS3 Ratio Theorem). For every 3-Way Semi-Stable ECC Code with $\mathcal{O}(K)$ points, our DS3 Coding method produces a system with the R/T ratio which is lower bounded by a constant.

Proof: We have $\mathcal{O}(K^3)$ cases $P1, P2, P3$ and by our 3-Way Semi-Stable assumption a constant fraction produces a D73 equation. The number of cubic monomials is also $\mathcal{O}(K^3)$.

20.1 Our Compiler Tool

We have implemented a command like tool for generating these equations for an **arbitrary** ECC Code expansion, not only when they are 3-Way Semi-Stable[31]. It works like a compiler takes as an input ECC equations with different variables, automatically detects dependencies and outputs a set of degree 3 equations mod p. The command line is as follows:

```
ax64 4741 filename.eqs /writeD73eqs
```

The input file can look like, for example:

```
//P=0x43 A=0x0 B=0x7
P1S001+P2S001=Q12S000
P1S001+P3S003=Q13S002
P2S001+P3S003=Q23S002
P1S001+P2S001+P3S003=Q123S001
Q12S000+P3S003=Q123S001
Q13S002+P2S001=Q123S001
Q23S002+P1S001=Q123S001
```

The output file is then:

```
modulo 67
-P3S003X*P1S001X*Q12S000X +P3S003X*P1S001X*Q23S002X +P3S003X*P2S001X*Q12S000X
-P3S003X*P2S001X*Q13S002X +P3S003X*Q13S002X*Q123S001X -P3S003X*Q23S002X*Q123S001X
+P1S001X*P2S001X*Q23S002X +P1S001X*Q13S002X*Q123S001X +P2S001X*Q12S000X*Q123S001X
-P1S001X*P2S001X*Q13S002X -P2S001X*Q23S002X*Q123S001X -P1S001X*Q12S000X*Q123S001X
 = 0
```

where X signs are appended automatically to variable names.

In addition our program can also generate and append MQ2 and MQ3 equations automatically which will be combined with D73 equations:

```
ax64 4741 filename.eqs /writeD73eqs /writeTriangles7D23
```

[31] This notion does not mean anything for a fixed system of equations, it is defined only for a family of systems of equations.

20.2 A Real-Life Example of Application of Thm. 20.0.3

We have developed a small demonstrator: a method of generating a 3-Way Semi-Stable expansion and then we generate a mix of cubic equations of type D73 and MQ2/M3.

```
ec2decomp 93973 256 K  /cubic
```

$$\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
P1-D & P2-D & P3-D & P1+P2-D & P1+P3-D & P2+P3-D & P1+P2+P3-D \\
P1 & P2 & P3 & P1+P2 & P1+P3 & P2+P3 & P1+P2+P3 \\
P1+D & P2+D & P3+D & P1+P2+D & P1+P3+D & P2+P3+D & P1+P2+P3+D \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

Fig. 14. An example of 3-Way Semi-Stable ECC Code with K Lines, D being a constant

The result is exactly as predicted by Thm. 20.0.3, we get approximately:

$$\lim_{K\to\infty} F/T \approx 0.19$$

21 Towards an Improved Ratio

We are not quite happy with $F/T \approx 0.19$ obtained with our software. In the next revision of this paper we are going to study the question whether and how it is possible to achieve $\lim_{K\to\infty} F/T = 1$ by this method. We claim that this is possible to achieve in some case by including also the D93 equations of Thm. 17.2 and few other disjoint families of cubic polynomial equations. In the following section we do an initial feasibility study which could disprove our claim but cannot confirm it [this requires generating more distinct classes of equations].

21.1 Initial Feasibility Study

We start by a computer simulation which show how many MAXIMUM cubic polynomials can be generated for some family of ECC Codes.

It could be true that there is no chance to achieve $\lim_{K\to\infty} F/T = 1$ by our or any similar method. Or that we can get $\lim_{K\to\infty} F/T = 1$ easily. We just need to add some more equations in addition to $D73$ to achieve a higher rank, We have already established that a combination of expanded MQ2/MQ3/D73 does allow to achieve a ratio F/T better than 0.19. However we could also use D93 equations and some other types of equations [it is easy to see that other similar classes of equations exist].

21.2 To Be Done

We plan to extend our software demonstrator above to add D93 equations and more similar equations. This will certainly improve the R/T ratio. The answer whether or not we can achieve $\lim F/T \to 1$ by this method or/and how will appear in a subsequent update of this paper.

At this moment we present a simpler example where it seems that we have achieved it already.

21.3 RD3 - More Than Just D73 Equations

Now we consider the following set of points. $P1$ and $P2$ are two unknown points on the curve, D is a known constant point on the curve.

We recall our definition of RD3 equations which is the same as on Fig. 13 previously.

$$(P1,P2) \mapsto \begin{array}{ccc} \vdots & \vdots & \vdots \\ P1-2D & P2-2D & P1+P2+2D \\ P1-D & P2-D & P1+P2+D \\ P1 & P2 & P1+P2 \\ P1+D & P2+D & P1+P2+D \\ P1+2D & P2+2D & P1+P2+2D \\ \vdots & \vdots & \vdots \end{array}$$

Fig. 15. RD3 Equations with Parameter K, D is a constant

Then for each point listed here we write polynomial equations which relates these variables. this could be done by any of the methods described in Section 18. Below we provide a more specific method.

1. First of all, if K is sufficiently large, with a certain probability which is lower-bounded by a constant, for every pair of points selected in the first 2 columns, we can find [32] in the third column one point such that the 3 points add to ∞. Therefore we can write the Semaev equation S3 for each such 3 points. This gives $\mathcal{O}(K^2)$ equations of degree 4.
   Unhappily this is highly unsatisfactory because the number of monomials in these equations grows[33] as $\mathcal{O}(K^3)$. We have no hope to obtain an interesting attack using just these equations.
   In fact we are going to ignore totally these equations as their degree is too large. Instead we write other equations.
2. Quite happily we can write a lot more equations: we can write $\mathcal{O}(K^3)$ equations of type MQ2 written following the explicit formula given in Section 11. These equations are NOT in the ideal generated by S3 equations and

[32] For example if we select $P1+2D$ and $P2-5D$ we need to use $-P1-P2-3D$.

[33] It does NOT behave as $\mathcal{O}(K^4)$ because all monomials of degree 4 are either of the form $ix1^2 * ix2 * ix3$ or $ix1^2 * ix2^2$.

following Section 13.3 they are linearly dependent and their rank is only $\mathcal{O}(K^2)$.

3. More importantly, we can also write $\mathcal{O}(K^3)$ equations of type MQ3 written following the explicit formula given in Section 11.2 All these equations ARE linearly independent, cf. Thm. 11.3.1.
4. Finally we also have the D73 equations written following the explicit formula given in Section 17.1. It is easy to see that we have $\mathcal{O}(K^3)$ such equations, this is because if K is large we can generate an equation D73 for every point in one column and every pair of points in another column, with $P3 =$ a certain multiple of D, again with a probability which is lower-bounded by a constant.
   There are $\mathcal{O}(K^3)$ monomials in these equations.
5. In addition we can also generate a much larger set of equations of type $D93$, at random. It is is easy to see that we have also $\mathcal{O}(K^3)$ such equations, and $\mathcal{O}(K^3)$ monomials which appear in these equations.

We are now going to provide an upper bound for the number of equations which can be obtained by this method. We have not yet implemented a full complete method with mix of D73 and D93 etc, so this is a preliminary result which shows what is the maximum possible rank which is achievable here. These simulations can be redone by the reader with our software as follows:

```
ax64 9390419 256 K 99223
```

Our simulations show that for every K we have approximately:

$$F(K) = 4.5K^3 - 7.4K^2 - 5K + 5$$

$$T(K) = 4.5K^3 + 9K^2 + 2.5K + 1$$

It seems that we achieve our goal:

$$\lim_{K\to\infty} F/T = 1$$

This needs to be confirmed [larger simulations pending, $K = 16$ requires 33G of RAM and takes > 1 week to complete].

Part VI

How To Encode Constraints a.k.a. Factor Basis Question

22 Objectives and Postulates

Until now it is possible to claim that we have done nothing and achieved nothing tangible. Yes we have demonstrated that Semaev polynomials can be replaced by very much simpler and yet more dense systems of equations which also are very highly overdefined. However we have not yet produced a single system of equations which overall allows to solve a point splitting problem in the full setting: with constraints which limit the values which can be taken by various unknowns to specific subspaces. We call a “factor basis” any such space.

22.1 Why We Need a Factor Basis

These constraints is required in order to have a unique and meaningful solution. We recall that this requirement comes from the idea of designing an index calculus algorithm for the ECDL problem. This was initially claimed very difficult or close to impossible at Crypto’85 by Victor Miller, cf. [40] and in 2004 there was a first attempt to define such an approach cf. [46]. We refer to [42, 17] for more explanations on how point splitting and index calculus are related.

22.2 A Rough Idea - Parallel Constraints Method

We first sketch what kind of solution we are looking for. Then we are going to study the actual solutions in more details.

Suppose we have a property T that is satisfied (denoted $T(P) = 0$) for about half the points on a curve $E(\mathbb{F}_p)$. We consider some fixed constant points $S_1, \ldots, S_k \in E(\mathbb{F}_p)$. Then we could define a factor basis Fb as follows:

$$Fb = \{P \in E | T(P) = T(P + S1) = ... = T(P + Sk) = 0\} .$$

and we expect that this set comprises about $q/2^k$ points where $\#E(\mathbb{F}_p) = q$.

In fact our properties will not be defined for points on the elliptic curve but for their x coordinates, arbitrary integers modulo p. Moreover not every such property will work well. For example, we need a way to code our property efficiently with polynomial equations. Moreover these polynomial equations should be very dense and highly connected following our general philosophy. We can consider that a ”good” property $T()$ is such that we have simultaneously:

F1. Extremely few new variables are added and we have “very few” new non-linear monomials.
F2. It achieves some sort of “dense connection topology”:
F2a. There are lots of algebraic relations between new and old variables.
F2b. Same for relations within new vars/constraints themselves: a sort of ”parallel connection of constraints” where different properties accounting for 1/2 factor each can interact somewhat ”strongly”.

Before we present our “parallel constraints” solution which will be based on MQ2() polynomials, we are going to expand on F1+F2 and reflect on more detailed requirements such a good solution might or should have.

22.3 Our Detailed Postulates

Let us envision what would be a dream method for breaking the ECDL problem through an index calculus target point decomposition approach. We will however restrict to methods which follow the general philosophy of highly overdefined / dense topology with an adjustment variable K of the present paper. We do not exclude that other more traditional methods also exist, cf. Section 3.6.

Here is our list of postulates:

1. **General Add-On Method**. It should be possible adapt the solution to so as to solve the point splitting problem for any $M \geq 2$. The constraint methodology should work as an plug-in addition to many different systems of equations such as defined in this paper and reduce the number of solutions they have.
2. **Flexibility**. The method should be flexible enough to allow for a lot of adaptations, tuning and optimizations. It should be able to encode constraints of arbitrary strength, for example asymmetric ones: for some points Pi could lie in a different or larger subspace chosen by the attacker.
   However we do allow the method to impose on us the type of constraints we will use, to make the method work well. The only requirement is that the attacker should be able to freely[34] adjust the probability that a random point satisfies some equations which encode these constraints or this particular factor basis.
3. **Highly Overdefined**. The R/T ratio should be quite high.
4. **Low Polynomial Type Complexity**. The method should be such that we have good hope that it will be polynomial for any value of M, or maybe subexponential. For example the total number of polynomials and equations generated should follow some formulas which are polynomial in K.
5. **High Density**. One monomial should appear in a large number of equations, preferably this number should not be a constant but something like $\mathcal{O}(K)$.
6. **High Connectivity**. The system of equations to solve at the end should not have a serial or other poorly connected topology where different equations and variables can be isolated and it could be extremely difficult to relate variables in remote parts of our system of equations. If possible the system of equations should have no center and periphery but two variables or monomials chosen at random should be somewhat always related and connected.
7. **Local Elimination**. In order to eliminate one arbitrary monomial, one should not need to manipulate the whole very large system of equations. Instead there should be algebraic shortcuts which allow to eliminate this monomial by some simple closed formulas or algebraic relations generated for a subset of equations, cf. for example Section 15.1 or Thm. 13.9.4.
8. **Clarity/Explicit Degree Falls**. The method should explicitly define and analyse the principal classes of degree falls which occur in the computation,

[34] We can remark that this goal is not achieved in [42], which methods are very rigid and exploit some (very few) multiplicative subgroups of a fixed size which exist for the base field mod p for some NIST elliptic curves.

study and enumerate them, so that the attacker will not have to wonder why on earth this method actually works [or maybe it doesn't] as it is very frequently the case in algebraic cryptanalysis.

9. **Elegance**. For example the method could follow "one equation breaks it all" philosophy which have previously achieved in algebraic cryptanalysis [21, 22]. If we can find just **one** algebraic equations of a certain type with certain characteristics, we can generate many copies of this single equation and crack our problem just by using these equations.

It appears that in this paper we have already shown some good examples of techniques and methodology to achieve **all** the above objectives. This except the following objective below, which has been very badly neglected so far in this paper, and for good reasons[35].

10. **Unique Solution**. The big overall system of equations we define should[36] have **a unique solution**.

Until now we have not provided a single clue about how this problem could be solved except a hint that **ideally** the coding of constraints should use more or less the same monomials which we already use. This objective is however probably totally unrealistic. Therefore we are going to relax it in a way which is very much following the general spirit of this paper [adding new variables] and we are going to postulate that:

11. **Constraints Economy**. The coding of constraints should **add as few new variables as possible**. If M is fixed, the number of new variables should be a logarithmic rather than polynomial number and should, if possible, not depend on the parameter K.

Now we are going to provide our first method which shows how to constrain the elliptic curve points to **subspaces of desired size** in such a way that we do NOT diverge from our general highly overdefined and highly connected philosophy.

[35] This question has so far resisted the algebraic cryptanalysts and very few solutions have ever been proposed, cf. [46, 42]

[36] This condition is necessary and sometimes sufficient, cf. [7]. It appears that in for equations over small finite fields highly overdefined equations which have a unique solution are ALWAYS efficiently solvable, no counterexample is known. It is a lot less obvious that this also holds for equations over larger finite fields, and we conjecture that it does.

22.4 Two Faulty Solutions

First we will describe two solutions which do NOT work well, in order to better understand what kind of solution we are looking for.

One method is that for every variable of type $P_i + S_j$ in our previous methods[37], we add a new variable which is a square root of this variable. For example we write:

$$P1X = T1^2$$

Which will be true with probability 1/2 and we can achieve any probability of type 2^{-N} for any N by adding several such new variables for different shifts $P_i + S_j$ for the same variable P_i. This method has good flexibility, cf. postulate 2. but nevertheless we believe that it will NEVER work. The main reason for this is that relationship between $T1^2$ and $T1$ is a sort of one-way function. This as far as very simple polynomial equations are concerned. Our algebraic solver operating at very low degree will **not** be easily[38] able to figure out any equations on $T1$, when given some equations involving $P1X = T1^2$ and other variables.

This is visibly a very bad method to encode constraints and it brings to us to make one additional postulate about how the constraints should operate:

12. **Constraints Uniformity**. Additional variables which define constraints should not be such that from various sets of monomials it is always equally easy or equally difficult to compute other monomials through very simple polynomials, avoiding situations where one direction would be more difficult.

22.5 Another Faulty Solution, Already A Lot Better

Another simple solution is that for each variable Pi or $P_i + S_j$ or maybe also a variable such as $P_i + P_j$ like in Thm. 17.1.1, to keep it simple we call it $P1$, we add a few extra variables, as few as possible, and one or a few extra polynomial equations F such that for example:

$$\mathbb{P}\mathrm{r}\left(F(P1, P1', P1'', P1''') = 0\right) = \frac{1}{2}$$

This is actually possible to do with **just 3** extra variables, as we will discover later. However this method is still not OK. The main complaint will be that new variables will only connect to $P1$. Not at all to thousands, possibly millions of **other** variables we generate. We need to do a lot better. We postulate that:

13. **Dense Inter-Constraints Topology or "Parallel" Connection of Constraints**. There should be a way to related ALL new variables added by the constraints **directly** to each other (a little bit like periphery to periphery connections). Not only that we have coded one variable say $P1$ and 3 extra variables $P1', P1'', P1'''$ and these 3 extra variables are connected to our equations just at one single point, $P1$ itself. Remark: this requirement 13. can be seen as basic/weak variant of F2b. postulated before.

[37] Here S_i is expected to be a constant known/chosen by the attacker like in Section 6.1, rather than another free variable, for example like in Thm. 17.1.1.

[38] Not without using polynomials of very high degree or many extra variables.

This means that quadratic monomials which are products of new variables from different constraint-imposed sets of new variables should essentially have a totally flat topology and interact uniformly with each other. It should be parallel rather than serial [39] constraints topology where different things which are done in parallel interact strongly.

Finally we will require new variables to interact particularly strongly between themselves. We postulate that:

14. **Strong Inter-Constraints Interaction**. For every variable $P1 + S_j$ and $P1 + S_j$ for which distinct independent constraints are written and for any subset of $K' + K'$ variables inside the new variables added for each variable, excluding any "old" variables $P1 + S_j$ themselves one should be able to write a system of quadratic equations which related these $K' + K'$ variables and such that
$$\lim_{K' \to \infty} F/T = 1$$
Remark: this requirement 14. can be seen as a strong version of F2b. postulated before.

23 First Proof of Concept or How to Achieve All Our Objectives

Now we are going to show a simple method to achieve all the objectives stated above which our first proof of concept for constraints coding [a.k.a. factor basis problem]. Actually the solution which we present is remarkably simple and was already contained in this paper, we have just not noticed it yet. We also stress a fact that it is just one solution and that many other solutions which satisfy our postulates surely do exist. Therefore we consider that formulating these postulates is at least as important as finding one such solution.

As already hinted above, all our objectives can be achieved by adding just 3 variables per 1/2 probability constraint which we want to add. In an index calculus attack we want to impose approximately (n) such constraints overall if M is fixed, cf. later Section 23.4 and Section 24.1.

The key result is as follows, which is a sort of "completeness" or "Describing Degree" cf. [20] result for our most basic linear ECC Code studied in this paper and which is very closely related but NOT[40] identical to the fact that MQ2 equations alone are "complete" at degree 2 cf. Section 13.3. Here is our key result which we can use in coding of contraints.

[39] We refer to [42] for an example of a serial constrain method where many new variables are added and each variable is connected just to the previous one and the next one. Not very good.

[40] The difference is more or less the same between "Describing Degree" and "Conditional Describing Degree" or counting the number of solutions to equations when one variable is fixed, cf. [20] for some similar considerations.

23.1 Basic Result Need to Our Later Constraint Coding Technique

We consider our basic linear ECC Code defined as follows, with some distinct non-zero constants chosen by the attacker:

$$P \mapsto [\ P + S_0, \quad P + S_1, \quad P + S_2, \quad \ldots \quad , P + S_{L-1}]$$

By abuse of language, we call DEx^L_{MQ2} a system of equations defined as union of MQ2 equations for each triangle, i.e. for each triple of variables, following formulas of Section 11, or generated using a more efficient decimated method which is quadratic and not cubic in L, cf. Section 13.2.

Let us consider **any** subset of 4 distinct variables out of L and for convenience we name them $x1, x2, x3, x4$.

We have the following result:

Theorem 23.1.1 (Constraining Power of $T_L(.)$ DEx^L_{MQ2}). For any $L \geq 4$ and for any variable $x1$ we have:

$$\exists_{y1} s.t. (x1, y1) \in E(\mathbb{F}_p) \iff \exists_{x2\neq 0, x3\neq 0, x4\neq 0, \ldots, x_L\neq 0} s.t. DEx^L_{MQ2} \ \text{ are all satisifed}$$

In other terms only 3 extra vars and simple quadratic equations MQ2 we have enough constraining power to achieve a constraint true with probability 1/2 as desired.

Proof: It remains an open problem how to prove this theorem. For small finite fields the exact value of "Describing Degree" cf. [20] is usually confirmed by brute force computer simulations. For larger fields (here) this would not work we expect here a sort of computer proof using a Gröbner basis computation or a result on existence of some equations which are uniquely solvable due to their specific parametric form. Heuristically our result is something which shows that MQ2 equations are "powerful enough" and contain exactly all the possible solutions intended and nothing more. We naturally expect such as result to be true starting from a certain threshold value L. The fact that $L - 1 = 3$ new variables suffices is quite surprising and we did not expect such a good result(!).

Remark. We see that MQ2 equations are not only excellent in terms of "economy", they use not more than 3 non-linear monomials each, but also in terms of constraining power.

Further Remarks. It seems that our DEx^L_{MQ2} equations have an all-or-nothing property, if one point is on ECC, all are. Exact result about this remains to be formulated. Most if not all conditions of type $x2 \neq 0$ can also probably be removed from the theorem above. Also we expect the same result to hold for many other ECC Code expansion codes seen in this paper.

23.2 Our Constraint Technique For One Expanded Variable

Here is how exactly we propose to add N contraints true with probability 1/2 to any variable P. The variable P can already be one of the variables from an expanded set of variables, we aim at a dual expansion solutions with connections which will be achieve by linear equations over the base field mod p. For example we initially we expand P as follows

$$P \mapsto [\ P+S_0, \quad P+S_1, \quad P+S_2, \quad \ldots \quad , P+S_{K-1}]$$

Then for a subset of N variables which without loss of generality we assume to be the first N, we are going to transform the x coordinates of these variables, for example as follows:

$$P \mapsto [\ a(P+S_0)_x+b, \quad a(P+S_1)_x+b, \quad \ldots \quad , a(P+S_{N-1})_x+b]$$

These are the N variables which we want to constrain with probability 1/2. For each of these variable we are going to add $L-1=3$ or more new variables [$L=4$ is the minimum required]. Then we apply the same constraint $T(.)$ to each of these variables. Our constraint $T()$ will be defined as follows:

Definition 23.2.1 (Constraint $T(.)$ Defined With DEx^L_{MQ2}). We will say that $T(x1)=1$ if an only if

$$\exists_{y1} s.t. (x1, y1) \in \text{curve} \quad E(\mathbb{F}_p)$$

and the actual coding is done with DEx^L_{MQ2} following the equivalence result of Thm. 23.1.1, i.e. without using any variables for the y1 coordinate(!).

23.3 More Details on Coding Constraints

Here is a more detailed explanation step by step:

1. We select N small integers a and N affine terms b.
   As it turns out it does not matter if these integers are distinct what exact values they take, we should however avoid some very peculiar multiples [or simple endomorphisms] which exist for certain elliptic curves such as secp256k1.
2. For up to N shifts P_i+S_j, which do not have to be distinct, we write a LINEAR equation modulo p as follows:
   $$x1^{(i,j)} \overset{def}{=} a\left(P_i+S_j\right)_x+b$$
   The name $x1$ is used here because this variable is going to become THE x1 variable in Thm. 23.1.1, or we simply add 3 more variables $x2^{(i,j)}-x4^{(i,j)}$ per each constraint.
3. In fact we do NOT add a new variable for $x1^{(i,j)}$ because it already exists in a certain sense: it can be replaced be an affine transformation of an existing variable $a\left(P_i+S_j\right)_x+b$.

4. In other terms we write
$$T\left(a\left(P_i+S_j\right)_x+b\right)=1$$
where T is defined following Def. 23.2.1 and code with $MQ2$ equations following Thm. 23.1.1 using many times the closed formulas provided by Thm. 11.0.5 page 38.
5. We have added $3N$ variables for N expanded variables for which we want to impose a constraint which overall has the probability of being correct 2^{-N}.

23.4 Coding Constraints For Multiple Points

More generally, if we want to add constraints to a problem of splitting a point in M parts like in our first and most general solution of Section 6.1, we could have N/M blocks of P_i+S_j variables with the same i, and we want to impose a constraint overall true with probability $2^{-N/M}$ for each of these P_is, for this we need to add $3N/M$ variables $x2^{(i,j)}-x4^{(i,j)}$ for each i.

Following Thm. 23.1.1 for each variable our constraints are equivalent to fact that each special $x1$ as defined above in each case, corresponds to some point on the elliptic curve which gives the probability of $1/2$ approximately[41] for each 3 new vars.

[41] Not exactly, cf. well known Hasse theorem.

23.5 Testing Our Parallel Constraint Method

We have done a number of experimental tests to show that this method seems to work well and that the events seem to be independent and probabilities we obtain are as expected.

23.6 Is Strong Interaction Achieved?

It remains to convince the reader that we have also achieved our postulate 14. which was the **strong** interaction objective.

For this we have done simulations on how many quadratic equations exist for two subsets of $K' + K'$ variables i.e. between the two expanded sets variables added for two different variables which we assume to be related on the ECC, for example P_1 and $P_1 + S_1$. So for example we are adding two constraints with twice $K' - 1$ new variables, which are:

$$T\left(2\left(P_1\right)_x + 1\right) = 1 \qquad AND \qquad T\left(5\left(P_1 + S_1\right)_x + 7\right) = 1$$

We will of course assume that the constraints $T()$ are identical which means that the ECC constant differences constants used in each $T()$ are identical. However we can vary the affine constants such as 5 and 7 above, and for each constraint T these constants are going to be different. It turns out that typically[42] the result does not depend on these affine constants. The command line for our testing software is:

```
ax64 9390417 256 K' 28211728
```

It turns out we have systematically about:

$$F \approx 2K'^2 - 16K' + 36 \qquad T = 2K'^2 + K + 1$$

This formula is not always accurate. A better prediction is obtained if we allow a different polynomial to be valid for $K' \leq 8$ and a different one for $K' > 8$. Then we obtain a prediction which is more accurate in practice:

$$F = \begin{cases} K'^2 - 3K' + 2 & \text{when } \ 2 \leq K' \leq 9 \\ 2K'^2 - 15K' + 24 & \text{when } \ K' > 9 \end{cases} \qquad T = 2K'^2 + K + 1$$

We omit the fact that all these equations can obviously be generated by close formulas as all other equations in this paper, thought the exact method would be a bit tedious to specify. In future update of this paper we could develop closed formulas to compute these equations but we already see that we will obtain what we postulated:

$$\lim_{K' \to \infty} F/T = 1$$

Our 2 sets strongly interact with each other.

[42] An attacker which wants at any price avoid any special cases might want to select these constants at random. However it is also known that this type of attack can be reinforced [additional linearly independent equations] if the constants are well chosen, see for example Thm. 20.0.3, the whole part V and Section 30.5.

Remark 1. What we describe here does ONLY work starting from a certain threshold. It is possible to see that for a subset $K' + K' = 8 + 8$ variables on each side, absolutely none non-trivial equations are generated, just twice the 21 equations which can be generated using $MQ2()$ formulas for triples of variable on only once side, they involve only 3 variables each, and their number follows the formulas of Section 13.8 cf. also Table 7 page 45. If no more equations were generated for larger K' there would be NO HOPE to have $\lim_{K'\to\infty} F/T = 1$. However for larger K' values, and more precisely for $K' \geq 10$ a lot more equations exist which are more complex than $MQ2()$ and involve more than 3 vars taken from both sides.

Remark 2. A similar threshold occurs for quadratic monomials present in these equations. Until $K' <= 9$ there exist many products of variables which are absent in these equations. then for $K' \geq 10$ all possible monomials are present.

Remark. This threshold phenomenon is very much the same as what is frequently seen in algebraic attacks with ElimLin: until our attack is sufficiently large, the equations which provide the interesting asymptotic growth do NOT exist at all, they start to exist starting from a certain threshold.

23.7 Another Type Of Interaction

Another question is when there are two affine versions of the same point. Again we have two subsets of $K' + K'$ variables and actually 1 variable will be shared, $x1$ in both sets are the same variable.

$$T\left(2\left(P_1\right)_x + 1\right) = 1 \qquad AND \qquad T\left(3\left(P_1\right)_x + 5\right) = 1$$

The command line for testing this cases is:

```
ax64 9390417 256 K’ 28211729
```

We have [most of the time exactly]:

$$F = 2K'^2 - 9K' + 11 \qquad T = 2K'^2 - K$$

Here again we have strong interaction:

$$\lim_{K'\to\infty} F/T = 1.$$

Remark. Here the threshold is lower. Interesting things happen when $K' \geq 6$. Starting from this point a lot of additional equations which involve more than 3 variables exist, while such equations do not exist for a small K' value and again, without these extra equations there would be no hope to have $\lim_{K'\to\infty} F/T = 1$.

23.8 Limitations

All this does not mean that our $T()$ constraints for two **distinct** ECC points will interact strongly. This is impossible and cannot be mandated in all generality. This is because distinct points in one of our methods can be totally independent.

However probably we achieve here a very strong objective. At this moment we are not aware if there exists a plausible stronger set of objectives for the constraints/factor basis. If it is possible to ask “more” from a Factor Basis than the properties 1.-14. we already achieve.

Part VII

Approaching The Big Challenge

24 Putting It All Together - General Case

In this paper we do not yet propose a full mature solution, but it appears that by using the techniques contained in this paper it should be possible to convert the problem of index calculus on the most popular elliptic curves of large prime type such as used by millions of people every day, to a system of equations with a parameter K' such that it is very highly overdefined, has very dense highly connected topology and has a unique solution. Overall it seems that we can achieve all our objectives in this paper except that we do not always achieve $\lim_{K' \to \infty} F/T = 1$ or not yet[43].

24.1 First General Solution For Any M

One natural method to design an index calculus algorithm for ECDL problem will be to:

1. Simply combine our high-level construction Ex_M^K of Section 6.1 with MK variables, cf. page 25.
2. We recall that each variable can be constrained with probability 1/2 with one $T()$ constraint.
3. Let $L \leq K$. We will add $M \cdot L$ such constraints
   Each constraint adds $K' - 1$ or more extra variables added for $M \cdot L$ constrained variables which are some or all of the variables used inside the Ex_M^K equations.
4. We assume that all $T()$ contraints will be applied to distinct variables from the initial K-times expansion of Ex_M^K.
5. This gives a total of $(K' - 1)ML$ extra variables added following Section 23.2 and Section 23.4.
6. Here L should be equal to $L = n - n/M$ for an n-bit elliptic curve which allows each set of L constraints to be true for approximately $2^{n/M}$ ECC points.

This attack probably does the job already [of solving arbitrary point decomposition problems] with a good Gröbner basis software as the last step. This even though it will not always or we not[44] yet achieve $\lim_{K \to \infty} F/T = 1$.

24.2 Other Solutions For Any M

Another method for solving arbitrary M problems would be to start with a system initially encoded with S3 polynomials following either Section a serial topology cf. Section 5.7 or **a tree topology**, cf. Section 5 of [34].

Then we apply the techniques of Part V with expansion of arbitrary sums of points and semi-invariant sets.

[43] In fact in this paper and future releases of it we intend to consider and study both weaker and yet stronger objectives together with questions of complexity/solvability of low degree multivariate equations mod p under various conditions.

[44] Therefore probably this attack is not yet a candidate for a really fast quasi-polynomial time attack.

24.3 Road Map - From General to Particular

In the pages that follows we will describe a particular method to instantiate and implement both the sort of attacks we have sketched in the two previous sections and many other. We will later further specify and implement it in a simplified variant for $M = 2$, however the method works in principle for any M and with a variety of ECC coding attacks described in this paper [and many other]. One problem is that these attacks can be quite technical. We are now going to describe an alternative approach which will allow us to study the attacks such as outlined above, without getting too technical or leaving some difficult optimization questions for later.

24.4 Zero-Equation Local Approximation Paradigm

We want to study attacks where initially we do not know well how to write the equations or we do not know [yet] which equations we want to use (!). We also want to be able to see if the equations obtained by constructive methods [such as in this paper] are suitable and how they compare to other equations which might exist and be used in cryptanalysis. This is the goal of our "Zero-Equation Local Approximation" paradigm which we will study now. First we will explain the general philosophy of this approach which is quite general and powerful and could be used elsewhere. Then we will apply this methodology [in the following pages] to point splitting problems and with $M = 2$ in particular. We do not claims that this approach is going to be particularly fast or efficient. Instead it is meant to lower the entry barriers for researchers to study implement and test [up to] a first full working attack for arbitrary point splitting problems (and for other problems). It is also suitable for a massively parallel implementation.

We call this a **"Zero-Equation"** method as initially it does **not** require a careful optimized implementation with very good expertise in adding well chosen new variables, and writing well-chosen equations such as for example in Thm. 17.2.1. It is a **universal** method which will **automatically discover all the equations it needs** by local approximation. The price to pay for this is possibly[45] low speed.

To be more precise, it will generate a number of consistent ECC examples for one subset of variables, and recover the equations by computing a kernel of a certain Macaulay matrix [classical method in algebraic cryptanalysis]. This will be repeated many times generating many additional equations.

Our "Zero-Equation Local Approximation" method should be viewed as a framework which allows to construct a variety of attacks and we expect in the future releases of this paper to present several different attacks and their implementations which follow this framework more or less exactly.

[45] Though actually if used and optimized carefully, this method could possibly be really good, as it is a real alternative paradigm for algebraic cryptanalysis. The dominant paradigm in algebraic cryptanalysis is to work with equations of certain degree and generate degree falls. Here we generate equations of a certain degree directly without degree falls and without manipulating equations of higher degree which is a big advantage compared to classical Gröbner basis methods [1, 2].

24.5 Detailed Description of Our Zero-Equation Local Approximation Framework - Suitable For Any M and Beyond

In this section we provide a high-level description of our new framework.

1. We start with a problem of splitting a point in M parts.
2. For each of M parts we apply a certain linear code or ECC code expansion.
3. We also consider an additional ECC code expansion for pairs or triples of points, cf. for example Thm. 17.1.1. [this part is optional].
4. We select a certain number ML of points and apply constraints $T()$ to them which expands these variables K' times to MLK' variables.
5. We call a "domain" a set of variables which are closely linked, i.e. knowing 1 or 2 variables determines all the other, or knowing 1 variable gives 2 possibilities for another variable.
6. We consider several additional small integers $K1$, $K2$, possibly more than 2.
7. For some subset of $K1$ variables in one "domain", and for some other subset of $K1$ variables in another "domain", we consider ONLY up to $(K1+K2)K'$ variables in these 2 domains.
8. Then we consider all possible I/O equations cf. slide 76 in [19] where I=one domain, O=second domain which satisfy a subset of all our ECC relations desired [specified by $T()$ and other expansion steps above] which involves the variables in 2 domains and no other variable.
9. We generate some examples consistent with our ECC relations and consider mod p relations involving the x coordinates of these ECC points.
10. We compute a basis for such equations by computing a Kernel of a Macaulay matrix.
11. This part of attack is exponential in $K1, K2$, due to the necessity to select many samples of ECC points which satisfy [up to] $K1 + K2$ contraints of type $T()$.
12. The values $K1$ and $K2$ are however expected to be very small and they should be constants which should not depend on other parameters of the attack such as K'.
13. This sort of attack can be improved and optimized later by generating equations by an optimized method in an ordered way or/and closed formulas[46] [optional].

We describe this general attack framework to show that developing some broad-brush form of a working attack on a given ECC problem can be very simple, optimizing it and making it really efficient can be a lot more difficult. One such attack with a working implementation is found in Section 25.1 below.

[46] For example all the equations in Section 13.5 can be generated by a [decimated] union of a set of MQ2() and MQ3() equations.

25 Practical Solutions for Point Splitting with $M = 2$

25.1 A Particularly Simple Solution for $M = 2$

Here is the simplest solution we can think of for the basic point splitting problem with $M = 2$ AND following our framework.

1. We will avoid writing ANY equations DEx_2^K for $P1, P2$ and we will apply the "Zero-Equation Local Approximation" approach of Section 24.4 above.
2. This amounts to using primarily the quadratic equations which we have studied in Section 23.6 and Section 23.7 indirectly.
   The two approaches are somewhat equivalent but NOT identical and will give different results in terms of time and memory complexity/cost. At this moment we ignore these questions and we concentrate on more basic questions of how many independent equation can be generated [the more the better].
3. In addition [optional] we can also use so called "higher order" extra equations which will study later in Section 29.
4. More precisely we will code K constraints of type
   $$T\left(a_i \left(P_1\right)_x + b_i\right) = 1$$
   and K similar constraints $T()$ for $P2$, with $K' - 1$ extra variables each.
5. We aim at $K = n/2$ and K' can be arbitrary.
6. We do not specify at this moment how to write the equations. We just need to concatenate many copies of the equations of Section 23.6 with many copies of the equations of Section 23.7, recovered by local interpolation as in Section 24.4 and 24.5 and this for each pair of sets of K' variables.
7. It goes without saying that the ECC constants used in each $T()$ constraint are the same in each set of K' points which was also assumed in Sections 23.6 and 23.7.

Now we are going to study and implement our attack.

26 Experimental Setup

This is our a [first and tentative] software demonstrator for this attack. It also displays our predictions for the number of different types of equations. It is our first implementation, of a full working attack in a fully realistic setting. It also follows quite closely the Zero-Equations Approximation-like framework of Section 24.5 which is neither a bug or a feature, this is simply our first version which is easier to implement and will be later re-developed and optimized.

Currently it only generates equations and does not attempt[47] to solve them. Our first objective is to check if the rank obtained is equal to the number we can or will predict [in the pages which follow]. In general it is almost always as predicted. Current implementation is quite slow, it is incremental and redundant [the same equations are generated many times for random subsets which save

[47] However the reader can immediately try to solve them with their favorite software.

memory but increased running time]. We expect to be able to improve this software and provide a full working working solution for point splitting which also solves the equations in some future update of this software.

Currently the command line for our tool is:

```
ax64.exe 9390489 652 K' K K1 K2 0x_point_Q_in_hex /disableparallel
[/id1] [/maxorder2] [/continue] [/allowsquares] [/addmult3Q/2]
```

Below we explain the role of different parameters:

1. The number 652 is a shortcut for NIST curve P-256. A number of other values and special codes are supported. For example 29 or 73 allows one to use our suggested choice of toy curve on 29 or 73 bits.
2. At level 1 of expansion, two variables $P1, P2$ are expanded into $2K$ variables.
3. At level 2 each of these variables becomes $x1$ in some constraint the $T()$ and we add $K' - 1$ extra variables per each variant of $P1, P2$. All exactly as in Sections 23.3 and 23.2.
4. The value K is chosen automatically or manually.
5. If we put $K = 0$ the number K used which is selected automatically as the default value $K = \lceil n/2 \rceil$ which is the optimal amount of constraints needed for a given curve for $M = 2$ case.
6. In general the expected number of solutions to our systems of equations is 2^{n-2K}.
7. The two parameters K1,K2 are 1,1 by default, and are the number of sets of K' considered by our program in one unitary Gaussian reduction. We will use up to $(K1 + K2)K'$ variables at one time, and in fact less as some variables are redundant, cf. more detailed analysis below.
8. For example we select $K1$ out of K variables per each variant of $P1$ at level 1.
9. This generates a set of quadratic equations each time for a given two-sides set of variables.
10. Later all these equations are combined for another subset and joint rank is investigated.
11. The affine coefficients are chosen to be small integers. they are identical in $P1$ and $P2$ expansion. In current version also $b_i = 0$ for easier debugging, which however apparently make no difference of any sort and is expected to change in future updates.
12. If we use command line option /id1 it means that $a_0 = 1$ and $b_0 = 0$. This means that one point on one side and another point on the adder side add and give Q. The reason why we make our live harder than required is that in the default version ALL the sets of K' are identical and indistinguishable. None interact stronger than other.

    The option /id1 could be considered later as a bonus which further improves the attack, but makes analysis harder. In addition it is easy to see the number of additional equations which could possibly be added due to this improvement can be at most $\mathcal{O}(KK'^2)$ while in the default version we expect to

generate roughly $\mathcal{O}(K^2 K'^2)$ equations. Therefore we have good reasons to neglect the improvement in our fist analysis.
Furthermore no difference of any sort was ever observed for larger curves [some differences were observed for small curves].

When K1,K2 are just 1,1 we expect to obtain a rank value as predicted in Section 27.4 which is an approximate formula. This corresponds to the union of very simple local quadratic equations relating up to $2K'$ variables at once. Larger values of $K1, K2$ might give the same rank but typically they give better, higher rank with additional equations which involve more variables than $2K'$, up to $(K1 + K2)K'$ variables. Increasing these value will require more memory and will allow to find a complete system of equations faster, however in general the attack is exponential in these values so they should remain small.

26.1 Some Results for $M = 2$

First we are going to see if we can confirm by computer simulations that the ratio F/T in this attack improves when K' grows, which is the crucial property we wish to have [another lever in this attack is the K1/K2 values which allow to use equations of higher order J cf. Section 28.3]. Here are some results in which K is relatively small and K' grows.

simulations with 9390489 incremental simulator							
K value	4						
$K1$ value	3						
$K2$ value	3						
/maxorder value	2						
K' value	4	6	8	10	12	14	16
#vars	26	42	58	74	90	106	122
T_2	352	921	1712	2776	4096	5672	7504
$F_{2+2}^{J\leq 2}(K)$	36	188	468	908	1764	2876	4244
F	36	188	468	???	1764	2876	????
F/T2	0.10	0.21	0.27	0.33	0.43	0.51	0.57

Fig. 16. Simulations with quadratic equations with increasing K'

Here T_2 is a theoretical number of monomials for arbitrary [48] quadratic equations without squares. This is a theoretical value, as for smaller parameters many monomials are simply never [49] present. Quite surprisingly we claim that

[48] In this section we only considered monomials without squares, in our software the option allows /allowsquares one can specify otherwise.

[49] For example, some variables interact more. For example the variable called $x1$ in each $T()$ contraint will be more frequently used compared to a variable called $x2$ in our description of $T()$. The same happens with monomials: only a subset are present, especially for small K'. For larger K' the proportion of monomials used increases dramatically except that for any K' we have tried so far, all quadratic monomials were absent [their number is asymptotically negligible].

this makes no difference[50] for larger K'. Later we will show the ratio $F/T2$ increases inexorably and tends to 1. More precisely it is possible to see when K is fixed and K' grows, both the $F/T2$ and therefore[51] also F/T ratio tend to 1 and can become as close to 1 as we want, cf. Section 27.5 below.

Current simulations are incremental [52] and open-ended: if we stop too early we will get an incorrect result. This is an artefact of the current implementation method and could change in a future version[53] of our program. In order to estimate estimate at which moment to stop the computation we currently recommend to wait four times the time after the result stabilizes and after our prediction $F_{2+2}^{J\leq 2}(K)$ was met or exceeded. These simulations can take up to many days on one single CPU to complete.

27 Simple Solution $M = 2$ - Basic Analysis and Predicting the Outcomes

Now the smallest attack which will be meaningful will not be one with $K1 = K2 = 1$, as we want to consider equations where we find relations on one side such as studied in Section 23.7, but rather one with $K1 = K2 = 2$ or higher and an additional limitation /maxorder2 which does not allow equations which involve more than 2 different sets of K' points (for a general definition of what we mean by order here, we refer to Def. 29.0.1 page 96). With this limitation /maxorder2 being active, the exact values of K1 and K2 used are irrelevant[54].

Can we predict what happens inside our attack in this simple case and other cases? In the following pages we will look at various types of equations starting from very simple to more complex ones and show how to predict their number with good precision. We will classify the equations following so called J value, which is the number of basic x1 points out of $2K$ which exist on both sides in our attack above. Later we will call this number J equation order Def. 29.0.1. In addition we will distinguish variables from both sides, and thus for example we call equations of type $a + b$ equations of order $J = a + b$ which use up to aK' variables from a different constraints for P1, and up to bK' variables from b different constraints for P2.

[50] We expect from Section 27 that $\lim_{K'\to\infty} T/T_2 = 1$ and by the fact that our equations can be seen as union of various equations from Section of Section 23.6 and 23.7, and that both categories have $\lim_{K'\to\infty} F/T_2 = 1$ as in these respective sections we have already used and displayed a theoretical formula for T. In fact it is possible that lots of monomials are bound to be missing for any $K' \leq 9$ and then for $K' \geq 10$ nothing similar happens as far as we can see.

[51] It is easy to see that $F/T2 \leq F/T \leq 1$.

[52] We can observe that in one first period of time the number of monomials displayed grows and the ratio F/T remains low, and at a later stage the F/T ratio increases steadily.

[53] It is easy to see that one can explore all possible equations systematically a certain order and add new equations to an already reduced Gröbner basis.

[54] This except that they need to be both equal to at least 2 to exploit the full power of /maxorder2.

27.1 Equations of Type 1+0

These are the equations which occur inside each section of K' points and for 1 such set they are the union of MQ2 equations and the same as studied in Section 13.1. the number of linearly independent equations following Section 13.1 or Section 13.8 (where we had a different notation and twice as many variables) is exactly (for one set of K' points on one side, for example for $P1$):

$$FQ = 0.5K'^2 - 1.5K' + 1$$

We can test this with our simulator which is described later in more detail. The command line is:

```
ax64 9390489 256 K'  1 0 1 0x1B  /disableparallel
```

27.2 More Equations of Order $J = 1$ - Type $1 + 0$ inside $a + 0$

Let $1 \leq a \leq K$. The next question is what happens if we have K sets of K' variables on one side (all a are for P1 or all a are for P2). We have a different sets of K' variables sharing one variable $x1$ due to the construction of our attack. We have $aK' - (a - 1)$ variables as all $x1$ variables are the same/identified. For both sides we expect that we have at most $2a$ times the FQ equations:

$$F^{J=1}_{1+0}(a) \leq 2a \cdot FQ$$

The number could however be smaller because of the contraction of variables: some variables are identical. We have not however observed that. It appears that we always have:

$$F^{J=1}_{1+0}(a) = 2a \cdot FQ$$

These equations have $J = 1$ and we consider equations for either P1 or P2 and not for both[55]. The command line to test this is:

```
ax64 9390489 256 K'  a 1 0 0x1B  /disableparallel
```

27.3 Equations of Order $J = 2$ Inside $a + 0$

We expect the same equations with $J = 2$ for each pair of sets of K' points on one side (e.g. for P1). Their expected number is the formula of Section 23.7 minus half of $F^{J=1}_{1+0}(a)$ studied above. For example if $a = 2$ we expect (for one side):

$$F^{J=2}_{2+0}(2)/2 = 2K'^2 - 9K' + 11 - 2 \cdot FQ = K'^2 - 6K' + 9$$

More generally we would expect that:

$$F^{J=2}_{2+0}(a) = 2\binom{a}{2} \cdot F^{J=2}_{2+0}(2)/2 = a(a-1)(K'^2 - 6K' + 9)$$

[55] We could call these equations 1+0 for P1 and 0+1 for P2, however in fact we consider that they are the same an call or them equations of type 1+0 and they are displayed as such as by our software.

27.4 Equations of Order $J = 2$ of Type 1+1 Inside $a + a$

There is only one type of equations with $J = 2$ which we have not studied yet: a^2 copies of the equations of type 1+1, of Section 23.6 which connect any one set of K' variables for P1 with any other set of K' variables for $P2$. It appears that these are disjoint, they simply add and are all linearly independent in presence of all the previous equations. Therefore we expect to have the following number:

$$F_{1+1}^{J=2}(a) = \begin{cases} 0 & \text{when } \; K' \le 9 \\ a^2\left(2K'^2 - 15K' + 24 - 2 \cdot FQ\right) & \text{when } \; K' > 9 \end{cases}$$
$$= \begin{cases} 0 & \text{when } \; K' \le 9 \\ a^2\left(K'^2 - 12K' + 22\right) & \text{when } \; K' > 9 \end{cases}$$

27.5 Combined Analysis Of All Equations At Order $J = 2$

Now we should be able to predict the total number of equations of order up to 2 in our attack for any K, K'. We expect to obtain:

$$F_{2+2}^{J\le 2}(K) = F_{1+0}^{J=1}(K) + F_{2+0}^{J=2}(K) + F_{1+1}^{J=2}(K) =$$

We also expect this number $F_{K1+K2}^{J\le 2}(K)$ to be the same for any $K1 \ge 2$ and $K2 \ge 2$. Overall our predictor will be [adding our formulas]:

$$F_{2+2}^{J\le 2}(K) = \begin{cases} (K^2 - K)(K'^2 - 6K' + 9) + 2K(0.5K'^2 - 1.5K' + 1) & \text{when } \; K' \le 9 \\ 2K^2K'^2 - 18K^2K' + 31K^2 + 3KK' - 7K & \text{when } \; K' > 9 \end{cases}$$

In addition we have:

1. The number of variables is $2(1 + K(K' - 1))$.
2. The number of possible homogenous quadratic monomials is exactly
$$T_2 = 2(1 + K(K' - 1))(K(K' - 1))$$

27.6 Example: Prediction vs. Actual Result

We have developed a simple implementation of our attack. We would like to see if we really understand what is happening inside our attacks and if our predictions work well.

We consider the case $K' = 12$, $K = 4$.

We predict that we should obtain approximately:

$$F_{2+2}^{J\le 2}(12) = F_{1+0}^{J=1}(12) + F_{2+0}^{J=2}(12) + F_{1+1}^{J=2}(12) = 440 + 972 + 352 = 1764$$

We predicted 1764 equations for the case $K' = 12, K = 4$. Our simulation gives 1764 exactly.

The exact command line to run our simulator is as follows:

```
ax64 9390489 256 12  4 2 2  0x1B  /maxorder2 /disableparallel
```

28 What Is Achieved in Our Prototype Attack with $M = 2$

28.1 Asymptotic Analysis of Our Quadratic Equations with $J = 2$

In our attack we achieve a complete coding of point splitting in 2 points **with** constraints. In Fig. 16 and later more detailed Fig. 17 we see that we obtain typically approximately as many linearly independent equations as predicted with our prediction formulas.

We assume that K is a constant and K' is a variable [we work with one fixed elliptic curve, e.g. P-256]. The dominant term for the number of linearly independent equations is

$$F_{2+2}^{J\leq 2}(K) \approx 2K^2K'^2$$

We recall that:

$$T_2 = 2(1 + K(K' - 1))(K(K' - 1)) \approx 2K^2K'^2$$

It should be now clear that we have achieved our objective:

$$\lim_{K' \to \infty} F/T = 1.$$

Moreover we already achieve this objective with our least powerful /maxorder2 setting. Moreover one can easily generate additional new equations, cf. Sections 28.3 and 29 below. Experience shows [cf. below] that increasing $K1$ and $K2$ values produces yet better F/T ratios which in turn should decrease the K' value necessary for the attack.

What's New. This type of point splitting attack with dense highly connected coding and with a unique solution has never been done before for the problem of EC point decomposition.

28.2 Our Main Objective

An overall goal is of course is to achieve optimal solvability in terms of the memory and running time required for an attack splitting a point on an elliptic curve in $M = 2$ parts with our constraints. The main idea in this attack is that **when K' grows, the regularity degree of the systems of equations obtained should decrease**. Moreover we expect that there exists an optimal K' value, for which the time or/and memory complexity to solve these equations will be minimal. This value K' should not be increased more than necessary and our current intuition is that the optimal value is closer to 10 than to 100.

28.3 On Increasing K1/K2

Better results can also be obtained by increasing K1/K2 values or removing the current /maxorder2 limitation. This, first of all, increases the proportion of equations found in one single gaussian elimination in our incremental process.

More importantly, using higher $K1$ and $K2$ or a higher order limitation does in general generate lots of **additional linearly independent quadratic equations** which equations we study later in Section 29 and Fig. 17 page 97.

In general we expect that K1 and K2 are very small, and the attack is in general exponential in K1/K2, so we need to assume these are fairly small constants and the primary adjustment variable is K'.

If in later study it turns out that our attacks scale very badly and K1 and K2 needs to be increased a lot, this will mean that we have designed an attack which is exponential. At this moment we believe that that this attack [or variant of it] should NOT need very larger $K1/K2$ values and we expect that IF this is correct, this attack should be sub-exponential and possibly polynomial [if $K1$ and $K2$ are constant]. This will be studied in Section 29: we will see that the number of higher-order equations can be large and grow fast with K' even and already for $K1 = K2 = 2$.

28.4 Limitations And Further Computer Simulations

Current solution is quite slow. We also dispose of a faster multi-core software solution. Moreover in order to save time for other researchers, we are going to produce ready examples of equations generated by this method. We would like to encourage all specialists of efficient computer algebra software to try to solve our equations, and contribute to developing efficient software solutions for the last step of our attack.

28.5 Solutions For Larger M?

It remains an open problem if a family of systems of equations with $\lim_{K \to \infty} F/T = 1$ can at all be achieved also for a **full** solution for a larger M, so far it was only achieved[56] for $M = 2$.

However it seems at present moment that all the key problems of the past are essentially already solved and suddenly the problem of ECDL for modulo p curves does not look as hard as it seemed to until recently, cf. also [35].

[56] We have achieved all our objectives for $M = 2$ and simulations show that the number of linearly independent equations is higher than expected, cf. Section 26.

29 Study of Higher Order Equations in Our Attack

Until now we have imposed an artificial limitation /maxorder2 in our attack. In this section we study what happens if we remove this limitation, and we are in particular interested in the equations of higher order. We propose a suitable general definition of order.

Definition 29.0.1 (Equation Order). We define an order of an equation J as the smallest number of constraints $T()$ which will cover all the variables it uses.

Remark. The formulation is not exactly straightforward because the sets of variables used in our $T()$ constraints overlap slightly. For example variables $x1$ are the same in each half of K constraints, or rather we identify them because they are linearly dependent over $\mathbb{F}_p$ due to our constraint coding which follows the description of Section 23.3.

29.1 Another Comparison to ElimLin

Exactly the same notion of order applies in ElimLin algorithm in block cipher cryptanalysis. We just need to replace "the minimum number of distinct constraints" $T()$ by "the minimum number of distinct encryptions". Our experience[57] shows that equations of very high order $J > 20$ do exist for ElimLin for block cipher Simon and these equations ARE THOSE which are responsible for the fast growth observed in Fig. 1 page 17.

Comparison to Cube Attacks. Well-known cube attacks [48, 29, 18] can also be viewed as as high order equations with $J = 2^d$ being a power of 2 and d being the size of a maxterm. This shows that many particularly high-order equations are already used[58] a lot in cryptanalysis.

29.2 Generation of Higher Order Equations

We use the same software tool to generate and count higher order equations. We consider the same series of simulations as in Section 26.1 however now we will distinguish various types equations we obtain. Again, we expect some threshold behavior: that initially nothing of interest exists, then for a higher K' the number of interesting equations will emerge and potentially new phenomena come to life. The interesting question is the speed or growth rate at which this happens and to be able to predict the number of equations obtained.

[57] Cf. our paper [12]. We also provide a working implementation of the attack cf. [23, 10] and a separate software tool for deep inspection of equations generated, cf. [10, 11] which classifies the equations according to their J order value, and according to many other criteria.

[58] Our experience shows that sets of equations generated by larger instances of ElimLin attacks are infinitely richer than in cube attacks. This is because in ElimLin, J can take arbitrary integer values and arbitrary sums of variables from arbitrary subsets of different encryptions are allowed. Moreover, experience shows that a great variety of equations for different J values actually exist, cf. [12, 22, 23] and [11].

29.3 Computer Simulations on Higher Order Equations

simulations with 9390489 incremental simulator $K = 4$								
K value		4						
$K1$ value		3						
$K2$ value		3						
/maxorder value		6						
K' value	predictor	4	6	8	10	12	14	16
#vars	$2(1 + K(K' - 1))$	26	42	58	74	90	106	122
$F_{1+0}^{J=1}(K) =$	$2K(0.5K'^2 - 1.5K' + 1)$	24	80	168	288	440	624	840
order 1	$2K \cdot FQ$	24	80	168	288	440	624	840
$F_{2+0}^{J=2}(K) =$	$K(K-1)(K'^2-6K'+9)$	12	108	300	588	972	1452	2028
order $2 + 0$	$F_{2+0}^{J=2}(K)$	12	108	300	588	972	1452	2028
$F_{1+1}^{J=2}(K) =$	$K^2(K'^2 - 12K' + 22)$	–	–	–	32	352	800	1376
order $1 + 1$	$F_{1+1}^{J=2}(K)$	0	0	0	32	352	800	1376
order 2		12	108	300	606	1324	2252	3404
predicted $J \leq 2$	$\sum$	36	188	468	908	1764	2876	4244
actual ≤ 2		36	188	468	894	1764	2876	4244
$F_{2+1}^{J=3}(K) =$	$(2K^2/2-2K)(K' - 2)$	–	–	–	192	240	288	336
order $2 + 1$		0	0	0	192	240	288	336
order $3 + 0$		0	0	0	0	0	0	0
order 3		0	0	0	192	240	288	336
order $4 - 6$		0	0	24	0	0	0	0
F total order 1-6		36	188	492	1100	2004	3164	4580

Fig. 17. Predicted and actual numbers of equations of different types and orders in our full attack simulator run with $K1 = K2 = 2$ and $K = 4$.

It is worth noting that at this moment for $J \leq 2$ our predictions on the number of **linearly independent** quadratic equations obtained in our prototype attack are 100 % exact and are confirmed by computer simulations. Can we also predict the number of higher-order equations?

29.4 Equations of Order $J = 3$ of Type 2+1 Inside $K + K$

The next question is whether we can also predict the number of equations of order 3 in Fig. 17. We expect that they exist and they behave in a very similar way as equations of type 1+1 in Section 27.4 except that we need to be more careful than before to generate and remove linear combinations of equations of lower orders and to study only equations with $J = 3$.

We are looking at equations with $J = 3$ which connect any one set of K' variables for P1 with any two sets of K' variables for $P2$, or vice versa. As in Section 27.4 we conjecture that AFTER removing all lower order equations the remaining equations with $J = 3$ exactly are disjoint and they can simply be added together. As in Section 27.4 such non-trivial equations exist only for all K' values starting at a certain threshold. Then it turns out that their number for one single configuration of 2 sets of K' points on one side, and one set $(2+1)$ can be predicted with 100 % accuracy by the following formula:

$$\begin{cases} 0 & \text{when} \;\; K' \leq 9 \\ K'-2 & \text{when} \;\; K' > 9 \end{cases}$$

From here we expect to have the following number:

$$F_{2+1}^{J=3}(K) \stackrel{?}{=} \begin{cases} 0 & \text{when} \;\; K' \leq 9 \\ K^2(K-1)(K'-2) & \text{when} \;\; K' > 9 \end{cases}$$

Unhappily when these equations are combined with each other at a larger scale, and with other equations with $J \leq$ studied in the previous sections, linear dependencies occur[59]. Computer simulations shows that we obtain rather something like the following formula:

$$F_{2+1}^{J=3}(K) = \begin{cases} 0 & \text{when} \;\; K' \leq 9 \\ (2K^2-2K)(K'-2)/2 & \text{when} \;\; K' > 9 \end{cases}$$

29.5 Are Our Equations Solvable?

Is the system of equations in our attack efficiently solvable?

We have already achieved $F/T \to \infty$ however the gap grows. We may need some extra linearly independent equations and this what we obtain above with $J = 3$. It seems that one can predict their number.

Now if K is a constant and K' grows, we have $F_{2+1}^{J=3}(K) = \mathcal{O}(K')$ and not longer $\mathcal{O}(K'^2)$. However since we have already achieved $F/T \to \infty$ this could [or not] be sufficient to solve our problem. Probably the answer is yes, and the gap grows as $\mathcal{O}(K)$ which is actually what we expect from Section 27.5. Therefore it seems that our system of equations could be efficiently solvable and we should be able to solve the point splitting problem for $M = 2$ in polynomial time, this is if equations of order 3 are sufficient. If equations of order 3 are not sufficient, equations of order 4 or 5 might be and the attack could still be polynomial. Otherwise this attack could still be sub-exponential, cf. also Section 28.3. We will further develop this question in a future update of this paper.

[59] This similar as with MQ2 equations in Section 12.5.

Part VIII

Discussion and Analysis

30 The Challenges Ahead

30.1 Optimistic or Pessimistic?

The reader might think that our attacks are yet flawed and will not[60] work well. However, and in our option, the opposite is at least equally likely. Quite possibly this paper is an overkill. We have made our lives more difficult than necessary because we wanted to really design an attack for which we can convince ourselves that it should work, or at least for which we can point out at WHY exactly we think it should work through some sort of combinatorial explosion in terms of the number of equations which can be generated. Our attacks are also probably and possibly unnecessarily symmetric/homogenous/simple and also possibly too much of "highly connected" sort. All these possibly being not quite necessary. In the future more efficient and more technical solutions will be developed which will achieve just the right amount of "overdefined" and "highly-connected".

Additional possibilities for [possibly quite substantial] improvement are listed in Sections 30.5 and 30.6 below.

30.2 Two Key Open Problems

At this moment we see two major open problems:

1. Developing a really good attack for larger M and with $\lim_{K\to\infty} F/T = 1$, and also possibly working attacks without this property.
2. In general a major open problem is what kind of threshold for F/T ratio would make systems of equations efficiently solvable. This question is apparently not yet solved in traditional Gröbner bases theory which have focused more on smaller fields [1, 2], though we might be wrong about it.

For the first question we claim that it is probably possible and all the building blocks are probably already present here. With general-purpose techniques such as Thm. 17.2.1. we can probably do a lot better than currently and we will release additional attacks soon.

We expect that all "good" attacks will work at very low degree (2 or 3) and it will probably not need to use Semaev polynomials at all, working along the lines of results such as Thm. 17.2.1 which are essentially methods to **replace** rather than recode arbitrary S3 polynomials in the context of specific redundant variables representations [our ECC codes]. This sort of general methods which starts with arbitrary ECC sums [needed for large M] should replace the simple attack sketched in Section 24.1 above.

For the second question we will probably need help of other researchers. For as long as this problem is not solved, **we cannot even start making estimations of what is the complexity of breaking the ECDL problem** with our method.

[60] Until now there isn't enough evidence that our basic idea is that designing a dense and overdefined system of equations is a good idea in spite of the fact that it generates a lot more variables.

30.3 Future Attacks, Future Software, Testing and Experimentation

Our philosophy is to develop tools which work as a compiler and convert/recode arbitrary systems of equations which allows the attacker a lot of flexibility and to combine several methods which improve the solvability of equations, to code at high level rather than at a low level, and yet to optimize the attacks very precisely. Developing and testing such solutions will require a lot of work and we also need to develop a better theory.

We are quite confident about the current results and have done a fair amount of experimentation with these techniques already, more to come. We volunteer to provide to researchers, today or in a very near future, a working software solution which allows one to test every single claim made in this paper, first individually then combined. We believe that our claims are in some sense "efficiently falsifiable" [41].

30.4 On Predictions and Incertitudes

In particular we are confident about an ability of anyone to predict and confirm outcomes of experiments with near 100 % accuracy see [12], or we expect that many attacks can be made to work better than expected by adjusting some additional secondary parameters, cf. Section 27.6 and Section 28.3.

Accurate predictions are yet a rare thing in algebraic cryptanalysis. In this paper we are actually making a deliberate effort in this direction. We postulate that the attacker **can** design his methods and experiments in such a way that on a close examination there are extremely few incertitudes one can have, even if this makes his attacks slower and less efficient. One can always claim that some unknown exceptions to the rules may be discovered at any moment. However we believe that some attacks **more than other**, should allow to avoid situations where we will think that there is a chance that a property which is true for a small K would no longer be true for a larger K. For example, we believe that working at low degree and counting primarily on the degree falls which we have explicitly constructed ourselves, offers less complexity and far fewer incertitudes than with traditional Gröbner basis methods.

30.5 Improvements Due To Well Chosen Constants

A major problem in Algebraic Cryptanalysis is that rank of our systems of equations, or F/T will maybe not be as high as the attacker would like to. An interesting observation again is that in our attack methodology we use constants chosen by the attacker. A nice proof of concept where choosing such constants in a specific way CAN create NEW linearly independent equations which would NOT otherwise exist is given in Section 14.5. We can therefore really expect that some "good" choice of constants in attacks such as described in this paper can increase the rank of equations which is known to make such equations solvable more efficiently, cf. [6, 1, 2]. We should however note that the opposite can also happen, that for specific constants, some specific results could also fail or not work as claimed: some polynomials could be degenerated or reduce to 0 w.r.t. other polynomials.

30.6 Further Improvements with Particular Elliptic Curves

In Section 19 and Section 20 it is easy to see if we use our "Semi-Invariant" methodology, then using constants which are low order points on special elliptic curves will allow the attacker to generate A LOT more equations. The reason for that is that it creates a lot more situations where sum of two points in our set still belongs to our set, cf. for example Section 19.2 and therefore more equations which can be coded with results such as Thm. 17.1.1. This will increase the success of our attacks.

31 Conclusion

In this paper we propose new methods and a whole new philosophy to expand and solve algebraic equations derived from elliptic curve point decomposition problems in ordinary elliptic curves mod p such as used on the Internet and in real-life financial systems. This is motivated by the idea of building an index calculus algorithm for ECDL problem. Initially we worked with so called Semaev polynomials and progressively we discovered that there exist other new and substantially simpler low degree polynomial algebraic relations.

Another major contribution of this paper is to work on equations topology. Having worked in algebraic block cipher cryptanalysis for years we have learned that if our system of equations looks like one derived from a block cipher, chances that it will be solvable in practice are small[61]. Now we do NOT have to imitate block cipher cryptanalysis, we can probably do a lot better. In this paper we postulate that the attacker needs to work on **connection topology** in systems polynomial of equations and that one needs to imitate dense and highly connected graphs.

Our methods are very different than any previous approaches known in the literature. We share similar objectives as many earlier works in algebraic cryptanalysis which attempt to generate overdefined systems of equations with high R/T ratio, try to create and exploit degree falls. However we do not expand the number of monomials and their degree, we expand the number of variables, which is a major innovation, even though both types of methods are known to specialists in algebraic cryptanalysis of block ciphers, cf. Section 3.5.

Our approach is NOT to blindly hope that some equations might be efficiently solvable cf. [46, 45, 43, 42]. Our primary technique is to carefully engineer a system of equations which are in our opinion more likely to be efficiently solvable, than in any previous approach. We show that one can provoke a combinatorial explosion in the number of equations generated and achieve systems of equations with high rank close to saturation. Then, we will not just hope that some degree falls might occur which could help to solve our equations. We explicitly generate degree falls by closed formulas which allow one to efficiently generate on the fly the sort of algebraic shortcuts one may need and in large quantities.

We do not yet offer a mature solution for a larger M however it seems that all our objectives are or can be achieved. We construct systems of equations which are **massively overdefined** with R/T ratio being substantially higher than in any previous works [46, 45, 43, 42]. We propose a first general method for point splitting with M parts with R/T being a large constant, cf. Section 6.6 completed in Section 24.1. We have also designed and implemented a demonstrator attack with $M = 2$ which achieves

$$\lim_{K' \to \infty} F/T = 1$$

[61] Well actually it seems that the attacker can to some extent break any cipher just by increasing the parameter K and through combinatorial explosion / super-linear growth, cf. Section 4.3. We do not know a counter-example where algebraic attacks would not exhibit this incredible super-linear growth phenomenon.

To the best of our knowledge these things have never been done before in ECDL research. We conjecture that these could be sufficient to solve the point splitting problem in quasi-polynomial time for $M = 2$ and probably for any fixed M, which should lead to full index calculus solutions for ECDL problem. We have not yet achieved this objective. Current paper is essentially about methods to **transform** some polynomial equations or to replace them by other larger more overdefined and arguably more "efficiently solvable" polynomial equations. In order achieve to the desired properties we introduced some interesting innovations: specific forms of ECC Codes and new types of polynomials which are NOT contained in the ideal generated by simple S3 equations which we initially used to describe the problem, cf. Thm. 10.0.3.

Another important contribution in this paper is a major new "parallel" methodology for coding the contraints, a.k.a. the factor basis which follows closely our "densely connected" philosophy. It was specifically designed to make sure that variables at arbitrary locations are able to effectively interact with each other, even at a low degree. A major improvement compared to all previous attempts to solve this problem cf. [42] which had poorly connected topology.

It is quite possible that this paper found a way to solve or rather circumvent up to **all** the traditional difficulties which made that an index calculus algorithm was not yet developed for elliptic curves modulo p. Until now no researcher have ever been able to demonstrate in practice the feasibility of point splitting for any realistic elliptic curve. We will present improved solutions in subsequent revisions of this paper.

The reader would like of course to know if our bizarre and unorthodox approach works well. At this moment we are not aware of the existence of a mathematical or practical obstacle of any sort to claim that the methods described in this paper will not work to solve the ECDL problem. We do not claim that these methods are particularly efficient or particularly well optimized but rather that they are such that they are more likely to work than any previous approach for the ECDL problem, or that it is easier to believe that they will work. The price of adding a lot of new variables needs however to be paid and it could be that these methods are good asymptotically but not great in practice.

Acknowledgments: This research was inspired by numerous interesting discussions on the security of ECs I had in the last 3 years with Christophe Petit, Jean-Jacques Quisquater, Steven Galbraith, Arjen Lenstra, Antoine Joux, Louis Goubin, Daniel Augot, Jens Groth, Jonathan Bootle and Guangyan Song.

Special Thanks to UCL Students: The following UCL students have participated in this project through study of polynomial equations in various types of elliptic curves, programming and experimentation with SAGE algebra software, during activities organized as a part of GA18 course Cryptanalysis taught at UCL in 2014-2016. These students are Ilyas Azeem, Iason Papapanagiotakis-Bousy, Patrick Hough, Huanyu Ma, Lim Min, Wei Shao and Weixiu Tan and each of these students have done some work related to this project.

References

1. Magali Bardet, Jean-Charles Faugère and Bruno Salvy, *On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations,* in Proceedings of International Conference on Polynomial System Solving (ICPSS, Paris, France), pp. 71-75, 2004. Also known as Research report RR-5049, INRIA 2003.
2. Ludovic Perret: *Gröbner bases techniques in Cryptography,* `http://web.stevens.edu/algebraic/Files/SCPQ/SCPQ-2011-03-30-talk-Perret.pdf`
3. Jiun-Ming Chen, Nicolas Courtois and Bo-Yin Yang: *On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis,* In ICICS'04, LNCS 3269, pp. 401-413, Springer, 2004.
4. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*; Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281, 2001.
5. Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007.
6. Nicolas Courtois, Adi Shamir, Jacques Patarin, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, In Advances in Cryptology, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
7. Nicolas Courtois, Jacques Patarin: *About the XL Algorithm over* $GF(2)$, Cryptographers' Track RSA 2003, San Francisco, April 13-17 2003, LNCS 2612, pp. 141-157, Springer.
8. Nicolas T. Courtois: *How Fast can be Algebraic Attacks on Block Ciphers?* In online proceedings of Dagstuhl Seminar 07021, *Symmetric Cryptography* 07-12 January 2007, E. Biham, H. Handschuh, S. Lucks, V. Rijmen (Eds.), `http://drops.dagstuhl.de/portals/index.php?semnr=07021`, ISSN 1862 - 4405, 2007. Also available from `http://eprint.iacr.org/2006/168/`.
9. Nicolas Courtois *CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited* Available at `http://eprint.iacr.org/2007/152/`.
10. Nicolas Courtois: Some algebraic cryptanalysis software `http://www.cryptosystem.net/aes/tools.html`.
11. Guangyan Song and Nicolas Courtois: *Java tool for Deep Inspection of equations generated with ElimLin over GF(2) in Cryptalanalysis of Block Ciphers,* available at `http://www.nicolascourtois.com/software/DeepElimlin-1.4-SNAPSHOT.jar` documentation can be found in the appropriate section of this web page: `http://www.cryptosystem.net/aes/tools.html`.
12. Nicolas T. Courtois, Iason Papapanagiotakis-Bousy, Pouyan Sepehrdad and Guangyan Song: *Predicting Outcomes of ElimLin Attack on Lightweight Block Cipher Simon,* In Secrypt 2016 proceedings.
13. Nicolas Courtois and Blandine Debraize: *Specific S-box Criteria in Algebraic Attacks on Block Ciphers with Several Known Plaintexts,* In WEWoRC 2007, LNCS 4945, pp 100-113, Springer, 2008.
14. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* preprint, 2010-2014, available at `http://eprint.iacr.org/2011/626`.
15. Nicolas Courtois, Pouyan Sepherdad, Petr Susil and Serge Vaudenay: *ElimLin Algorithm Revisited,* In FSE 2012, LNCS, Springer.

16. Nicolas Courtois, Jerzy A. Gawinecki, Guangyan Song: *Contradiction Immunity and Guess-Then-Determine Attacks On GOST,* In Tatra Mountains Mathematic Publications, Vol. 53 no. 3 (2012), pp. 65-79. At `http://www.sav.sk/journals/uploads/0114113604CuGaSo.pdf`.
17. Nicolas Courtois: *On Splitting a Point with Summation Polynomials in Binary Elliptic Curves,* preprint 6 January 2015, `http://eprint.iacr.org/2016/003.pdf`.
18. Gregory Bard, Nicolas Courtois, Jorge Nakahara Jr, Pouyan Sepehrdad and Bingsheng Zhang *Algebraic, AIDA/Cube and Side Channel Analysis of KATAN Family of Block Ciphers* In Indocrypt 2010, December 12-15, 2010, Hyderabad, India, Springer LNCS.
19. Nicolas T. Courtois: *New Frontier in Symmetric Cryptanalysis,* Invited talk at Indocrypt 2008, 14-17 December 2008.
    Extended version of slides presented: `http://www.nicolascourtois.com/papers/front_indocrypt08.pdf`. A much shorter version was presented at the rump session of Eurocrypt 2007, `http://www.iacr.org/conferences/eurocrypt2007/slides/rumpt26.pdf`.
20. Nicolas Courtois and Blandine Debraize: *Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0.,* In ICICS 2008, 10th International Conference on Information and Communications Security, 20 - 22 October, 2008, Birmingham, UK. In LNCS 5308, pp. 328-344, Springer, 2008.
21. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4, LNCS 3373, pp. 67-83, Springer, 2005.
22. Nicolas Courtois: *Algebraic Attacks vs. Design of Block and Stream Ciphers,* slides used in GA18 course Cryptanalysis taught at University College London, 2014-2016, `http://www.nicolascourtois.com/papers/ algat_all_teach_2015.pdf`
23. Nicolas Courtois: *Software and Algebraic Cryptanalysis Lab, a lab used in GA18 course Cryptanalysis taught at University College London, 17 March 2016,* `http://www.nicolascourtois.com/papers/ga18/AC_Lab1_ElimLin_Simon_CTC2.pdf`
24. *Nicolas Courtois and Josef Pieprzyk: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
25. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Available at `http://eprint.iacr.org/2002/044/`. This preprint contains two different (earlier) versions of the XSL attack, see also [24].
26. Nicolas Courtois, Theodosis Mourouzis, Guangyan Song, Pouyan Sepehrdad and Petr Susil: *Combined Algebraic and Truncated Differential Cryptanalysis on Reduced-round Simon,* in post-proceedings of SECRYPT 2014, 28-30 August 2014, Vienna, Austria.
27. Petr Susil, Pouyan Sepehrdad, Serge Vaudenay, Nicolas Courtois: *On selection of samples in algebraic attacks and a new technique to find hidden low degree equations.* in International Journal of Information Security vol. 15 iss. 1, pp. 51-65, Springer, 2016.
28. Claus Diem: *On the discrete logarithm problem in elliptic curves,* In Compos. Math., 147(2011), pp. 75-104.
29. Itai Dinur, Adi Shamir: *Cube Attacks on Tweakable Black Box Polynomials,* In Eurocrypt 2009, LNCS 5479, pp. 278 - 299, Springer.
30. Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Gwenaël Renault: *Improving the complexity of index calculus algorithms in elliptic curves over binary fields,* In Eurocrypt 2012, LNCS 7237, pp. 27-44, Springer 2012.

31. Steven D. Galbraith, Pierrick Gaudry: *Recent progress on the elliptic curve discrete logarithm problem,* preprint, 22 Oct 2015, `https://eprint.iacr.org/2015/1022.pdf`
32. Darrel Hankerson, Alfred Menezes, Scott Vanstone: *Guide to Elliptic Curve Cryptography,* book, hardcover, 331 pages, Springer 2004.
33. T. Hodges, C. Petit, and J. Schlather: *First fall degree and Weil descent,* In Finite Fields Appl., 30(2014), pp. 155-177.
34. Koray Karabina: *Point Decomposition Problem in Binary Elliptic Curves,* at `https://eprint.iacr.org/2015/319`, 27 Oct 2015.
35. Neal Koblitz and Alfred J. Menezes *A Riddle Wrapped in an Enigma,* `https://eprint.iacr.org/2015/1018.pdf`
36. *Lorenz Minder: Cryptography based on error correcting codes,* PhD thesis 3846 (2007), EPFL, 27 July 2007, at `http://algo.epfl.ch/_media/en/projects/lorenz_thesis.pdf`.
37. Ming-Deh A. Huang, Michiel Kosters, Sze Ling Yeo: *Last Fall Degree, HFE, and Weil Descent Attacks on ECDLP,* In Crypto 2015, LNCS 9215, pp. 581-600, August 2015.
38. Michiel Kosters, Sze Ling Yeo: *Notes on summation polynomials,* revised 8 Jun 2015 `http://arxiv.org/abs/1503.08001`
39. Antoine Joux, Jean-Charles Faugère: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, LNCS 2729, pp. 44-60, Springer.
40. Victor Miller: *Use of elliptic curves in cryptography,* in proc. Crypto'85, pp. 417-426, LNCS 218, Springer, 1986.
41. Moni Naor: *On cryptographic assumptions and challenges,* In Crypto'03, LNCS 2729, pp. 96109, Springer, 2003.
42. Christophe Petit, Michiel Kosters, Ange Messeng: *Algebraic Approaches for the Elliptic Curve Discrete Logarithm Problem over Prime Fields,* In PKC 2016, vol. 2, LNCS 9615, pp. 3-18, Springer, 2016.
43. Christophe Petit and Jean-Jacques Quisquater: *On polynomial systems arising from a Weil descent,* In Asiacrypt 2012, LNCS 7658, pp. 451-466, Springer 2012.
44. Håvard Raddum: *Algebraic Analysis of the Simon Block Cipher Family,* In LatinCrypt 2015, LNCS 9230, pp. 157-169, Springer, 2015, cf. `https://www.simula.no/file/simonpaperrevisedpdf/download`.
45. Igor Semaev: *New algorithm for the discrete logarithm problem on elliptic curves,* Preprint, 10 April 2015, available at `eprint.iacr.org/2015/310/`.
46. Igor Semaev: *Summation polynomials and the discrete logarithm problem on elliptic curves,* Preprint, available at `eprint.iacr.org/2004/031/`.
47. M. Shantz and E. Teske: *Solving the elliptic curve discrete logarithm problem using Semaev polynomials, Weil descent and Gröobner basis methods - an experimental study,* In LNCS 8260, pp. 94-107, Springer 2013.
48. Michael Vielhaber: *Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack,* In `https://eprint.iacr.org/2007/413`.