A MULTISTEP STRATEGY FOR POLYNOMIAL SYSTEM SOLVING OVER FINITE FIELDS AND A NEW ALGEBRAIC ATTACK ON THE STREAM CIPHER TRIVIUM

ROBERTO LA SCALA*, FEDERICO PINTORE*, SHARWAN K. TIWARI**, AND ANDREA VISCONTI†

Abstract. In this paper we introduce a multistep generalization of the guess-and-determine or hybrid strategy for solving a system of multivariate polynomial equations over a finite field. In particular, we propose performing the exhaustive evaluation of a subset of variables stepwise, that is, by incrementing the size of such subset each time that an evaluation leads to a polynomial system which is possibly unfeasible to solve. The decision about which evaluation to extend is based on a preprocessing consisting in computing an incomplete Gröbner basis after the current evaluation, which possibly generates linear polynomials that are used to eliminate further variables. If the number of remaining variables in the system is deemed still too high, the evaluation is extended and the preprocessing is iterated. Otherwise, we solve the system by a Gröbner basis computation.

Having in mind cryptanalytic applications, we present an implementation of this strategy in an algorithm called MultiSolve which is designed for polynomial systems having at most one solution. We prove explicit formulas for its complexity which are based on probability distributions that can be easily estimated by performing the proposed preprocessing on a testset of evaluations for different subsets of variables. We prove that an optimal complexity of MultiSolve is achieved by using a full multistep strategy with a maximum number of steps and in turn the classical guess-and-determine strategy, which essentially is a strategy consisting of a single step, is the worst choice. Finally, we extensively study the behaviour of MultiSolve when performing an algebraic attack on the well-known stream cipher Trivium.

1. Introduction

One of the most challenging and useful tasks in Computational Algebra is solving a non-linear system of multivariate polynomial equations over a finite field. Applications ranges from Discrete Logarithm Problem [2] to SAT Problem [28], from the computation of Error Locator Polynomials [31] to the study of classical solutions of

2000 *Mathematics Subject Classification.* Primary 13P15. Secondary 11T71, 12H10.

Key words and phrases. Polynomial system solving; Finite fields; Cryptanalysis.

The first author acknowledges the partial support of PNRR MUR projects "Security and Rights in the CyberSpace", Grant ref. CUP H93C22000620001, Code PE00000014, Spoke 3, and "National Center for HPC, Big Data and Quantum Computing", Grant ref. CUP H93C22000450007, Code CN00000013, Spoke 10. The same author was co-funded by Università di Bari, Horizon Europe Seeds project, Grant ref. PANDORA - S22 and "Fondo acquisto e manutenzione attrezzature per la ricerca", Grant ref. DR 3191. The second author was funded by the "Research for Innovation Project" (POR Puglia FESR-FSE 2014/2024), Grant ref. 366ABF6C. The fourth author was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NextGenerationEU.

Quantum Designs [33]. The problem of solving a Multivariate Polynomial system over a finite field is called "MP problem" and it is known to be NP-hard (see, for instance, [19]). Therefore, determining the worst-case complexity of this problem has become of great relevance within Complexity Theory. Furthermore, the assumed difficulty of the MP problem has been exploited as the security backbone for numerous cryptographic schemes, motivating further research on polynomial system solving algorithms for both cryptanalytic and design purposes.

Among symbolic algorithms to solve the MP problem, Gröbner bases generally represent one the most effective options. The current best-known methods include Faugère's F_4, F_5 algorithms [15, 16], as well as Courtois et al. XL algorithm [12]. Both these approaches essentially rely on performing Gaussian elimination over a "Macaulay matrix" which is a natural way of represent polynomials as vectors with respect to a monomial basis. Even in accordance with the classical Buchberger's algorithm (see, for instance, [1]), the complexity of such methods is then affected by the size of the largest Macaulay matrix involved in the computations, which depends on a parameter known as the "solving degree" of the polynomial system. The difficulty in precisely determining this degree in advance has led to the introduction of the concept of "semi-regular sequences" [5] which have theoretically-computable solving degree. Alternative parameters to analyze the complexity of Gröbner bases algorithms have been also considered and we refer the interested reader to [9] for an overview on these parameters which form a still active area of research.

It should be mentioned that there are other effective symbolic methods available for the MP problem such as characteristic (triangular) sets [36] and involutive bases [20]. Among non-symbolic solvers for the binary field GF(2), we have well-developed and widely used algorithms such as SAT solvers, besides new promising methods as Quantum Annealing which are about to be fully exploited for the MP problem [34]. Nevertheless, providing accurate estimates for the complexity of solving a non-linear system over a finite field remains a problem largely open. In the present paper we propose a reliable statistical way to estimate this complexity when the polynomial system has at most one solution - a quite common case in cryptography.

When solving an instance of the MP problem, an upper bound for the complexity is clearly provided by the exhaustive evaluation of all variables. In real-world applications, this approach is generally unfeasible because of a large number of variables, as well as solving the given system by a single instance of any existing solver. A standard approach is therefore to try a "guess-and-determine" or "hybrid" strategy which consists in combining the exhaustive evaluation of a subset of variables with the solving of all resulting polynomial systems. This sort of divide-and-conquer strategy generally reduces an unfeasible MP problem to many feasible instances of it and hence its complexity is essentially the number of such instances. Different applications of this strategy in the cryptography context appear, for instance, in the papers [6, 12, 24, 26, 27]. In all these applications, one assumes that for a suitably large number of evaluated variables the corresponding systems can be all solved in a reasonable time. Let us call this standard strategy a "one-step strategy". Note that, in order to enhance the impact of the evaluations in reducing the number of variables, a kind of interreduction of the Gröbner bases theory is usually performed after the evaluation which can give rise to some linear polynomials that are used to eliminate further variables. This trick was introduced for the first time in cryptanalysis by Curtois et al. [11] as the procedure ElimLin. A main drawback

of a one-step strategy is that the number of required variables to provide that all polynomial systems corresponding to their evaluations are feasibly solvable may be quite large leading to a huge exponential number of solving instances.

In order to reduce the total number of polynomial systems to be solved, in this paper we propose a "multistep strategy" where one starts with a limited number of evaluated variables which is increased only if the number of remaining variables after performing an evaluation and an incomplete Gröbner basis computation stopped at some chosen degree D (including elimination via the linear polynomials obtained) is strictly greater than a given bound B. Such algorithm terminates because there is some final step where we have enough evaluated variables so that the number of remaining variables is always smaller or equal than B. The multistep strategy we propose is extremely flexible and optimizable by tuning the pararameters D and B and the considered set of evaluated variables. Moreover, we are able to prove that a multistep strategy with maximum number of steps, that is, where variables are added for evaluation one by one, is optimal with respect to the total number of systems to be solved. In other words, the standard one-step strategy is the worst one with respect to this number.

As a proof-of-concept, we have implemented an algebraic attack on the well-known stream cipher Trivium by means of the proposed multistep strategy. This cipher was proposed by De Cannière and Preneel in 2005 as a submission to the European project eSTREAM [13]. Indeed, Trivium was one of the winners of the competition for the category of hardware-oriented ciphers. The register of this cipher is a 288-bit string and the initial state consists of 80 bits as a private key (i.e. the secret seed), further 80 bits are the initial vector and the remaining bits are constant. Trivium is designed to generate up to 2^{64} bits of keystream where the first keystream bit outputs only after $4 \cdot 288 = 1152$ warm-up updates of the initial state. Despite its simple and elegant design, Trivium has so far brilliantly resisted all cryptanalytic attacks, none of which has a better complexity than brute force over the private key only, that is, 2^{80}.

Amongst the algebraic cryptanalysis of Trivium, one finds the attack of Raddum [32] which aims at recovering an internal state rather than the initial one by applying techniques from graph theory to solve a system of 954 multivariate polynomial equations in 954 variables, obtained from the knowledge of 288 keystream bits. The attack, further analysed in [7], has an estimated complexity of 2^{164}.

Another algebraic attack to recover an internal state was proposed in [24]. This attack exploits a standard one-step strategy which, after the evaluation of 115 variables, runs a kind of ElimLin procedure to further reduce the number of variables. The obtained systems for a testset of different keys and evaluations have at most 33 remaining variables and they are all solvable with a variant of the characteristic set method for finite fields, called MFCS [18]. In order to reduce the degree of the equations, this attack uses only 190 bits from the keystream allowing possible spurious solutions.

Among other kind of attacks, we mention the one by Maximov and Biryukov [30] which guesses the value of some specific state bits (or the products of state bits) leading, in some cases, to solve of a system of linear equations rather than a system of quadratic equations. The complexity of this attack is $\mathcal{O}(c \cdot 2^{83.5})$, where the constant c is $\mathcal{O}(2^{16.2})$. Note that for the attack to be successful, a rather prohibitive string of 2^{61} keystream bits is required.

Finally, in the cube attack proposed by Dinur and Shamir in [14], the adversary determines special polynomials, called superpolies, whose variables involve key bits. One computes superpolies by summing over a set of initial vectors which is called a "cube". In fact, in this attack the stream cipher is essentially treated as a black-box that grants the opponent access to IVs and corresponding keystreams. Many variations and improvements of such differential attack have targeted reduced variants of Trivium (see, for instance, [10] for an overview). By reduced we mean any modification of Trivium which considers either a reduced number N of warm-up updates (with the resulting scheme usually named N-round Trivium) or a smaller register with simplified update equations. Among the most recent results of this line of research is worth mentioning [23] where the superpolies are obtained for up to 848-round Trivium.

The present paper is organized as follows. In Section 2 we present basic facts and techniques for polynomial system solving over any finite field $\mathbb{F}$. In particular, we recall a polynomial bound for the solving degree of such polynomial systems and we derive a single exponential asymptotic complexity formula for the computation of a DegRevLex-Gröbner basis in terms of the number of variables and the cardinality of the field $\mathbb{F}$. Finally, we prove an elimination theorem for polynomial systems containing a set of explicit equations. In Section 3, we introduce the MultiSolve algorithm able to compute the $\mathbb{F}$-solution set $V_{\mathbb{F}}(J)$ of a polynomial ideal J such that $V_{\mathbb{F}}(J) \leq 1$. This algorithm implements the multistep strategy we propose and we discuss how its parameters can modify the behaviour of the algorithm. In Section 4 we study the complexity of the algorithm MultiSolve in terms of the number of calls to its core subroutines GBElimLin (generation of linear polynomials and elimination by means of them) and GrobnerBasis. These numbers are obtained by explicit formulas that include some probability distributions which can be reliably estimated by applying GBElimLin on suitably large testsets for different numbers of evaluated variables. We prove that a minimum number of calls to GrobnerBasis (or other polynomial system solvers) is obtained for a full multistep strategy, that is, a strategy which consists in adding a single variable to the evaluation set. In Section 5 we recall the notion of difference stream cipher [26, 27] as a suitable formalization of stream ciphers based on feedback shift registers and in Section 6 we discuss possible algebraic attacks on such ciphers. In Section 7 we describe the stream cipher Trivium as a difference stream cipher and we compute the inverse of its state transition map which allows an internal state attack. In Section 8 we present an extensive experimental study of the complexity of performing such an algebraic attack on Trivium by means of the algorithm MultiSolve. We obtain an average complexity of $2^{106.2}$ calls to GrobnerBasis solver which improves any other previous algebraic attack but which is still worse than brute force over the 80-bit private key only. However, recall that Trivium has also a 80-bit initial vector which, together with the private key, actually define a string of 160 unknown bits in the 288-bit register. Finally, some conclusions are drawn in Section 9.

2. Solving polynomial systems over finite fields

In this section we briefly review some basic results about solving a polynomial system with coefficients and solutions over a finite field, having a low number of such solutions. This situation is quite natural in algebraic attacks on cryptosystems

where the key is generally determined in a unique way by the given data. We start fixing some notations. Let $\mathbb{F} = \mathrm{GF}(q)$ be a finite field and consider the polynomial algebra $R = \mathbb{F}[x_1, \ldots, x_n]$ and the ideal $L = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle \subset R$. If $J \subset R$ is an ideal and $\bar{\mathbb{F}}$ is the algebraic closure of the field $\mathbb{F}$, we denote

$$V(J) = \{(a_1, \ldots, a_n) \in \bar{\mathbb{F}}^n \mid f(a_1, \ldots, a_n) = 0 \text{ for all } f \in J\}.$$

and $V_{\mathbb{F}}(J) = V(J) \cap \mathbb{F}^n$.

The Nullstellensatz over finite fields (see, for instance, [22]) implies immediately the following result.

Proposition 2.1. *Let $J \subset R$ be an ideal. We have that $V(L) = \mathbb{F}^n$ and $V_{\mathbb{F}}(J) = V(J + L)$ where $J + L$ is a radical ideal of R.*

Since $V(L) = \mathbb{F}^n$, the generators of the ideal L are called *field equations*. An immediate consequence of the above result and the Nullstellsatz is the following result which provides a method to solve a polynomial system with at most one solution over the base field $\mathbb{F}$.

Proposition 2.2. *Let $J \subset R$ be an ideal such that $\#V_{\mathbb{F}}(J) \leq 1$. Then, the (reduced) universal Gröbner basis G of the ideal $J + L$, that is, its Gröbner basis with respect to any monomial ordering of R is*

$$G = \begin{cases} \{x_1 - a_1, \ldots, x_n - a_n\} & \text{if } V_{\mathbb{F}}(J) = \{(a_1, \ldots, a_n)\}, \\ \{1\} & \text{otherwise.} \end{cases}$$

Indeed, as a consequence of Proposition 2.2, we can choose the most efficient monomial orderings as DegRevLex to solve a polynomial system with a single or no solutions. Moreover, when $V_{\mathbb{F}}(J)$ consists of few solutions, note that the cost for obtaining them is again essentially that of computing a DegRevLex-Gröbner basis of $J + L$. In fact, for computing $V_{\mathbb{F}}(J) = V(J + L)$ one needs to convert such a basis into a Lex-Gröbner basis by means of the FGLM-algorithm [17] which has complexity $\mathcal{O}(nk^3)$ where $k = \#V_{\mathbb{F}}(J) = \dim_{\mathbb{F}} R/(J + L)$. If the integer k is small, such complexity is dominated by the cost of computing the DegRevLex-Gröbner basis.

A precise estimation of such cost is generally difficult but it is known that the worst case is doubly exponential for fields of characteristic zero. If $\mathbb{F} = \mathrm{GF}(q)$ and hence $V_{\mathbb{F}}(J) = V(J + L)$ is a finite set, algorithms for computing DegRevLex-Gröbner bases using linear algebra have a complexity formula [4, 6] of the form

$$\mathcal{O}\left(\binom{n + d_s}{d_s}^{\omega}\right) \tag{1}$$

where $2 < \omega \leq 3$ is the linear algebra exponent, that is, $\mathcal{O}(n^\omega)$ is the complexity for solving a linear system in n variables and d_s is the *solving degree*, that is, the highest degree of the S-polynomials involved in a complete Gröbner basis computation.

Note that $\binom{n+d_s}{d_s}$ is the number of monomials in n variables of degree $\leq d_s$. It refers to the number of columns of the so-called "Macaulay matrices" where polynomials are viewed as vectors of their coefficients with respect to the monomial basis.

If d_s is constant with respect to the number of variables n, one obtains clearly a polynomial complexity. In general, the solving degree d_s is bounded by a linear function of n, as established in the following consequence of the Macaulay bound for finite fields (see, for instance, Theorem 11 in [9]).

Proposition 2.3. *Let $J = \langle f_1, \dots, f_m \rangle$ be an ideal of $R = \mathbb{F}[x_1, \dots, x_n]$ ($\mathbb{F} = \mathrm{GF}(q)$) and consider $L = \langle x_1^q - x_1, \dots, x_n^q - x_n \rangle \subset R$. Put $d = \max\{d_1, \dots, d_m, q\}$ where $d_i = \deg(f_i)$. Then, the solving degree d_s for computing a DegRevLex-Gröbner basis of $J + L$ is bounded as follows*

$$d_s \leq (n+1)(d-1) + 1. \tag{2}$$

By assuming that the generators of J are in normal form modulo L, it holds that $d \leq n(q-1)$ ($n \geq 2$) and one has also the following bound

$$d_s \leq n^2(q-1) + n(q-2). \tag{3}$$

We remark that, in many concrete instances, the above bounds are not at all tight. However, they are a fundamental tool in showing that DegRevLex-Gröbner bases algorithms over finite fields have a single-exponential complexity.

Proposition 2.4. *Let $n \geq \max\{2, q-1\}$. A (linear algebra) algorithm for computing a DegRevLex-Gröbner basis of an ideal $J + L \subset R$ satisfies the following complexity*

$$\mathcal{O}(k^{k\omega}) \text{ where } k = n^2 q. \tag{4}$$

Proof. We start by applying the Stirling's formula to the factorials occurring in the binomial complexity (1). Recall this formula is

$$\sqrt{2\pi n} \cdot e^{\frac{1}{12n+1}} \cdot (n/e)^n \leq n! \leq \sqrt{2\pi n} \cdot e^{\frac{1}{12n}} \cdot (n/e)^n.$$

By applying the above estimate also to $(n+d_s)!$ and $d_s!$, we obtain

$$\binom{n+d_s}{d_s} = \frac{(n+d_s)!}{n! d_s!} \leq \sqrt{\frac{n+d_s}{2\pi n d_s}} \cdot e^{\frac{1}{12(n+d_s)+1} - \frac{1}{12n+1} - \frac{1}{12d_s+1}} \cdot \frac{(n+d_s)^{n+d_s}}{n^n d_s^{d_s}}.$$

The first factor $((n+d_s)/(2\pi n d_s))^{1/2}$ is a non-increasing function bounded above by 1, while the second factor $e^{\frac{1}{12(n+d_s)} - \frac{1}{12n+1} - \frac{1}{12d_s+1}}$ is bounded above by e. Hence, we further obtain

$$\binom{n+d_s}{d_s} \leq e \frac{(n+d_s)^{n+d_s}}{n^n d_s^{d_s}} \leq e(n+d_s)^{n+d_s}.$$

Let us consider now the bound (3) for the solving degree. We obtain

$$n + d_s \leq n + n^2(q-1) + n(q-2) = (n^2+n)(q-1).$$

By observing that $(n^2+n)(q-1) \leq n^2 q$ when $n \geq q-1$, we conclude

$$\binom{n+d_s}{d_s} \leq e(n+d_s)^{n+d_s} \leq e(n^2 q)^{n^2 q}.$$

□

Note that better estimates of the solving degree d_s are used within cryptography when the generators of the ideal $J = \langle f_1, \dots, f_m \rangle$ are considered randomly generated semi-regular sequences (see, for instance, [4, 5]). By means of such estimates and the complexity (1) for a DegRevLex-Gröbner basis, an approximation of the computational effort for obtaining the solution set $V_{\mathbb{F}}(J)$ is provided and used to assess cryptosystem security.

In the following sections, we make use of the idea that having linear equations in a polynomial system, or more generally explicit equations, provides a method to eliminate variables which preserving solutions. Since solving complexity heavily

depends on the number of variables, such an elimination can significantly improve the search for solutions. The following results provide a formalization of this idea and its effectiveness. Till the end of this section, we assume that $\mathbb{F}$ is any base field.

Let $R = \mathbb{F}[x_1, \ldots, x_n], R' = \mathbb{F}[x_1, \ldots, x_m]$ be polynomial algebras with $n \geq m$. Let $f_1, \ldots, f_{n-m} \in R'$ and consider the algebra homomorphism $\psi : R \to R'$ such that

$$x_i \mapsto \begin{cases} x_i & \text{if } i \leq m, \\ f_{i-m} & \text{otherwise.} \end{cases}$$

We have clearly that $\operatorname{Ker} \psi = I = \langle x_{m+1} - f_1, \ldots, x_n - f_{n-m} \rangle$ and $\operatorname{Im} \psi = R'$. In other words, we have that the quotient algebra R/I is isomorphic to the subalgebra $R' \subset R$ by the mapping $f + I \mapsto \psi(f)$.

Proposition 2.5. *Let $I \subset J \subset R$ be an ideal. We have that the elimination ideal $J \cap R'$ is equal to $\psi(J)$.*

Proof. Since ψ is the identity map on R', we have that $J \cap R' = \psi(J \cap R') \subset \psi(J)$. Let $f \in J$ and consider $f' = \psi(f) \in \psi(J) \subset R'$. We want to prove that $f' \in J$. Since $f' \in R'$ we have that $\psi(f') = f' = \psi(f)$ and hence $f - f' \in I \subset J$. Because $f \in J$ we conclude that $f' \in J$ and therefore $\psi(J) \subset J \cap R'$. □

Consider the map $\varphi : \mathbb{F}^m \to \mathbb{F}^n$ such that

$$(a_1, \ldots, a_m) \mapsto (a_1, \ldots, a_m, f_1(a_1, \ldots, a_m), \ldots, f_{n-m}(a_1, \ldots, a_m)).$$

Note that φ is the injective polynomial map corresponding to the surjective algebra homomorphism ψ, that is, $\psi(g)(a_1, \ldots, a_m) = g(\varphi(a_1, \ldots, a_m))$ for all $g \in R$ and $(a_1, \ldots, a_m) \in \mathbb{F}^m$. Consider now the solution set $V(J) = \{(a_1, \ldots, a_n) \in \bar{\mathbb{F}}^n \mid f(a_1, \ldots, a_n) = 0 \text{ for all } f \in J\}$ where $\bar{\mathbb{F}}$ is the algebraic closure of the field $\mathbb{F}$.

Proposition 2.6. *Let $I \subset J \subset R$ be an ideal. We have that $V(J) = \varphi(V(J \cap R')) = \varphi(V(\psi(J)))$.*

Proof. Let $\alpha = (a_1, \ldots, a_n) \in V(J)$, that is, $f(\alpha) = 0$ for all $f \in J$. If $\alpha' = (a_1, \ldots, a_m)$, it is clear that $f(\alpha') = 0$ for all $f' \in J \cap R'$, that is, one has that $\alpha' \in V(J \cap R')$. Since $I = \langle x_{m+1} - f_1, \ldots, x_n - f_{n-m} \rangle \subset J$, it holds that $a_{m+1} = f_1(\alpha'), \ldots, a_n = f_{n-m}(\alpha')$ and we conclude that $V(J) \subset \varphi(V(J \cap R'))$.

Assume now that $\alpha' = (a_1, \ldots, a_m) \in V(J \cap R')$ and consider $\alpha = (a_1, \ldots, a_n) = \varphi(\alpha')$. Let $f \in J$ and denote $f' = \psi(f) \in \psi(J)$. By Proposition 2.5, we have that $f' \in J \cap R'$ and since $\psi(f') = f' = \psi(f)$ it holds that $f - f' \in I$. This implies that $f'(\alpha') = 0$ and $f = f' + g$ with $g \in I$. Since $\alpha = \varphi(\alpha')$ we have that $f(\alpha) = 0$. In fact, it holds that $g(\alpha) = g(\varphi(\alpha')) = \psi(g)(\alpha') = 0$ because $g \in I = \operatorname{Ker} \psi$. We conclude that $\varphi(V(J \cap R')) \subset V(J)$. Finally, because $J \cap R' = \psi(J)$ by Proposition 2.5, it holds that $V(J) = \varphi(V(J \cap R')) = \varphi(V(\psi(J)))$. □

Informally, the above proposition states that the solutions of a polynomial system containing some explicit equations $x_{m+i} = f_i$ can be all computed by extending the solutions of the polynomial system obtained by eliminating the variables x_{m+i}. Observe that if $\deg(f_i) > 1$, the advantage of eliminating variables from equations is partially reduced by the growth of their degrees which may affect the solving degree of the system. It is not the case when we eliminate via linear equations, which is therefore the preferable situation. We elaborate and explore this viewpoint in the next section.

3. Solving by a multistep strategy

The complexities in the previous section show that computing a Gröbner basis for a quite large number of variables may be a hard task. The amount of computation does not change significantly if we substitute Gröbner bases with other solvers like SAT, BDD, etc. (see, for instance, [26]). Indeed, it is well-know that the problem of solving multivariate polynomial systems over finite fields is an NP-hard one [19]. A useful technique consists therefore in substituting an unfeasible Gröbner basis computation with many feasible ones for polynomial systems obtained from the given one by evaluating a subset of variables over the base finite field. Of course, the exponential complexity is hidden here in such exhaustive evaluation. This approach is usually called a *guess-and-determine* or *hybrid strategy*, where the latter name is especially used for polynomial systems that are randomly generated [5, 6]. In this case, by evaluating some variables one essentially obtains again a random system whose complexity can be estimated in a theoretical way. The computation of the total complexity may show that in some cases the hybrid approach is faster than a single Gröbner basis computation [6].

In this section, we essentially introduce a variant of this technique where given polynomial systems are generally not random and hence the evaluation of a subset of their variables may lead to systems with different behaviours. In particular, the main property we analyze for the resulting systems is the number of linear equations that can be obtained by an incomplete Gröbner basis computation which is stopped at some chosen degree D. In fact, the presence of these linear equations allows the elimination of a further amount of variables, without increasing the degrees of the system equations, with respect to the set of evaluated variables. The number of remaining variables, NRV in short, in these systems gives us an idea of the complexity we will face in solving them completely. Given a bound $0 \leq B \leq n$, we decide to compute a complete Gröbner basis only if NRV $\leq B$. Otherwise, we evaluate some additional amount of variables and we compute again the NRV which is then compared to B. Proceeding in this way, we are able to solve the given polynomial system with Gröbner bases over a number of variables which is always bounded by B. Assuming that each of these Gröbner basis computations can be performed in a reasonable time, the complexity is essentially the number of them. We will show how to estimate in practice this number by means of explicit formulas and simple statistics and how to optimize it by choosing appropriate steps when adding new variables to evaluate. We call this approach a *multistep strategy* because of such steps and because Gröbner bases are also computed stepwise. After the above high-level description of the strategy, in the following we formally describe it.

Let $R = \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial algebra over the finite field $\mathbb{F} = \mathrm{GF}(q)$. Consider an ideal $J = \langle f_1, \ldots, f_m \rangle \subset R$ and assume that $\#V_{\mathbb{F}}(J) \leq 1$. Denote as usual $L = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ and put $H = \{f_i\} \cup \{x_i^q - x_i\}$ which is a generating set of the ideal $J + L$. Let $D > 0$ be an integer and denote by GrobnerBasis(H, D) the algorithm performing an incomplete Gröbner basis computation which is stopped when all S-polynomials of degree $\leq D$ have been consider at a current step. This algorithm corresponds to put a bound on the size of the Macaulay matrix when using linear algebra methods. If D is suitably large, that is, it is a solving degree we have that $G =$ GrobnerBasis(H, D) is a complete Gröbner basis of $J + L$. In the considered case that $\#V_{\mathbb{F}}(J) \leq 1$,

the complete Gröbner basis G contains only linear polynomials, namely $G = \{1\}$ or $G = \{x_1 - a_1, \ldots, x_n - a_n\}$ (see Proposition 2.2). Of course, computing the complete Gröbner basis is generally expensive because of the complexity (1) and to know linear polynomials belonging to the ideal $J + L$ is useful for eliminating variables, without increasing degrees, according to Proposition 2.6. It is reasonable therefore to compute $G = \text{GROBNERBASIS}(H, D)$, for a quite low degree D in order to have a fast computation, and use the linear polynomials that we possibly found in G to eliminate variables from the generators of $J + L$. If $G = \{1\}$ or $G = \{x_1 - a_1, \ldots, x_n - a_n\}$, no further computation is needed. Otherwise, such a preprocessing will generally ease the task of computing $V_{\mathbb{F}}(J)$. We formalize it as the Algorithm 3.1.

Algorithm 3.1 GBELIMLIN

Input: a generating set H of an ideal $J + L$ ($\#V_{\mathbb{F}}(J) \leq 1$) and an integer $D > 0$;

Output: either a Gröbner basis of $J + L$ or a generating set of an ideal obtained by eliminating some variables, via linear equations, from $J + L$.

$G :=$ GROBNERBASIS(H, D);
if $\max\{\deg(g) \mid g \in G\} \leq 1$ **then**
  return G;
end if;
$G_1 := \{g \in G \mid \deg(g) = 1\}$;
$G_2 := G \setminus G_1$;
$S := \{x_i^q - x_i \mid x_i \text{ occurs in } G\}$;
$G :=$ REDUCE$(G_2, G_1 \cup S)$;
$S := \{x_i^q - x_i \mid x_i \text{ occurs in } G\}$;
$G := G \cup S$;
return G;

In the algorithm GBELIMLIN we assume that the linear polynomials of the set G_1 are completely interreduced, that is, they are obtained in reduced echelon form in the Macaulay matrix. The procedure REDUCE is the complete reduction algorithm of Gröbner bases theory (see, for instance, [1]). If it holds that $\max\{\deg(g) \mid g \in G\} \leq 1$, namely $G = \{1\}$ or $G = \{x_1 - a_1, \ldots, x_n - a_n\}$, then GBELIMLIN provides immediately $V_{\mathbb{F}}(J) = V(J + L)$ because it happens that D is a solving degree. Otherwise, we have that the solution set of the ideal $J' + L' = \langle G \rangle$ ($G =$ GBELIMLIN(H, D)) is immediately related to $V(J + L)$ according to Proposition 2.6. In this case, we define NRV > 0 as the number of variables occurring in G.

Since NRV is generally less than the number of variables in H and the degrees in these generating sets remain the same, the task of computing the solution set $V_{\mathbb{F}}(J')$ is easier than computing $V_{\mathbb{F}}(J)$. This simplification is obtained at the price of performing GBELIMLIN which may be quite cheap whenever D is a low degree. This happens, for instance, when $D = \max\{\deg(g) \mid g \in H\}$ and q is a small integer. Note that in this case, the procedure GROBNERBASIS(H, D) is essentially a kind of interreduction of Gröbner bases theory and the procedure GBELIMLIN practically coincides with linear algebra based algorithm ELIMLIN [3, 11].

Fix now a bound $0 \leq B \leq n$ and assume that a complete Gröbner basis over a number of variables less or equal than B can be computed in a reasonable time. If the number of variables n is quite large as in polynomial systems arising in cryptography, we cannot immediately apply GBElimLin(H, D). Then, we add to the generating set H of $J + L$ a set of evaluations of some suitable amount of variables, say

$$E = \{x_1 - a_1, \ldots, x_k - a_k\} \ (a_i \in \mathbb{F})$$

and compute $G = \text{GBElimLin}(H \cup E, D)$. As a result of the (incomplete) Gröbner basis computation in GBElimLin, one obtains that the corresponding NRV is generally less than $n - k$. This suggests to moderate the number k which determines an exponential complexity q^k corresponding to the exhaustive evaluation of k variables over the field $\mathbb{F} = \text{GF}(q)$. If NRV $\leq B$, we proceed with the compution of GrobnerBasis(G). Otherwise, we extend the evaluation set E to

$$E' = \{x_1 - a_1, \ldots, x_l - a_l\} \ (k < l)$$

and compute $G' = \text{GBElimLin}(H \cup E', D)$. By iterating the above steps, the solution set $V_{\mathbb{F}}(J)$ ($\#V_{\mathbb{F}}(J) \leq 1$) is obtained by adding each time a small amount of variables to evaluate, till either all considered evaluations are exhausted or the output of GBElimLin or GrobnerBasis (NRV $\leq B$) is a Gröbner basis G such that $\max\{\deg(g) \mid g \in G\} = 1$. The Algorithm 3.3 implement this multistep strategy together with a suitable management of the exhaustive evaluation of different amounts of variables. A fundamental subroutine is the Algorithm 3.2.

Note that StepSolve may output a Gröbner basis $G \neq \{1\}$ of an ideal obtained by eliminating some variables from the input ideal $J + L$ by means of GBElimLin. By Proposition 2.6, one obtains the actual Gröbner basis $G' = \{x_1 - a_1, \ldots, x_n - a_n\}$ of $J + L$, or equivalently $V_{\mathbb{F}}(J) = \{(a_1, \ldots, a_n)\}$, simply by extending the coordinates in G via linear equations.

Moreover, the algorithm MultiSolve provides the intended output under the condition that after the call to StepSolve at the final step k_r, we have $A_2 = \emptyset$ when $A_1 = \texttt{wild} - \texttt{set}$. In fact, we are assuming that $1 \leq k_r \leq n$ is the least integer such that this holds. Of course, such an integer exists because for $k_r = n$ we have the evaluation of all variables.

We remark that MultiSolve generalizes the standard guess-and-determine (or hybrid) strategy in the case that we have a single iteration, namely for $r = 1$ and $k_r = k_1$. The only difference with the standard strategy is that MultiSolve compute the Gröbner bases for all evaluations $(a_1, \ldots, a_k) \in \mathbb{F}^k$ $(k = k_r = k_1)$ in two steps, first executing GBElimLin and then by GrobnerBasis. If NRV$(\leq B)$ is fairly smaller than $n - k$, this approach can reduce computing times, especially for Gröbner bases algorithms based on Macaulay matrices and linear algebra. Indeed, this improvement of the standard strategy already appeared in the algorithm ElimLin [3, 11]. In the next section we will show that the algorithm MultiSolve for $r > 1$ outperforms the standard strategy by reducing the total number of Gröbner bases required to compute the solution set $V_{\mathbb{F}}(J)$.

Observe that if $B = 0$ the algorithm MultiSolve computes the output only by applying GBElimLin, that is, by Gröbner bases computations which are stopped at degree D. This is possible because for a sufficiently large final step $k_r \leq n$, the solving degree of all obtained systems becomes lower or equal than D.

Algorithm 3.2 StepSolve

Input: a generating set H of an ideal $J+L$ ($\#V_{\mathbb{F}}(J) \leq 1$), a subset $W \subset \mathbb{F}^k$ ($0 \leq k \leq n$) and three integers $k < l \leq n, 0 \leq B \leq n, D > 0$;

Output: either a pair (`solution`, G) where `solution` is a character string and $G \neq \{1\}$ is a Gröbner basis of $J+L$ or a Gröbner basis of an ideal obtained by eliminating some variables from $J+L$, or the pair (`wild-set`, W') where `wild-set` is a character string and W' is a subset of $\mathbb{F}^l$.

if $W = \{\emptyset\}$ **then**
  $W := \mathbb{F}^l$;
else
  $W := W \times \mathbb{F}^{l-k}$;
end if;
$W' := \emptyset$;
for all $(a_1, \ldots, a_l) \in W$ **do**
  $E := \{x_1 - a_1, \ldots, x_l - a_l\}$;
  $G :=$ GBElimLin$(H \cup E, D)$;
  if $\max\{\deg(g) \mid g \in G\} = 1$ **then**
    return (`solution`, G);
  end if;
  if $G \neq \{1\}$ **then**
    NRV := number of variables in G;
    if NRV $\leq B$ **then**
      $G :=$ GrobnerBasis(G);
      if $\max\{\deg(g) \mid g \in G\} = 1$ **then**
        return (`solution`, G);
      end if;
    else
      $W' := W' \cup \{(a_1, \ldots, a_l)\}$;
    end if;
  end if;
end for;
return (`wild-set`, W');

Before applying MultiSolve, it would be helpful to run GBElimLin on the generating set H of the ideal $J+L$. Indeed, this is generally unfeasible if H has a large number of variables. Nevertheless, if there are given linear polynomials in H, a good practice consists in initially eliminating variables by means of these generators. We finally remark that beside the choice of the integers D and B, a fundamental issue in the optimization of the algorithm MultiSolve consists in the choice of the subsets

$$\{x_1, \ldots, x_{k_1}\} \subset \{x_1, \ldots, x_{k_2}\} \subset \ldots \subset \{x_1, \ldots, x_{k_r}\}.$$

In the next section, we will show that an optimal choice for the steps k_i is to put $k_{i-1} = k_i - 1$. The main issue is therefore to single out the subset $\{x_1, \ldots, x_{k_r}\}$ which can have the most favorable impact on the total running time of MultiSolve. We suggest that in the search for an optimal subset of variables to evaluate, one can use as objective function the average value of the NRV numbers

Algorithm 3.3 MultiSolve

Input: a generating set H of an ideal $J + L$ ($V_{\mathbb{F}}(J) \leq 1$), a sequence of integers $1 \leq k_1 < \ldots < k_r \leq n$ and two integers $0 \leq B \leq n, D > 0$;
Output: either a Gröbner basis of $J + L$ or a Gröbner basis of an ideal obtained by eliminating some variables, via linear equations, from the ideal $J + L$.

$W := \{\emptyset\}$;
for all $i \in \{1, \ldots, r\}$ **do**
  $(A_1, A_2) :=$ StepSolve(H, W, k_i, D, B);
  if $A_1 =$ `solution` **then**
    return A_2;
  end if;
  $W := A_2$;
end for;
return $\{1\}$;

obtained by GBElimLin for the evaluations of $\{x_1, \ldots, x_{k_r}\}$ on a suitably large testset $T \subset \mathbb{F}^{k_r}$. Of course, if the number of such subsets of the set of all variables is huge, such an optimization cannot be performed by brute force. In this case, we can possibly search among a restricted amount of subsets which appear to be good candidates due to the specific form of the given system. Devising more effective optimization strategies remains an open problem which we defer to future work.

4. MultiSolve complexity

In this section we analyze the complexity of the algorithm MultiSolve which essentially corresponds to the number of calls to the subroutines GBElimLin and GrobnerBasis. In fact, both these procedures can be executed in a reasonable time for suitable values of the parameters $D > 0$ and $0 \leq B \leq n$.

We call *k-guess* an evaluation of the first k variables of the polynomial algebra $R = \mathbb{F}[x_1, \ldots, x_n]$ ($\mathbb{F} = \mathrm{GF}(q)$), that is, to make a guess $(a_1, \ldots, a_k) \in \mathbb{F}^k$ means that we are computing modulo the ideal generated by the set

$$E = \{x_1 - a_1, \ldots, x_k - a_k\}.$$

A guess is called *wild* if the procedure GBElimLin$(H \cup E, D)$ (H is a generating set of the ideal $J + L$) outputs a set of generators G such that $\max\{\deg(g) \mid g \in G\} > 1$ and its corresponding NRV is strictly greater than B. Otherwise, we say that the guess is *tamed*. Note that the latter includes the case that GBElimLin outputs a complete Gröbner basis, that is, $\max\{\deg(g) \mid g \in G\} \leq 1$ because $V_{\mathbb{F}}(J) \leq 1$.

Denote $p_B(k)$ the probability of wild k-guesses in the set $\mathbb{F}^k$ of all k-guesses, that is, $p_B(k)q^k$ is the number of wild k-guesses that the algorithm GBElimLin determines in the set $\mathbb{F}^k$. In practice, such probability is estimated on some random testset $T \subset \mathbb{F}^k$ of reasonable large size.

The results about the complexity of MultiSolve we are going to present, are all based on the safe assumption that extending a tamed k-guess to an l-guess ($k < l$) we always obtain again a tamed l-guess. This can be explained because reducing the number of variables in an ideal by evaluation also reduces its solving degree and the incomplete Gröbner basis computed by GBElimLin approaches the

complete one which is full linear. A first consequence is that the map $k \mapsto p_B(k)$ is a decreasing function.

Proposition 4.1. *Let* $1 \leq k_1 \leq \ldots \leq k_r \leq n$. *One has that* $1 \geq p_B(k_1) \geq \ldots \geq p_B(k_r) \geq 0$ *where* $p_B(n) = 0$.

Proof. Let $1 \leq k \leq l \leq n$. The number of all l-guesses extending wild k-guesses is

$$p(k)q^k q^{l-k} = p(k)q^l.$$

Since restricting a wild l-guess we always obtain a wild k-guess for the assumption discussed above, we have that all wild l-guesses in the set $\mathbb{F}^l$ are obtained by extending some wild k-guess. This implies that $p(k)q^l \geq p(l)q^l$ and hence $p(k) \geq p(l)$. Notice that some extensions of a wild k-guess may become tamed l-guesses. In particular, it is clear that $p_B(n) = 0$ since we have the evaluation of all variables of R. □

Starting from now, we denote

$$k'' = \min\{1 \leq k \leq n \mid p_B(k) = 0\}.$$

Given a sequence of steps $1 \leq k_1 \leq \ldots \leq k_r = k''$, we want to compute the total number of guesses that are considered in the algorithm MultiSolve which coincides with the number of calls to the subroutine GBElimLin. We show that this number only depends on the probabilities $p_B(k_1) \geq \ldots \geq p_B(k_{r-1})$ that can be estimated on a testset without actually performing MultiSolve.

Proposition 4.2. *The number of guesses in the algorithm* MultiSolve *for the steps* $1 \leq k_1 \leq \ldots \leq k_r = k''$ *is*

$$C_1 = q^{k_1} + p_B(k_1)q^{k_2} + \ldots + p_B(k_{r-1})q^{k_r}.$$

Proof. To simplify notations, let $k < l < m < \ldots$ be the steps of the algorithm MultiSolve. In the first step k, we have to consider all k-guesses whose number is q^k. In the second step l, we have to extend to l-guesses all wild k-guesses whose number is $p_B(k)q^k$. The number of such extensions is therefore

$$p_B(k)q^k q^{l-k} = p_B(k)q^l.$$

For the third step m, in this subset of all l-guesses extending wild k-guesses we have now to consider wild l-guesses to be extended to m-guesses. Since by restricting a wild l-guess we always obtain a wild k-guess, this number is exactly the number of all wild l-guesses in the set $\mathbb{F}^l$ which is $p_B(l)q^l$. Then, the number of m-guesses which extends wild l-guesses (extending wild k-guesses) is

$$p_B(l)q^l q^{m-l} = p_B(l)q^m.$$

We deduce that up to the third step, the total number of guesses considered in the algorithm MultiSolve is

$$q^k + p_B(k)q^l + p_B(l)q^m.$$

Iterating for all steps of the algorithm, one obtains the complexity C_1. □

Let us consider now the number of times the subroutine GrobnerBasis is executed in the algorithm MultiSolve, which generally gives main contribution to its total running time. By definition, this number is the cardinality of tamed guesses considered in the algorithm.

Proposition 4.3. *The number of tamed guesses in the algorithm* MultiSolve *is*

$$C_2 = (1 - p_B(k_1))q^{k_1} + (p_B(k_1) - p_B(k_2))q^{k_2} + \dots \\ + (p_B(k_{r-2}) - p_B(k_{r-1}))q^{k_{r-1}} + p_B(k_{r-1})q^{k_r}.$$

Proof. As shown in the above results, the number of wild guesses in algorithm MultiSolve only depends on the probability sequence $1 \geq p_B(k_1) \geq \dots \geq p_B(k_{r-1}) \geq p_B(k_r) = 0$. This total number is

$$C = p_B(k_1)q^{k_1} + p_B(k_2)q^{k_2} + \dots + p_B(k_{r-1})q^{k_{r-1}}.$$

We conclude that the total number of tamed guesses is $C_2 = C_1 - C$. □

We can ask now if there is an optimal choice for the subset $\{k_1, \dots, k_r\} \subset \{1, \dots, n\}$ with respect to the main complexity formula C_2. Recall that in this formula the last step is fixed as $k_r = k'' = \min\{1 \leq k \leq n \mid p_B(k) = 0\}$. In other words, the step k'' is the least integer such that GBElimLin$(H \cup E, D)$ (H is a generating set of $J + L$ and $E = \{x_1 - a_1, \dots, x_{k''} - a_{k''}\}$) obtains NRV $\leq B$ for all guesses $(a_1, \dots, a_{k''}) \in \mathbb{F}^{k''}$. We have then the following result.

Theorem 4.4. *The minimum value of the complexity C_2 is obtained for $k_i = i$ $(1 \leq i \leq k'')$, that is, for the maximum number of steps.*

Proof. To prove the statement is true, it is sufficient to show that each time we add a new step between two steps or before all steps, the sum C_2 decreases. Let $1 \leq k < l < m \leq k''$ be three consecutive steps in the algorithm MultiSolve. We distinguish two cases.

Case 1. If all three steps occur and k is not the first one, the formula C_2 contains the following two summands

$$(p_B(k) - p_B(l))2^l + (p_B(l) - p_B(m))2^m.$$

Otherwise, if step l is missing, we have a single corresponding summand in C_2, namely

$$(p_B(k) - p_B(m))2^m.$$

Note that all remaining summands in C_2 are the same in these two subcases. Now, since

$$(p_B(k) - p_B(m))2^m = (p_B(k) - p_B(l))2^m + (p_B(l) - p_B(m))2^m$$

and $(p_B(k) - p_B(l))2^m \geq (p_B(k) - p_B(l))2^l$ because $p_B(k) \geq p_B(l)$ and $m > l$, we conclude that

$$(p_B(k) - p_B(m))2^m \geq (p_B(k) - p_B(l))2^l + (p_B(l) - p_B(m))2^m.$$

Case 2. Either the step k or l is the first one. If k is the first step, the formula C_2 contains the summands

$$(1 - p_B(k))2^k + (p_B(k) - p_B(l))2^l.$$

Otherwise, if the step l is the first one, that is, the step k is missing, we have a single corresponding summand in C_2, namely

$$(1 - p_B(l))2^l.$$

Similarly to case 1, one has that

$$(1 - p_B(l))2^l = (1 - p_B(k))2^l + (p_B(k) - p_B(l))2^l$$

where $(1-p_B(k))2^l \geq (1-p_B(k))2^k$ because $1 \geq p_B(k)$ and $l > k$. We conclude therefore that

$$(1-p_B(l))2^l \geq (1-p_B(k))2^k + (p_B(k)-p_B(l))2^l.$$

□

Note that the above result shows in particular that the standard guess-and-determine strategy which corresponds to apply MultiSolve for the single step k'' is actually the worst one with respect to complexity C_2. As for the first step, say k', the above result suggests that $k' = 1$ but in practice k' is the smallest integer such that GBElimLin$(H \cup E, D)$ ($E = \{x_1 - a_1, \ldots, x_{k'} - a_{k'}\}$) can be performed in a reasonable time for the given parameter $D > 0$ and for all $(a_1, \ldots, a_{k'}) \in \mathbb{F}^{k'}$. For polynomial systems with many variables, such as the ones arising in cryptography, the first step $1 \leq k' \leq n$ will be a not so small integer. We call *full multistep strategy* the one corresponding to the minimum value of C_2 which has the following complexity formulas

$$\begin{aligned} C_1 &= \sum_{k' \leq k \leq k''} p_B(k-1)q^k; \\ C_2 &= \sum_{k' \leq k \leq k''} (p_B(k-1) - p_B(k))q^k, \end{aligned} \tag{5}$$

where $p_B(k'') = 0$ and by convention $p_B(k'-1) = 1$.

Of course, each summand of the above complexity formulas should be multiplied by suitable average timings in order to obtain running times. Indeed, we have the following corresponding total running times

$$\begin{aligned} T_1 &= \sum_{k' \leq k \leq k''} \sigma(k-1,k)p_B(k-1)q^k; \\ T_2 &= \sum_{k' \leq k \leq k''} \tau(k-1,k)(p_B(k-1) - p_B(k))q^k, \end{aligned} \tag{6}$$

where $\sigma(k-1,k)$ is the average running time of GBElimLin for the extensions of wild $(k-1)$-guesses to k-guesses and $\tau(k-1,k)$ is the average running time of GrobnerBasis for the extensions of wild $(k-1)$-guesses to tamed k-guesses. Note that for the first step k' we have the timings $\sigma(k'-1,k')$ and $\tau(k'-1,k')$ that, by convention, correspond to all k'-guesses and all tamed k'-guesses, respectively. The total execution time of MultiSolve is therefore $T = T_1 + T_2$.

We remark that in our experimental studies, which we will detail in the next sections, we observe that the probabilities $p_B(k)$ ($k' \leq k \leq k''$) appear to be a reliable statistical data about a multivariate polynomial system. These data can be safely used therefore to estimate the complexity of solving a polynomial system by the multistep algorithm MultiSolve, as well to tune it to an optimal complexity by varying its main parameters, namely the integers D, B and the subset of k'' variables to evaluate. Note also that in the algorithm MultiSolve one can possibly substitute GrobnerBasis solver with any other polynomial system solver over finite fields, such as XL solvers, SAT solvers and so on. The only step where an (incomplete) Gröbner basis computation is essential in MultiSolve is in the subroutine GBElimLin where this is used to generate additional linear equations starting from variable evaluations. An alternative way may be to use the linear

algebra algorithm ElimLin [3, 11]. We also recall that the probabilities $p_B(k)$ and hence the multistep complexities C_1, C_2 are obtained by performing GBElimLin over some testset $T \subset \mathbb{F}^k$, for all $k' \leq k \leq k''$.

We finally notice that if $V_{\mathbb{F}}(J) \neq \emptyset$ and we can estimate that the solution is found on the average by means of a tamed (correct) l-guess with $l \leq k < k''$, we can essentially consider k as a final step for MultiSolve even if $p_B(k) \neq 0$. This generally implies a consistent reduction of the complexities C_1, C_2 in this average case. We may have such an estimation, for instance, when studying polynomial systems arising from the algebraic cryptanalysis of a cryptosystem by using a testset of many different keys.

To show that the algorithm MultiSolve can be practically used in the context of cryptography, in the next sections we introduce and attack the well-known stream cipher Trivium [13]. In particular, we present this cryptographic scheme and its cryptanalysis in the general framework provided by the notion of "difference stream cipher" that has been recently introduced in the papers [26, 27].

5. Stream ciphers

Stream ciphers are essentially practical realizations of pseudorandom functions (see, for instance, [25, 29]). At a high level, on input of two fixed-length random strings usually named *seed* and *initialisation vector*, a stream cipher produces a random-looking string of arbitrary length which is called the *keystream*. The characters in these strings are generally elements of a finite field and they are most commonly bits. To obtain a symmetric cipher, the keystream can be used to encrypt a stream of plaintexts or decrypt a stream of ciphertexts simply by addition or subtraction. In this case, the seed and the initialization vector are respectively the key and the nonce of the cipher. A stream cipher is deemed secure if it is indistinguishable from a proper random function.

Most of the known constructions for stream ciphers rely on the use of feedback shift registers - FSR, in short. An FSR is an array of memory cells, with each cell storing a single element of a given finite field $\mathbb{F} = \mathrm{GF}(q)$, which are updated by means of some function f. More precisely, within an update the values of the cells are shifted to the left while the right-most cell gets as a new value the image under the function f of the current values of a given subset of cells of the same register and possibly that of some other FSRs. For finite fields, the function f can be always converted into a multivariate polynomial and it is called therefore *update polynomial*. An FSR is called *linear* or *non-linear* is the polynomial f is linear or not, respectively.

A stream cipher $\mathcal{C}$ can be obtained by combining a few FSRs $\mathcal{F}_1, \ldots, \mathcal{F}_n$, their corresponding update polynomials $f_1, \ldots, f_n$ and a further multivariate polynomial g with coefficients in $\mathbb{F}$ which is called *keystream polynomial*. Indeed, the keystream of $\mathcal{C}$ is obtained by evaluating the polynomial g over the current values of the registers of the $\mathcal{F}_i$. In particular, the memory cells of the FSRs are initially filled with the elements of a seed, an initialisation vector and, possibly, a constant string. Then, the $\mathcal{F}_i$ are updated in parallel $(h+u)$-times, where h is the intended length of the keystream and u is the number of updates (including initialization) in the *warm-up stage*. This stage places some distance between the initialization of the FSRs and the output of the keystream in order to prevent attacks on the stream cipher that are based on the knowledge of some amount of elements in the keystream.

A natural language for describing stream ciphers obtained from FSRs is that of explicit difference equations, as first observed in [27] where such stream ciphers are called *difference stream ciphers*. For the sake of readability, we recall some notations before their formal definition (see also [21]).

Given a finite field $\mathbb{F} = \mathrm{GF}(q)$ and $n \in \mathbb{N}^* = \mathbb{N} \setminus \{0\}$, we denote by R the polynomial algebra $\mathbb{F}[X]$ in the infinite set of variables $X = \bigcup_{t\in\mathbb{N}}\{x_1(t), \ldots, x_n(t)\}$. The algebra endomorphism $\sigma : R \to R$ defined by putting $\sigma(x_i(t)) = x_i(t+1)$ $(1 \leq i \leq n, t \geq 0)$ is called the *shift map* of R. For any $r_1, \ldots, r_n \in \mathbb{N}$, the following subset of variables

$$\bar{X} = \{x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1))\}$$

defines a (finitely generated) subalgebra $\bar{R} = \mathbb{F}[\bar{X}] \subset R$.

Definition 5.1. *A* difference stream cipher $\mathcal{C}$ *is a system of (algebraic ordinary) explicit difference equations of the form*

$$\left\{\begin{array}{rcll} x_1(r_1+t) & = & \sigma^t(f_1), & \\ & \vdots & & (t \in \mathbb{N}) \\ x_n(r_n+t) & = & \sigma^t(f_n), & \end{array}\right. \tag{7}$$

together with a polynomial $g \in \bar{R}$, where $r_1, \ldots, r_n \in \mathbb{N}$ and $f_1, \ldots, f_n \in \bar{R}$.

An $\mathbb{F}$-*solution of the system* (7) is a solution of all its explicit difference equations, that is, an n-tuple $(a_1, \ldots, a_n)$ of maps from $\mathbb{N}$ to $\mathbb{F}$ such that, given $v(t) = (a_1(t), \ldots, a_1(r_1 - 1 + t), \ldots, a_n(t), \ldots, a_n(r_n - 1 + t))$, it holds that $a_i(r_i + t) = f_i(v(t))$, for every integer $t \geq 0$. We call $v(t)$ the *t-state* of $(a_1, \ldots, a_n)$ and in particular $v(0)$ is the *initial state*. The system (7) has a unique $\mathbb{F}$-solution once the initial state is fixed [27, Thm. 2.4].

Remark 5.2. *A difference stream cipher $\mathcal{C}$ defined by an explicit difference system* (7) *and a polynomial $g \in \bar{R}$ can be realized as a stream cipher obtained from a set $\mathcal{F}_1, \ldots, \mathcal{F}_n$ of FSRs having update polynomials $f_1, \ldots, f_n$ and keystream polynomial g. In particular, the variable $x_i(t)$ $(1 \leq i \leq n)$ corresponds to the symbolic value of the left-most cell of the register of $\mathcal{F}_i$ at the clock $t \geq 0$. If $(a_1, \ldots, a_n)$ is an $\mathbb{F}$-solution of* (7)*, its initial state is composed by a seed, an initialisation vector and, possibly, a constant string. If $v(t) \in \mathbb{F}^r$ $(r = r_1 + \ldots + r_n)$ is the t-state of $(a_1, \ldots, a_n)$, the function $b : \mathbb{N} \to \mathbb{F}$ such that $b(t) = g(v(t))$ for all $t \geq 0$, is called the* keystream of $(a_1, \ldots, a_n)$. *By design, the j-th bit of the keystream can be set equal to $b(u + j - 1)$, with u being the number of warm-up updates of the stream cipher. Vice versa, a stream cipher obtained from some FSRs can be formally described as a difference stream cipher.*

6. Algebraic attacks

As already mentioned, a secure stream cipher $\mathcal{C}$ can be used to construct a CPA-secure symmetric encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where CPA stands for Chosen Plaintext Attack (see [25]). In particular, the key-generation algorithm Gen takes in input a security parameter $\lambda \in \mathbb{N}$ and returns a random seed s as the key. The encryption algorithm Enc, on input of a seed s and a message m, samples a uniform initialisation vector IV and outputs the ciphertext $c = (\mathrm{IV}, m + \mathsf{str}(|m|))$, where $\mathsf{str}(|m|)$ denotes the keystream produced by $\mathcal{C}$ on input of s and IV which is truncated after $|m|$ elements (here m is a string of elements of the finite field $\mathbb{F}$

and $|m|$ denotes its length). Decryption is then performed by subtraction of the keystream $\mathsf{str}(|m|)$.

CPA-security is assessed by means of a security game where the adversary is granted access to an encryption oracle. Therefore, the adversary has access to a polynomial number (in the security parameter λ) of strings $(\mathrm{IV}, \mathsf{str}(|m|))$ with the seed s fixed and IV and m which vary. Such knowledge can then be used by the adversary to pursue a key-recovery attack where he attempts to deduce information about the secret seed s.

As a consequence, if $\mathcal{C}$ is a difference stream cipher, the adversary has access to some string $(b(u), \dots, b(u+h-1))$ where $h \in \mathbb{N}$ is polynomial in λ and b is the keystream of a given $\mathbb{F}$-solution $(a_1, \dots, a_n)$ of the system (7). This implies the knowledge of the generators of the ideal

$$J_{u,h} = \sum_{u \le t < u+h} \langle \sigma^t(g) - b(t) \rangle \subset R.$$

We denote by $V_{\mathbb{F}}(J_{u,h})$ the set of simultaneous $\mathbb{F}$-solutions of the generators of $J_{u,h}$. Given the subset $S = \{x_1(r_1) - f_1, \dots, x_n(r_n) - f_n\} \subset R$ and the ideal

$$I = \langle \sigma^t(f) \mid f \in S, t \ge 0 \} \subset R$$

we consider $V_{\mathbb{F}}(I + J_{u,h}) = V_{\mathbb{F}}(I) \cap V_{\mathbb{F}}(J_{u,h})$, where $V_{\mathbb{F}}(I)$ is the set of all $\mathbb{F}$-solutions of the difference system (7). Since the function b is the keystream of a given $\mathbb{F}$-solution $(a_1, \dots, a_n) \in V_{\mathbb{F}}(I)$, we have that $(a_1, \dots, a_n) \in V_{\mathbb{F}}(I + J_{u,h}) \neq \emptyset$. Indeed, for actual stream ciphers one has that $\#(V_{\mathbb{F}}(I + J_{u,h})) = 1$ for some sufficiently large $h \in \mathbb{N}$. In other words, there is a unique solution of (7) that is compatible with a sufficiently long keystream.

Denote by $\bar{V}_{\mathbb{F}}(I + J_{u,h}) \subset \mathbb{F}^r$ the set of the initial states of the $\mathbb{F}$-solutions $(a_1, \dots, a_n) \in V_{\mathbb{F}}(I + J_{u,h})$. The following result is essentially a consequence of Proposition 2.6.

Theorem 6.1. [27, Thm. 5.4] *Let $\bar{\mathrm{T}} : \bar{R} \to \bar{R}$ be the algebra endomorphism defined for all $1 \le i \le n$ as follows*

$$\begin{array}{rcl} x_i(0) & \mapsto & x_i(1), \\ & \vdots & \\ x_i(r_i - 2) & \mapsto & x_i(r_i - 1), \\ x_i(r_i - 1) & \mapsto & f_i. \end{array}$$

Moreover, define the ideal

$$J'_{u,h} = \sum_{u \le t < u+h} \langle \bar{\mathrm{T}}^t(g) - b(t) \rangle \subset \bar{R}.$$

Then, it holds that $\bar{V}_{\mathbb{F}}(I + J_{u,h}) = V_{\mathbb{F}}(J'_{u,h})$.

Under the reasonable assumption that for a sufficiently large number h of keystream elements there is a unique $\mathbb{F}$-solution $(a_1, \dots, a_n)$ compatible with them, any attack which aims at determining the initial state of $(a_1, \dots, a_n)$, that is, $V_{\mathbb{F}}(J'_{u,h})$ is said to be an *algebraic (key-recovery) attack*. Note that the case in which we have spurious solutions, that is, $\#V_{\mathbb{F}}(J'_{u,h}) > 1$ can be identified by means of a Gröbner basis computation, namely by the linear dimension

$$\#V_{\mathbb{F}}(J'_{u,h}) = \dim_{\mathbb{F}} \bar{R}/(J'_{u,h} + L)$$

where $L \subset \bar{R}$ is the field equation ideal corresponding to the polynomial algebra $\bar{R}$. Recall in fact that $V_{\mathbb{F}}(J'_{u,h}) = V(J'_{u,h} + L)$. To compute this solution set, one can use Gröbner bases according to Proposition 2.2 or any other kind of solver. However, if the update polynomials f_i are non-linear ones, the polynomial $\bar{\mathrm{T}}^t(g)$ may have a high degree for large values of the clock t. Since $t \geq u$ in the definition of $J'_{u,h}$, this happens whenever the number u of warm-up updates of the stream cipher $\mathcal{C}$ is large, as it is usually the case. High degrees surely affect the Gröbner basis computation for $J'_{u,h} + L$ and hence a long warm-up stage is a good security strategy also with respect to algebraic attacks.

Note that the initial state usually contains the seed, the initial vector and some constant elements as well, and therefore it would seem desirable to attack it since the actual unknown entries are less than the total length of the registers. However, for the special class of invertible difference stream ciphers, the initial state can be uniquely recovered by any internal state and hence it is more convenient to attack the state at clock u where the keystream starts to output. Indeed, in this case we can assume $u = 0$ and compute $V_{\mathbb{F}}(J'_{0,h})$ where the generators of $J'_{0,h}$ have much lower degrees than those of the generators of $J'_{u,h}$. Of course, in this case all the entries of the considered initial state are completely unknown.

Definition 6.2. *We call an explicit difference system* (7) invertible *if the algebra endomorphism* $\bar{\mathrm{T}}$ *defined in Theorem 6.1 is actually an automorphism. A difference stream cipher* $\mathcal{C}$ *defined by an explicit difference system is said* invertible *accordingly.*

Note that the algebra endomorphism $\bar{\mathrm{T}} : \bar{R} \to \bar{R}$ has a corresponding polynomial map $\mathrm{T} : \mathbb{F}^r \to \mathbb{F}^r$ $(r = r_1 + \ldots + r_n)$. If $v(t) \in \mathbb{F}^r$ is the t-state of a $\mathbb{K}$-solution $(a_1, \ldots, a_n)$ of (7) we have that $\mathrm{T}(v(t)) = v(t+1)$, for all clocks $t \geq 0$. We call $\bar{\mathrm{T}}$ the *state transition endomorphism* and T the *state transition map of the explicit difference system* (7). If $\bar{\mathrm{T}}$ is an automorphism, the map T is also invertible and we have that $\mathrm{T}^{-1}(v(t+1)) = v(t)$. This implies that for invertible difference stream ciphers, we can uniquely obtain the initial state $v(0)$ starting from any internal state $v(t) = \mathrm{T}^t(v(0))$.

To establish invertibility and compute the inverse of an algebra automorphism one has the following general result based on Gröbner bases theory. For a comprehensive reference about automorphisms of polynomial algebras we refer to the book [35].

Theorem 6.3. *Let* $\mathbb{K}$ *be any field and let* $X = \{x_1, \ldots, x_r\}, X' = \{x'_1, \ldots, x'_r\}$ *be two disjoint variable sets. Define the polynomial algebras* $P = \mathbb{K}[X], P' = \mathbb{K}[X']$ *and* $Q = \mathbb{K}[X \cup X']$*. Consider an algebra endomorphism* $\varphi : P \to P$ *such that* $x_1 \mapsto g_1, \ldots, x_r \mapsto g_r$ $(g_i \in P)$ *and the corresponding ideal* $J \subset Q$ *which is generated by the set* $\{x'_1 - g_1, \ldots, x'_r - g_r\}$*. Moreover, endow the polynomial algebra* Q *by a product monomial ordering such that* $X \succ X'$*. Then, the map* φ *is an automorphism of* P *if and only if the reduced Gröbner basis of* J *is of the kind* $\{x_1 - g'_1, \ldots, x_r - g'_r\}$ *where* $g'_i \in P'$*, for all* $1 \leq i \leq r$*. In this case, if* $\varphi' : P' \to P'$ *is the algebra endomorphism such that* $x'_1 \mapsto g'_1, \ldots, x'_r \mapsto g'_r$ *and* $\xi : P \to P'$ *is the isomorphism* $x_1 \mapsto x'_1, \ldots, x_r \mapsto x'_r$*, we have that* $\xi\, \varphi^{-1} = \varphi'\, \xi$*.*

The invertibility property is satisfied, for instance, by the stream cipher Trivium on which we will experiment our algorithm MultiSolve by performing an algebraic attack. Note that the notion of inverse system of an invertible explicit difference

system can be introduced in a natural way [27, Def. 3.6]. Moreover, invertible systems (having independent subsystems) can be used to define block ciphers [27, Def. 5.6].

7. Trivium

The stream cipher Trivium was proposed by De Cannière and Preneel in 2005. It was submitted to the eSTREAM competition and therein selected as part of the final portfolio. We recall that eSTREAM was an European project which aimed at identifying new secure and efficient stream ciphers. Despite a wide cryptanalytic effort, best known attacks against Trivium are still significantly slower than brute-force attacks. Trivium is a difference stream cipher whose corresponding explicit difference system consists only of three quadratic equations over the base field $\mathbb{F} = \mathrm{GF}(2)$, namely

$$\begin{cases} x(93+t) = z(t) + x(24+t) + z(45+t) + z(1+t)z(2+t), \\ y(84+t) = x(t) + y(6+t) + x(27+t) + x(1+t)x(2+t), \\ z(111+t) = y(t) + y(15+t) + z(24+t) + y(1+t)y(2+t). \end{cases} \quad (t \in \mathbb{N}) \tag{8}$$

Its keystream polynomial is the following homogeneous linear polynomial

$$g = x(0) + x(27) + y(0) + y(15) + z(0) + z(45). \tag{9}$$

Consequently, a t-state is a string of $288 = 93 + 84 + 111$ bits, for any $t \geq 0$. The number of warm-up updates is $u = 4 \cdot 288 = 1152$. Seeds (i.e. private keys) and initialisation vectors are 80-bit strings and together they form 160 bits of an initial state. The remaining 128 bits are constants.

If $\bar{R} = \mathbb{F}[x(0), \ldots, x(92), y(0), \ldots, y(83), z(0), \ldots, z(110)]$, the state transition endomorphism $\bar{\mathrm{T}} : \bar{R} \to \bar{R}$ of Trivium is therefore the following one

$$\begin{array}{l} x(0) \mapsto x(1), \ldots, x(91) \mapsto x(92), x(92) \mapsto z(0) + x(24) + z(45) + z(1)z(2), \\ y(0) \mapsto y(1), \ldots, y(82) \mapsto y(83), y(83) \mapsto x(0) + y(6) + x(27) + x(1)x(2), \\ z(0) \mapsto z(1), \ldots, z(109) \mapsto z(110), z(110) \mapsto y(0) + y(15) + z(24) + y(1)y(2). \end{array} \tag{10}$$

By means of Theorem 6.3 we obtain that $\bar{\mathrm{T}}$ is in fact an automorphism whose inverse $\bar{\mathrm{T}}^{-1} : \bar{R} \to \bar{R}$ is defined as

$$\begin{array}{l} x(92) \mapsto x(91), \ldots, x(1) \mapsto x(0), x(0) \mapsto y(5) + x(26) + y(83) + x(0)x(1), \\ y(83) \mapsto y(82), \ldots, y(1) \mapsto y(0), y(0) \mapsto y(14) + z(23) + z(110) + y(0)y(1), \\ z(110) \mapsto z(109), \ldots, z(1) \mapsto z(0), z(0) \mapsto x(23) + z(44) + x(92) + z(0)z(1). \end{array} \tag{11}$$

The invertibility of the stream cipher Trivium allows therefore an algebraic attack on its u-state that we have analyzed by means of the algorithm MultiSolve. In fact, the high number of variables, namely 288, makes impossible to solve the corresponding polynomial system by a single Gröbner basis computation. Our multistep strategy divides instead this solving task in many subproblems, whose number can be easily estimated, where the number of variables is bounded at will.

8. A multistep attack on Trivium

In this section we essentially introduce the reader to the practical use of the complexity analysis of MultiSolve contained in Section 4. We make use to this purpose of an algebraic attack on the stream cipher Trivium whose computational cost we are able to estimate. This cryptanalysis is an instance of an algebraic attack on an internal state of an invertible difference stream cipher as described in Section 6. Recall that for Trivium we consider the polynomial algebra

$\bar{R} = \mathbb{F}[x(0), \ldots, x(92), y(0), \ldots, y(83), z(0), \ldots, z(110)]$ ($\mathbb{F} = \mathrm{GF}(2)$) and the corresponding field equation ideal $L \subset \bar{R}$. We apply MultiSolve to the generating set H of the ideal $J'_{0,h} + L \subset \bar{R}$ where

$$J'_{0,h} = \sum_{0 \le t < h} \langle \bar{\mathrm{T}}^t(g) - b(t) \rangle \subset \bar{R}.$$

Here $\bar{\mathrm{T}} : \bar{R} \to \bar{R}$ is the state transition automorphism (10) of Trivium, $g \in \bar{R}$ is its keystream polynomial (9) and $b(t)$ ($0 \le t < u$) is the portion of the keystream that we assume to know for the attack. The goal of the attack is to compute $V_{\mathbb{F}}(J'_{0,h}) = V(J'_{0,h} + L)$, that is, the internal state of Trivium at the keystream initial clock (see Section 6).

In our attack the length of the keystream is fixed to $h = 240$ because this provides polynomials of quite low degree (≤ 5) in the generating set H and at most one solution in all polynomial systems we computed to analyze MultiSolve.

A first useful observation is that the generators $\bar{\mathrm{T}}^t(g) - b(t) \in H$ ($0 \le t \le 65$) are independent linear polynomials and hence we can eliminate immediately 66 variables by means of them. The remaining set of 222 variables we consider is

$$\bar{X}' = \{x(0), \ldots, x(92), y(0), \ldots, y(83), z(0), \ldots, z(44)\}.$$

This set is still too large to actually perform GBElimLin and hence we decide to fix a set of 116 variables to be stepwise evaluated, namely $V = V' \cup V''$ where

$$V' = \{x(2), y(2), z(2), x(5), y(5), z(5), x(8), y(8), z(8), x(11), y(11), z(11), x(14), y(14), z(14),$$
$$x(17), y(17), z(17), x(20), y(20), z(20), x(23), y(23), z(23), x(26), y(26), z(26), x(29), y(29), z(29),$$
$$x(32), y(32), z(32), x(35), y(35), z(35), x(38), y(38), z(38), x(41), y(41), z(41), x(44), y(44), z(44),$$
$$x(47), y(47), x(50), y(50), x(53), y(53), x(56), y(56), x(59), y(59), x(62), y(62), x(65), y(65), x(68),$$
$$y(68), x(71), y(71), x(74), y(74), x(77), y(77), x(80), y(80), x(83), x(86), x(89), x(92)\};$$

$$V'' = \{x(3), y(3), z(3), x(6), y(6), z(6), x(9), y(9), z(9), x(12), y(12), z(12), x(15), y(15), z(15),$$
$$x(18), y(18), z(18), x(21), y(21), z(21), x(24), y(24), z(24), x(27), y(27), z(27), x(30), y(30), z(30),$$
$$x(33), y(33), z(33), x(36), y(36), z(36), x(39), y(39), z(39), x(42), y(42), z(42), y(45)\}.$$

The set V is chosen by means of GBElimLin, as it has the lowest average value of NRV, for a testset of different keys and evaluations, compared to many other subsets of $\bar{X}'$ of the same size. Of course, we could not consider all 116-subsets of $\bar{X}'$ and hence our search was restricted to a large group of candidates.

In our multistep strategy we start with the evaluation of 106 variables which are obtained by deleting the last 10 variables from the set V'' and we proceed by evaluating one further variable at each step. We label these steps by the corresponding number of evaluated variables, say k, and so we fix $106 \le k \le 116$ in our experiments. We apply GBElimLin with parameter D set to the maximum input degree which is generally equal to 3. This provides quite low average running time for GBElimLin which is, for instance, 4.5 seconds for $k = 106$ in our experiments.

For the testing activities, we run our code on a server with the following hardware configuration:

- CPU: 2 x AMD EPYC(TM) 7742;
- Number of CPU Cores/Threads: 2 x 64 Cores/2 x 128 Threads;
- Maximum CPU frequency achievable: 3.4GHz;
- L3 cache: 256Mb;
- RAM: 2 TB.

On this server, we install a Linux distribution as operating system – Ubuntu 22.04 LTS – and MAGMA version 2.27, a software package designed to solve problems in algebra and number theory [8]. We set MAGMA to use 16 threads when performing parallel linear algebra. Our main testset consists of 2^{16} different tests, namely 2^{12} random guesses of variables for 2^4 keystreams obtained starting from a same number of random initial states. We call this set a *random testset*. We also use a testset of 2^{16} different correct guesses corresponding to 2^{16} random initial states and we call it a *correct testset*. Recall that an initial state is here the internal state of TRIVIUM at the initial clock of the keystream.

The complexity of the algorithm MULTISOLVE strictly depends on the parameter B which defines the tamed guesses, that is, the Gröbner bases that are actually computed. In our experiments, we fix $32 \leq B \leq 38$. One reason for this choice is that we found $p_{32}(116) = 0$, that is, $k'' = 116$ is the possible last step for $B = 32$ where 116 is the maximum step considered. Moreover, we have that NRV $\leq B = 38$ allows the computation of GROBNERBASIS by MAGMA in few minutes at most which is for us "a reasonable time" for quite large testsets. Indeed, the solving degree we generally have in our tests is at most 5. For a number of variables greater than 38, we also observed that some internal errors occur when performing GROBNERBASIS with the parameter "HFE". By using this parameter, MAGMA runs the algorithm F4 with parallel computations over dense Macaulay matrices which generally imply a significant speedup.

The main statistics for computing the complexity of MULTISOLVE are the probabilities $p_B(k)$ for B and k in the considered intervals. To this purpose, we actually ran two independent random testsets on our server finding no significative differences between the two statistics obtained. Recall that these testsets consist each of 2^{12} random guesses for 2^4 random initial states.

TABLE 1. Probabilities $p_B(k)$ estimated on our *random* testsets.

k/B	32	33	34	35	36	37	38
106	0.63153	0.61703	0.60867	0.58736	0.55925	0.50848	0.44923
107	0.61670	0.60675	0.58188	0.55359	0.50471	0.44638	0.38121
108	0.57581	0.54169	0.49049	0.43468	0.37077	0.31258	0.23341
109	0.49127	0.43814	0.38190	0.29349	0.23219	0.16003	0.10303
110	0.43263	0.37582	0.28784	0.22722	0.15845	0.10179	0.04910
111	0.28415	0.22218	0.14967	0.09698	0.04762	0.02165	0.00670
112	0.14362	0.08954	0.04715	0.01627	0.00650	0.00053	0
113	0.08838	0.04610	0.01549	0.00650	0.00053	0	0
114	0.01498	0.00725	0.00053	0	0	0	0
115	0.00043	0	0	0	0	0	0
116	0	0	0	0	0	0	0

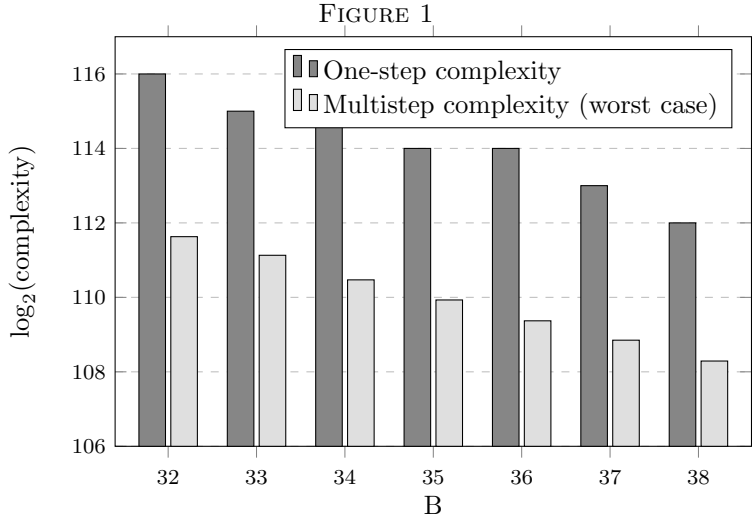
The Table 1 provide us with the possible final steps k'' for each value of B, in the worst case where the correct guess corresponding to the initial state becomes tamed at such steps.

Note that the above final steps gives us essentially the complexity $C = 2^{k''}$ of the one-step strategy, that is, the standard guess-and-determine or hybrid strategy. Assuming that the first step k' of the full multistep strategy is set $k' = 106$ for all

TABLE 2. Final steps k'' estimated on our *random* testsets (worst case).

B	32	33	34	35	36	37	38
k''	116	115	115	114	114	113	112

FIGURE 1



B, we can compare C with main complexity of the algorithm MULTISOLVE, namely

$$C_2 = \sum_{k' \leq k \leq k''} (p_B(k-1) - p_B(k))2^k.$$

Recall that by convention $p_B(k'-1) = 1$ and Theorem 4.4 implies that $C > C_2$. The Table 3 and Figure 1 provide a comparison between the one-step and the multistep strategy for our algebraic attack on TRIVIUM.

TABLE 3. One-step complexity VS Multistep complexity (worst case).

B	32	33	34	35	36	37	38
$\log_2(C)$	116	115	115	114	114	113	112
$\log_2(C_2)$	111.63	111.13	110.47	109.93	109.37	108.85	108.29

To obtain the complexity of MULTISOLVE in the average case, we need a second statistics, namely the probabilities $p_B(k)$ in the case of the correct testset. In order to prevent confusion, let us denote by $\bar{p}_B(k)$ the probabilities obtained in this way. For the following statistics, we make use of two independent testsets of 2^{16} different initial states each.

We use the above statistics to determine at which step $\bar{k}''$ one has probability $\bar{p}_B(\bar{k}'') < 0.5$, that is, at least half of the correct guesses become tamed in a step $l \leq \bar{k}'' < k''$. In other words, this provides that with probability greater than or equal to 0.5, the algorithm MULTISOLVE will stop at a final step $l \leq \bar{k}''$.

By considering k'' and $\bar{k}''$ as last steps for the algorithm MULTISOLVE, the following Table 6 and Figure 2 provide a comparison between the worst case complexity

TABLE 4. Probabilities $\bar{p}_B(k)$ estimated on our *correct* testsets.

k/B	32	33	34	35	36	37	38
106	0.98145	0.96414	0.93700	0.89612	0.83978	0.76630	0.67615
107	0.96077	0.93288	0.89172	0.83521	0.76161	0.67128	0.56934
108	0.88625	0.82880	0.75348	0.66212	0.55914	0.44754	0.33824
109	0.75659	0.66649	0.56439	0.45430	0.34366	0.24200	0.15723
110	0.66171	0.56049	0.45132	0.34123	0.24028	0.15625	0.08948
111	0.44080	0.33250	0.23412	0.15137	0.08672	0.04370	0.01810
112	0.23505	0.15071	0.08466	0.04102	0.01611	0.00458	0.00078
113	0.14989	0.08432	0.04094	0.01608	0.00458	0.00078	0
114	0.03961	0.01570	0.00450	0.00075	0	0	0
115	0.00381	0.00056	0	0	0	0	0
116	0.00056	0	0	0	0	0	0

TABLE 5. Final steps $\bar{k}''$ estimated on our *correct* testsets (average case).

B	32	33	34	35	36	37	38
$\bar{k}''$	111	111	110	109	109	108	108

C_2 and the average case complexity

$$\bar{C}_2 = \sum_{k' \le k \le \bar{k}''} (p_B(k-1) - p_B(k)) 2^k.$$

TABLE 6. Multistep complexity (worst case) VS Multistep complexity (average case).

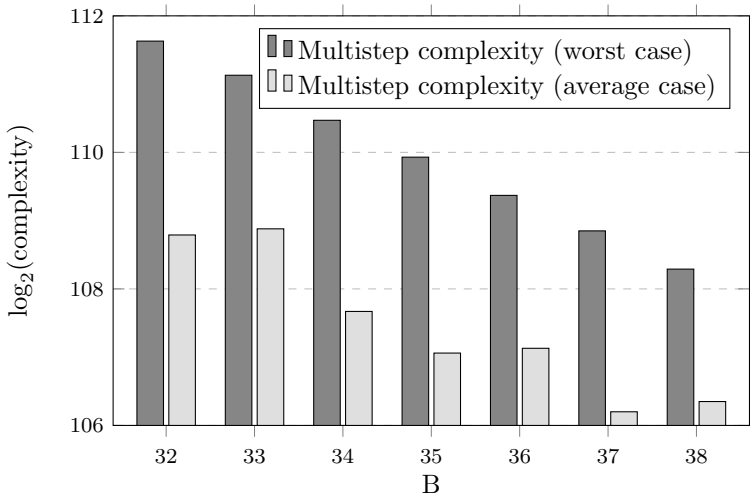
B	32	33	34	35	36	37	38
$\log_2(C_2)$	111.63	111.13	110.47	109.93	109.37	108.85	108.29
$\log_2(\bar{C}_2)$	108.79	108.88	107.67	107.06	107.13	106.20	106.35

We emphasize that the logarithm of our best complexity, namely $\log_2(\bar{C}_2) = 106.2$ for $B = 37$, is very close to the minimum number $k' = 106$ of evaluated variables that we have chosen for our optimization. This happens with $p_{37}(106) = 0.5$, that is, with half of the Gröbner bases in the step 106 that are not computed because NRV > 37. Using our code in MAGMA running on our server, for this average case complexity we estimate that an algebraic attack on TRIVIUM by MULTISOLVE takes approximately 2^{112} seconds. This estimation is obtained as $T = T_1 + T_2$ (see formulas (6)) where the running time of the Gröbner bases only is $T_2 = 2^{111.5}$ seconds.

9. CONCLUSIONS AND FURTHER DIRECTIONS

In this paper we have shown that the multistep strategy is, in theory and in practice, a viable way to estimate and improve the complexity of solving polynomial systems over finite fields. Although the neat stream cipher TRIVIUM remains secure after our algebraic attack based on this solving strategy, we obtain a relevant reduction to $2^{106.2}$ of the required complexity with respect to previous attacks of

FIGURE 2



the same kind. In fact, the experimentation of the proposed method is only at the beginning and we believe that an extended optimization of the many parameters of the multistep strategy together with possible advances in solving algorithms will make this technique even more effective as a tool to evaluate the security level of a cryptographic system. Of course, the multistep strategy is a general algorithm that could also be useful outside the cryptographic context.

As a further direction in this line of research, we suggest that the limitation of the number of remaining variables can be generalized to bounds involving other critical indicators that a polynomial system could be difficult to solve, such as the degree of its equations, their density, a theoretical value of the solving degree and so on. In general, we believe that the concept of statistical evaluation of the complexity of a polynomial system over a finite field is promising and requires further investigation.

10. Acknowledgements

First, we would like to thank Lorenzo D'Ambrosio for fruitful discussions and suggestions. Moreover, we gratefully acknowledge the use of the HPC resources of ReCaS-Bari (University of Bari, INFN).

References

[1] Adams, W.W.; Loustaunau, P., An introduction to Gröbner bases. Graduate Studies in Mathematics, 3. American Mathematical Society, Providence, RI, 1994.
[2] Amadori, A.; Pintore, F.; Sala, M., On the discrete logarithm problem for prime-field elliptic curves. Finite Fields Appl., 51 (2018), 168–182.
[3] Bard, G.V., Algebraic cryptanalysis. Springer, Dordrecht, 2009.
[4] Bardet, M.; Faugère, J.-C.; Salvy, B., On the complexity of the F5 Gröbner basis algorithm. J. Symbolic Comput. 70 (2015), 49–70.
[5] Bardet, M.; Faugère, J.-C.; Salvy, B.; Spaenlehauer, P.-J., On the complexity of solving quadratic Boolean systems. J. Complexity 29 (2013), no. 1, 53–75.
[6] Bettale, L.; Faugère, J.-C.; Perret, L., Hybrid approach for solving multivariate systems over finite fields. J. Math. Cryptol. 3 (2009), no. 3, 177–197.

[7] Borghoff, J.; Knudsen, L.R.; Matusiewicz, K., Hill Climbing Algorithms and Trivium. Selected Areas in Cryptography. SAC 2010 (2011), Lecture Notes in Comput. Sci., 6544. Springer
[8] Bosma, W.; Cannon, J.J.; Fieker, C.; Steel, A. (eds.), Handbook of Magma functions, Edition 2.27 (2022), 6359 pages.
[9] Caminata, A.; Gorla E., Solving Multivariate Polynomial Systems and an Invariant from Commutative Algebra. Arithmetic of Finite Fields (2021), 3–36, Lecture Notes in Comput. Sci., 12542. Springer, Cham.
[10] Cianfriglia, M.; Onofri, E.; Onofri, S.; Pedicini, M., Ten years of cube attacks. Cryptology ePrint Archive, 2022/137.
[11] Courtois, N.; Bard, G.V., Algebraic cryptanalysis of the data encryption standard. Cryptography and Coding, 152–169. Lectures Notes in Comput. Sci., 4887, Springer, Berlin, 2007.
[12] Courtois, N.; Klimov, A.; Patarin, J.; Shamir, A., Efficient algorithms for solving overdefined systems of multivariate polynomial equations. Advances in cryptology - EUROCRYPT 2000 (Bruges), 392–407, Lecture Notes in Comput. Sci., 1807, Springer, Berlin, 2000.
[13] De Cannière, C.; Preneel, B., Trivium Specifications. eSTREAM - ECRYPT Stream Cipher Project, https://www.ecrypt.eu.org/stream/triviumpf.html.
[14] Dinur, I.; Shamir, A., Cube Attacks on Tweakable Black Box Polynomials. Advances in cryptology - EUROCRYPT 2009, 278-299, Lecture Notes in Comput. Sci., 5479, Springer, Berlin, 2009.
[15] Faugère, J.C., A new efficient algorithm for computing Gröbner bases (F4). Effective methods in algebraic geometry (Saint-Malo, 1998). J. Pure Appl. Algebra 139 (1999), no. 1-3, 61–88.
[16] Faugère, J.C., A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, 75–83, ACM, New York, 2002.
[17] Faugère, J. C.; Gianni, P.; Lazard, D.; Mora, T., Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symbolic Comput. 16 (1993), no. 4, 329–344.
[18] Gao, X.-S.; Huang, Z., Characteristic set algorithms for equation solving in finite fields. J. Symbolic Comp. 47 (2012), no. 6, 655–679.
[19] Garey, M.-R.; Johnson, D.-S., Computers and intractability: a guide to the theory of NP-completeness. Freeman, 1979.
[20] Gerdt, V.P.; Blinkov, Y.A., Involutive bases of polynomial ideals. Simplification of systems of algebraic and differential equations with applications. Math. Comput. Simulation 45 (1998), no. 5-6, 519–541.
[21] Gerdt, V.; La Scala, R., Noetherian quotients of the algebra of partial difference polynomials and Gröbner bases of symmetric ideals. J. Algebra 423 (2015), 1233–1261.
[22] Ghorpade, S.R., A note on Nullstellensatz over finite fields. Contributions in Algebra and Algebraic Geometry, 23–32, Contemp. Math., 738, Amer. Math. Soc., Providence, RI, 2019.
[23] He, J.; Hu, K.; Preneel, B.; Wang, M., Stretching Cube Attacks: Improved Methods to Recover Massive Superpolies. Advances in Cryptology – ASIACRYPT 2022. Lecture Notes in Comput. Sci., 13794. Springer, Cham, 2022.
[24] Huang, Z.; Lin, D., Attacking Bivium and Trivium with the characteristic set method. Progress in cryptology - AFRICACRYPT 2011, 77–91, Lecture Notes in Comput. Sci., 6737, Springer, Heidelberg, 2011.
[25] Katz, J.; Lindell, Y., Introduction to modern cryptography. Second edition. Chapman & Hall/CRC Cryptography and Network Security. CRC press, Boca Raton, FL, 2015.
[26] La Scala, R.; Polese S.; Tiwari S.K.; Visconti A., An algebraic attack to the Bluetooth stream cipher E0. Finite Fields Appl. 84 (2022), Paper No. 102102, 29 pp.
[27] La Scala, R.; Tiwari S.K., Stream/block ciphers, difference equations and algebraic attacks. J. Symbolic Comput. 109 (2022), 177–198.
[28] Marek, V.W., Introduction to mathematics of satisfiability, Chapman & Hall/CRC Studies in Informatics Series. CRC Press, Boca Raton, FL, 2009.
[29] Mascia, C.; Piccione, E.; Sala, M., An algebraic attack on stream ciphers with application to nonlinear filter generators and WG-PRNG. arXiv:2112.12268, 2022, 16 pp., to appear in Adv. Math. Commun.
[30] Maximov, A.; Biryukov, A., Two Trivial Attacks on Trivium. Selected Areas in Cryptography. SAC 2007. Lecture Notes in Comput. Sci., 4876. Springer, 2007.

[31] Orsini, E.; Sala, M., Correcting errors and erasures via the syndrome variety. J. Pure Appl. Algebra 200 (2005), no. 1-2, 191–226.
[32] Raddum, H., Cryptanalytic Results on Trivium. eSTREAM - ECRYPT Stream Cipher Project, Report 2006/039 (2006).
[33] Rajchel-Mieldzioć, G., Quantum mappings and designs, PhD thesis, Centrum Fizyki Teoretycznej Polskiej Akademii Nauk, 2022.
[34] Ramos-Calderer, S.; Bravo-Prieto, C.; Lin, R.; Bellini, E.; Manzano, M., Aaraj, N., Latorre, J.I., Solving systems of Boolean multivariate equations with quantum annealing, Physical Review Research 4(1), 2022.
[35] van den Essen, A., Polynomial automorphisms and the Jacobian conjecture. Progress in Mathematics, 190. Birkhäuser Verlag, Basel, 2000.
[36] Wu, W.T., On the Decision Problem and the Mechanization of Theorem-Proving in Elementary Geometry, Sci. Sinica 21 (1978), no. 2, 159–172. Also in: Automated theorem proving (Denver, Col., 1983), 213–234, Contemp. Math., 29, Amer. Math. Soc., Providence, RI, 1984.

* Dipartimento di Matematica, Università degli Studi di Bari "Aldo Moro", Via Orabona 4, 70125 Bari, Italy
Email address: roberto.lascala,federico.pintore@uniba.it

** Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, United Arab Emirates
Email address: sharwan.tiwari@tii.ae

† CLUB – Cryptography and Coding Theory Group, Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, 20133 Milano, Italy
Email address: andrea.visconti@unimi.it