arXiv:2304.07820v2 [cs.SC] 5 Jun 2024

有限域上多项式系统求解的多步策略，以及针对流密码 Trivium 的一种新型代数攻击

Roberto La Scala*、Federico Pintore**、Sharwan K. Tiwari† 和 Andrea Visconti‡

Abstract. 本文提出一种多步推广，将猜测-确定策略或混合策略用于求解有限域上的多元多项式方程组。具体而言，我们提出分步对变量的一个子集进行穷举求值：每当一次赋值产生的多项式系统可能难以求解时，就增加该子集的大小。关于扩展哪种赋值的决策，基于预处理：在当前赋值后计算一个不完全 Gröbner 基，这可能会生成用于进一步消去变量的线性多项式。如果认为方程组中的剩余变量数量仍然过多，就扩展赋值并重复预处理。否则，我们通过计算完整 Gröbner 基来求解方程组。

考虑到密码分析应用，我们介绍了这种策略的一种实现，算法名为 MultiSolve，专为至多有一个解的多项式系统设计。我们证明了该算法复杂度的显式公式，这些公式基于概率分布；通过对不同变量子集的求值测试集执行所提出的预处理，即可轻松估算这些概率分布。我们证明，采用最大步骤数的完整多步策略可实现 MultiSolve 的最优复杂度；而标准的猜测并确定策略本质上只包含一个步骤，是最差的选择。最后，我们深入研究了 MultiSolve 在对著名流密码 Trivium 实施代数攻击时的表现。

谨以此纪念 Maria Jos ´e。

2000 *Mathematics Subject Classification.* 主要分类号：13P15。次要分类号：11T71、12H10。
Key words and phrases. 多项式系统求解；有限域；密码分析。第一作者感谢 PNRR MUR 项目“网络空间安全与权利”（资助编号：CUP H93C22000620001，代码 PE00000014，工作组 3）以及“国家高性能计算、大数据与量子计算中心”（资助编号：CUP H93C22000450007，代码 CN00000013，工作组 10）的部分资助。该作者还获得了巴里大学 Horizon Europe Seeds 项目（资助编号：P ANDORA - S22）以及“研究设备购置与维护基金”（资助编号：DR 3191）的共同资助。第二作者获得了“创新研究项目”（普利亚大区POR FESR-FSE 2014/2024，资助编号：366ABF6C）和 Ripple大学区块链研究计划的部分资助。第四作者获得了 SERICS 项目（PE00000014）的部分资助。该项目属于由欧盟下一代欧盟计划资助的 NRRP MUR 计划。

1. Introduction

计算代数中最具挑战性且最有用的任务之一，是求解有限域上的多元多项式非线性方程组。其应用范围广泛，从离散对数问题 [3] 到 SAT问题 [33], ，从计算错误定位多项式 [36] 到研究量子设计的经典解 [38]。有限域上的多元多项式方程组求解问题称为“MP问题”，已知该问题是 NP-困难的（例如，参见 [23]）。因此，确定这一问题的最坏情况复杂度已成为复杂性理论中的重要课题。此外，MP问题被认为具有的求解难度已成为众多密码方案的安全基石，这也推动了人们进一步研究用于密码分析和设计的多项式方程组求解算法。

在用于求解 MP问题的符号算法中，Gr¨obner 基通常是最有效的方法之一。目前已知的最佳方法包括 Faug`ère 的 F_4, F_5 算法 [18, 19], 以及 Courtois 等人的 XL 算法 [14]。这两种方法本质上都依赖于对 “Macaulay 矩阵 ” 进行高斯消元；该矩阵是将多项式表示为关于单项式基的向量的一种自然方式。即使采用经典的 Buchberger算法（例如参见 [2]），此类方法的复杂度也会受到计算中涉及的最大 Macaulay 矩阵规模的影响，而该规模取决于一个称为多项式系统“求解次数”的参数（例如参见 [11]）。由于难以预先精确确定这一次数，人们引入了“半正则序列”的概念 [6, 5] ，其求解次数在理论上可以计算。人们也考虑了用于分析 Gr¨obner 基算法复杂度的其他参数；关于这些参数的综述，可参见 [11] 。这些参数仍是一个活跃的研究领域。

值得一提的是，还有其他用于求解 MP问题的有效符号方法，例如特征（三角）集 [41] 和对合基[24]。在二元域GF(2)的非符号求解器中，除了 SAT 求解器等成熟且广泛使用的算法外，还有量子退火这类颇具前景的新方法，它们即将被充分应用于 MP问题[39]。然而，如何准确估计有限域上非线性方程组的求解复杂度，仍是一个悬而未决的问题。本文提出了一种可靠的统计方法，用于估计多项式系统至多只有一个解时的复杂度——这在密码学中相当常见。

求解一个 MP问题实例时，穷举所有变量的赋值显然给出了复杂度的上界。在实际应用中，由于变量数量庞大，这种方法通常不可行；同样，使用任何现有求解器单独求解给定方程组也往往不可行。因此，标准做法是尝试“猜测并确定”或“混合”策略，即将对一部分变量进行穷举求值与求解由此产生的所有多项式系统结合起来。这种分治策略通常能将一个不可行的 MP问题转化为许多个可行的实例，因此，其复杂度本质上取决于此类实例的数量。在密码学领域中，这种策略的不同应用见于例如论文 [7, 14, 29, 31, 32]。在所有这些应用中，人们都假设，对于数量足够多的已赋值变量，相应的方程组都能在

合理的时间内求解。我们将这种标准策略称为“单步策略”。请注意，为了增强求值在减少变量数量方面的效果，通常会在求值后执行 Gröbner 基理论中的某种互约化，这可能会产生一些线性多项式，用于进一步消去变量。Courtois 等人 [13] 首次在密码分析中引入了这一技巧，将其作为过程 ElimLin。单步策略的一个主要缺点是，为了使所有与这些变量取值相对应的多项式系统都能在可行时间内求解，所需变量的数量可能相当庞大，从而导致需要求解的实例数量呈指数级增长。

为了减少需要求解的多项式系统总数，本文提出了一种“多步策略”：首先从数量有限的已赋值变量开始，只有当执行一次赋值并进行在某个选定次数 D 处停止的不完全 Gr¨obner 基计算（包括利用所获得的线性多项式进行消元）后，剩余变量的数量严格大于给定界限 B 时，才增加已赋值变量的数量。该算法一定会终止，因为存在最后一个步骤，此时我们拥有足够多的已赋值变量，使得剩余变量的数量始终小于或等于 B。我们提出的多步策略极其灵活，并且可以通过调整参数 D 和 B 以及所考虑的已赋值变量集合进行优化。此外，我们能够证明，具有最大步骤数的多步策略，即逐个增加用于赋值的变量的策略，就需要求解的系统总数而言是最优的。换言之，标准的单步策略在系统数量这一指标上是最差的。

作为概念验证，我们通过所提出的多步策略，对知名流密码 Trivium 实施了代数攻击。该密码由 De Canni`ere 和 Preneel 于 2005年提出，作为参选方案提交给欧洲项目 eSTREAM [15]。事实上， Trivium 是面向硬件的密码类别中的获胜方案之一。该密码的寄存器是一个 288 比特的字符串，初始状态由 80 比特的私钥（即秘密种子）、另外 80 比特的初始向量以及其余的常量比特组成。 Trivium 旨在生成最多 2^{64} 比特的密钥流，而第一个密钥流比特只有在对初始状态进行 $4 \cdot 288 = 1152$次预热更新后才会产生。尽管设计简洁优雅， Trivium 迄今仍成功抵御了所有密码分析攻击；这些攻击的复杂度均不低于仅针对私钥进行暴力破解，即 2^{80}。

在针对 Trivium 的代数密码分析中，有 Raddum[37] 发起的攻击，其目标是通过应用图论技术，利用已知的 288 比特密钥流，求解由此得到的包含 954 个多元多项式方程和 954 个变量的方程组，从而恢复内部状态而非初始状态。该攻击的复杂度估计为 2^{164} ，并在 [9] 中通过使用模拟退火进行了进一步分析。

文献 [29] 提出了一种用于恢复内部状态的另一种代数攻击。该攻击利用一种标准的单步策略：对于 115 个变量的每次赋值，运行一种 ElimLin过程，以进一步减少变量的数量。针对不同密钥和求值构成的测试集所得到的方程组，最多包含 33 个剩余变量，并且都可以通过一种称为 MFCS [22] 的有限域特征集方法变体求解。为了降低方程的次数，

该攻击仅使用密钥流中的 190 比特，因此可能产生伪解。

在其他类型的攻击中，值得注意的是 Maximov 和 Biryukov [35] 提出的攻击。该攻击会猜测某些特定状态位（或状态位的乘积）的值，在某些情况下，这可以将求解二次方程组转化为求解线性方程组。因此，这种代数攻击包括求解大约 $2^{83.5}$ 个由 192 个方程组成的稀疏线性方程组。由于方程组具有稀疏性，求解每个方程组的复杂度约为 2^{16}。需要注意的是，要使攻击成功，需要一段相当长的密钥流，比特数为 $2^{61.5}$，这十分难以实现。

最后，谈到对 Trivium 的差分密码分析，我们提及 Dinur 和 Shamir 在 [16], 中提出的立方攻击。在该攻击中，攻击者会确定一些称为超级多项式的特殊多项式，其变量包含密钥位。超级多项式的计算方法是，对一组初始向量求和，这组初始向量称为“立方体”。事实上，在这种攻击中，流密码基本上被视为一个黑盒，攻击者可以访问 IVs 及其对应的密钥流。这种差分攻击的许多变体和改进都针对 Trivium的简化变体（例如，参见 [12] 中的综述）。所谓简化，是指对 Trivium 进行修改，减少预热更新的次数 N （所得方案通常称为 N-轮 Trivium），或使用规模更小、更新方程更简单的寄存器。在这一研究方向的最新成果中，值得一提的是 [28]，其中针对最多 848 轮的Trivium得到了超级多项式。

本文的组织结构如下。第 2 节介绍在任意有限域 $\mathbb{F}$ 上求解多项式系统的基本事实和技术。特别地，我们回顾此类多项式系统求解次数的一个多项式界，并推导出一个单指数渐近复杂度公式，用于计算包含 $\mathbb{F}$ 的域方程的理想的 DegRevLex-Gr¨obner 基。该公式的指数涉及变量数和域 $\mathbb{F}$ 的基数的平方。最后，我们证明了一个关于包含一组显式方程的多项式系统的消元定理。第 3 节介绍能够计算多项式理想 J 的 $\mathbb{F}$-解集 $V_{\mathbb{F}}(J)$ 的MultiSolve 算法，其中 $V_{\mathbb{F}}(J) \leq 1$。该算法实现了我们提出的多步策略，并讨论其参数如何改变算法的行为。第 4 节从调用核心子程序 GBElimLin （生成线性多项式并利用它们进行消元）以及 GrobnerBasis 的次数方面，研究算法 MultiSolve的复杂度。这些次数由显式公式给出，其中包含一些概率分布；针对不同数量的已赋值变量，在足够大的测试集上应用 GBElimLin，即可可靠地估计这些概率分布。我们证明，对于完整多步策略（即每次向评估集添加一个变量的策略），调用 GrobnerBasis （或其他多项式方程组求解器）的次数最少。第 5 节回顾差分流密码 [31, 32] 的概念，该概念可作为基于反馈移位寄存器的流密码的一种恰当形式化；第 6 节讨论对此类密码可能实施的代数攻击。第 7 节将流密码 Trivium 描述为差分流密码，并计算其状态转移映射的逆映射，从而实施内部状态攻击。第 8 节介绍一项广泛的实验研究，考察此类代数攻击的复杂度

如何通过算法 MultiSolve 对 Trivium 实施。我们实现了平均复杂度为调用 GrobnerBasis 求解器 $2^{106.2}$ 次，且所有基都能轻松计算。这优于以往任何一种基于已知相当短密钥流的代数攻击，例如 [29] 中的攻击需要求解 2^{115} 个多项式系统。尽管这证实了 Trivium 的安全性，我们的内部状态攻击仍不如针对其 80 位私钥的暴力破解。不过，我们乐观地认为，随着多项式系统求解器不断改进，并针对 MultiSolve 算法的参数对 Trivium 进行更精细的调优，两者之间的差距有望进一步缩小。最后，第 9 节总结了其他一些结论。

2. Solving polynomial systems over finite fields

本节简要回顾一些基本结果，这些结果涉及如何求解系数和解均取自有限域、且解的数量较少的多项式系统。在密码系统的代数攻击中，这种情形相当常见，因为给定数据通常可以唯一确定密钥。我们先固定一些记号。设 $\mathbb{F}$ =GF(q) 为一个有限域，并考虑多项式代数 $R = \mathbb{F}[x_1, \ldots, x_n]$ 和理想 $L = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle \subset R$。若 $J \subset R$ 是一个理想，且 $\mathbb{F}$ 是域 $\mathbb{F}$ 的代数闭包，我们记

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

和 $V_{\mathbb{F}}(J) = V(J) \cap \mathbb{F}^n$。

有限域上的零点定理（例如，参见 [26]）立即推出以下结果。

命题 2.1. 设 $J \subset R$ 为一个理想。我们有 $V(L) = \mathbb{F}^n$ 且 $V_{\mathbb{F}}(J)$ =$V(J + L)$，其中 $J + L$ 是 R 的一个根理想。

由于 $V(L) = \mathbb{F}^n$，理想 L 的生成元称为域方程。上述结果和零点定理的一个直接推论是以下结果，它提供了一种求解基域 $\mathbb{F}$ 上至多有一个解的多项式系统的方法。

命题 2.2. 设 $J \subset R$ 是一个理想，且满足 $\#V_{\mathbb{F}}(J) \leq 1$。那么，理想 $J + L$ 的（约化）通用 Gr¨obner 基 G，即相对于 R 的任意单项式序的 Gr¨obner 基，为

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

事实上，根据命题 2.2 的推论，我们可以选择最高效的单项式序（如反字典序），来求解只有一个解或无解的多项式系统。此外，当 $V_{\mathbb{F}}(J)$ 只有少数解时，请注意，求出这些解的代价本质上仍相当于计算 $J + L$ 的反字典序-Gr¨obner 基。实际上，要计算 $V_{\mathbb{F}}(J) = V(J + L)$，需要借助 FGLM 算法 [20] 将这样的基转换为 Lex-Gr¨obner 基，其复杂度为 $\mathcal{O}(nk^3)$，其中 $k = \#V_{\mathbb{F}}(J) = \dim_{\mathbb{F}} R/(J + L)$。若整数 k 较小，这种复杂度便由计算反字典序-Gr¨obner 基的代价所主导。

对这类成本进行精确估算通常很困难，但已知对于任何单项式序，最坏情形的复杂度相对于变量数呈双指数增长 [17]。如果 $\mathbb{F} = \text{GF}(q)$，因此 $V_{\mathbb{F}}(J) = V(J + L)$ 是有限集，那么算法

利用线性代数计算 DegRevLex-Gr¨obner 基的算法具有如下形式的复杂度公式（例如参见 [6, 8]）：

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

其中 $2 < \omega \leq 3$ 是线性代数指数，也就是说，$\mathcal{O}(n^{\omega})$ 是求解一个包含 n 个变量的线性方程组所需的复杂度，而 d_s 是求解次数，即完整 Gröbner 基计算中涉及的 S 多项式的最高次数（例如，参见 [11]）。

注意，$\binom{n+d_s}{d_s}$ 是由 n 个变量构成的、次数为 $\leq d_s$ 的单项式的数量。它对应于所谓的“Macaulay 矩阵”的列数；在这些矩阵中，多项式被视为相对于单项式基的系数向量。

如果 d_s 相对于变量数量 n 为常数，则显然可得到多项式复杂度。如果一个理想包含 $\mathbb{F}$ 的域方程，则求解次数 d_s 受 n 的二次函数界定，这由 Macaulay 界的以下推论确立（例如，参见 [11] 中的定理 11）。

命题 2.3. 设 $J = \langle f_1, \ldots, f_m \rangle$ 是 $R = \mathbb{F}[x_1, \ldots, x_n]$（$\mathbb{F}$ =GF(q)）的一个理想，并考虑 $L = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle \subset R$。令 $d = \max\{d_1, \ldots, d_m, q\}$，其中 $d_i = \deg(f_i)$。那么，计算 $J + L$ 的一个反字典序-Gr ¨obner 基的求解次数 d_s 满足以下界

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

假设 J 的生成元模 L 均为标准形，则有 $d \leq n(q-1)$ ($n \geq 2$)，并且还满足以下界

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

我们注意到，在许多具体情形中，上述界并不紧。然而，这些界有助于说明，对包含有限域域方程的理想进行反字典序-Gr¨obner 基计算具有单指数复杂度（另见 [21] 中的命题 4 和 [29] 中的定理 5.2）。

命题 2.4. 令 $n \geq \max\{2, q-1\}$。一个用于计算理想 $J + L \subset R$ 的 DegRevLex-Gr ¨obner 基的（线性代数）算法满足以下渐近复杂度

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

证明。我们首先对二项式复杂度 (1) 中出现的阶乘应用斯特林公式。回忆该公式为

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

将上述估计也应用于 $(n+d_s)!$ 和 $d_s!$，我们得到

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

第一个因子 $((n+d_s)/(2\pi n d_s))^{1/2}$ 是一个上界为 1 的非递增函数，而第二个因子 $e^{12^{1}_{(n+d_s)} - \frac{1}{12n+1} - \frac{1}{12d_s+1}}$ 的上界为 $\sqrt{e}$。因此，我们进一步得到

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

现在来考虑求解次数的界 (3)。我们得到

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

通过观察到当 $n \geq q-1$时，$(n^2+n)(q-1) \leq n^2q$，我们得出结论

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

□

请注意，在密码学中，当理想的生成元 $J = \langle f_1、\ .....、\ f_m \rangle$ 被视为随机生成的半正则序列时，会使用更好的求解次数 d_s 估计（例如，参见 [5, 8]）。借助这类估计以及 DegRevLex-Gr¨obner 基的复杂度 (1)，可近似计算求得解集 $V_{\mathbb{F}}(J)$ 所需的计算量，并据此评估密码系统安全性。

在接下来的几节中，我们将利用这样一个观点：在多项式系统中含有线性方程，或更一般地说，含有显式方程，可以在保留解的同时消去变量。由于求解复杂度在很大程度上取决于变量的数量，因此这种消元可以显著改善解的搜索过程。以下结果将对这一观点及其有效性进行形式化阐述。在本节结束之前，我们假设 $\mathbb{F}$ 是任意基域。

设 $R = \mathbb{F}[x_1, \ldots, x_n]$、$R' = \mathbb{F}[x_1, \ldots, x_m]$ 是满足 $n \geq m$ 的多项式代数。设 $f_1, \ldots, f_{n-m} \in R'$，并考虑代数同态 $\psi\colon R \to R'$，使得

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

显然，$\mathrm{Im}\ \psi = R'$ 和 $I = \langle x_{m+1} - f_1, \ldots, x_n - f_{n-m} \rangle \subset \subset \mathrm{Ker}\ \psi$。在代数 R 中对理想 I 取模进行改写后，可得 $\mathrm{Ker}\,\psi = I$。换言之，商代数 R/I 在映射 $f + I \mapsto \psi(f)$ 下与子代数 $R' \subset R$ 同构。

命题 2.5. 设 $I \subset J \subset R$ 是一个理想。消去理想$J \cap R'$ 等于 $\psi(J)$。

证明。由于 ψ 是 R' 上的恒等映射，我们有 $J \cap R' = \psi(J \cap R') \subset \psi(J)$。设 $f \in J$，并考虑 $f' = \psi(f) \in \psi(J) \subset R'$。我们要证明 $f' \in J$。由于 $f' \in R'$，我们有 $\psi(f') = f' = \psi(f)$，因此 $f - f' \in I \subset J$。因为$f \in J$，我们得出 $f' \in J$，从而 $\psi(J) \subset J \cap R'$。□

考虑映射 $\varphi\colon \mathbb{F}^m \to \mathbb{F}^n$，使得

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

注意，φ 是与满射代数同态 ψ相对应的单射多项式映射，也就是说，对于所有 $g \in R$ 和 $\bar{}(a_1, \ldots, a_m) \in \mathbb{F}^m$，都有 $\psi(g)(a_1, \ldots, a_m) = g(\varphi(a_1, \ldots, a_m))$。现在考虑解集 $V(J) = \{(a_1, \ldots, a_n) \in \mathbb{F}^n \mid f(a_1, \ldots, a_n) = 0$，其中 $\mathbb{F}$ 是域 $\mathbb{F}$ 的代数闭包，对所有 $f \in J\}$ 均成立。

命题 2.6。设 $I \subset J \subset R$ 是一个理想。我们有 $V(J) = \varphi(V(J \cap R')) = \varphi(V(\psi(J)))$。

证明。设 $\alpha = (a_1, \ldots, a_n) \in V(J)$，即对所有 $f \in J$，都有 $f(\alpha) = 0$。若 $\alpha' = (a_1, \ldots, a_m)$，显然对所有 $f' \in J \cap R'$，都有 $f'(\alpha') = 0$，也就是说，有 $\alpha' \in V(J \cap R')$。由于 $I = \langle x_{m+1} - f_1, \ldots, x_n - f_{n-m} \rangle \subset J$，可得 $a_{m+1} = f_1(\alpha') \ldots, a_n = f_{n-m}(\alpha')$，因此我们得出 $V(J) \subset \varphi(V(J \cap R'))$。

现在假设 $\alpha' = (a_1, \ldots, a_m) \in V(J \cap R')$，并考虑 $\alpha = (a_1, \ldots, a_n) = \varphi(\alpha')$。令 $f \in J$，并记 $f' = \psi(f) \in \psi(J)$。根据命题 2.5，我们有$f' \in J \cap R'$；又由于 $\psi(f') = f' = \psi(f)$，因此有 $f - f' \in I$。这意味着$f'(\alpha') = 0$且 $f = f' + g$，其中 $g \in I$。由于 $\alpha = \varphi(\alpha')$，我们有 $f(\alpha) = 0$。事实上，有 $g(\alpha) = g(\varphi(\alpha')) = \psi(g)(\alpha') = 0$，因为 $g \in I = \mathrm{Ker}\, \psi$。由此可得 $\varphi(V(J \cap R')) \subset V(J)$。最后，由于根据命题 2.5 有 $J \cap R' = \psi(J)$，因此有 $V(J) = \varphi(V(J \cap R')) = \varphi(V(\psi(J)))$。□

非正式地说，上述命题指出，包含一些显式方程 $x_{m+i} = f_i$ 的多项式系统的所有解，都可以通过扩展消去变量 x_{m+i} 后得到的多项式系统的解来求出。注意，如果 $\deg(f_i) > 1$，那么消去方程中的变量所带来的优势，会因方程次数增加而部分抵消，而次数增加可能会影响方程组的求解次数。通过线性方程消去变量时则不会出现这种情况，因此这是更理想的情形。下一节将进一步阐述并探讨这一观点。

3. 通过多步策略求解

上一节中的复杂度分析表明，对于变量数量相当多的情形，计算 Gr¨obner 基可能是一项艰巨的任务。若将 Gr¨obner 基替换为 SAT、BDD 等其他求解器，计算量也不会有显著变化（例如，参见 [31]）。事实上，众所周知，在有限域上求解多元多项式系统是 NP-困难问题 [23]。因此，一种实用的技术是：将难以完成的 Gr¨obner 基计算，替换为多个可行的计算；这些计算针对原方程组经过如下处理后得到的多项式系统：在基有限域上对一部分变量赋值。当然，指数级复杂度隐藏在这种穷举求值中。这种方法通常称为猜测并确定策略或混合策略；后一名称尤其用于随机生成的多项式系统 [5, 8]。在这种情况下，对一些变量赋值，本质上会再次得到一个随机方程组，其复杂度可以通过理论方法估算。对总复杂度的计算可能表明，在某些情况下，混合方法比单次 Gr¨obner 基计算更快 [7, 8]。

本节主要介绍这种技术的一种变体：给定的多项式系统通常并非随机，因此对其变量的某个子集进行赋值，可能会得到表现各异的方程组。具体而言，我们分析所得方程组的一项主要性质：通过不完全的 Gr¨obner 基计算，并在选定的次数 D 处停止计算，可以得到多少个线性方程。事实上，这些线性方程的存在，使我们能够在不提高系统方程次数的情况下，相对于已赋值变量集合进一步消去一定数量的变量。这些方程组中剩余变量的数量（简称 NRV）让我们得以估计彻底求解它们所需面对的复杂度。给定一个界限 $0 \leq B \leq n$，只有在 NRV $\leq B$ 时，我们才决定计算完整的 Gr¨obner 基。否则，我们会再对一定数量的变量赋值，并重新计算 NRV

然后将其与 B 进行比较。通过这种方式，我们能够利用 Gr¨obner 基求解给定的多项式系统，且所涉及的变量数量始终不超过 B。假设每次 Gr¨obner 基计算都能在合理时间内完成，那么复杂度实质上取决于计算次数。我们将说明如何通过显式公式和简单统计，在实践中估算这一次数；以及如何通过选择合适的步骤来优化它，即逐步加入新的待赋值变量。我们将这种方法称为多步策略，因为它包含这些步骤，而且 Gr¨obner 基也是分步计算的。在对该策略作了上述高层次的描述之后，下面将对其进行正式阐述。

设 $R = \mathbb{F}[x_1, \ldots, x_n]$ 是有限域 $\mathbb{F} =$ GF(q) 上的多项式代数。考虑一个理想 $J = \langle f_1, \ldots, f_m \rangle \subset R$，并假设 #$V_{\mathbb{F}}(J) \leq 1$。按惯例记作 $L = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$，并令 $H = \{f_i\} \cup \{x_i^q - x_i\}$ 为理想 $J + L$ 的一个生成集。设 $D > 0$ 为整数，并用 GrobnerBasis(H, D) 表示一个执行不完全 Gr¨obner 基计算的算法；当某一当前步骤中所有次数为 $\leq D$ 的 S 多项式都已处理后，算法即停止。使用线性代数方法时，该算法相当于对 Macaulay 矩阵的大小设置上限。如果 D 足够大，也就是说，它是一个求解次数，那么 $G =$ GrobnerBasis(H, D) 就是 $J + L$ 的完整 Gr¨obner 基。在所考虑的情形中，若 #$V_{\mathbb{F}}(J) \leq 1$，完整 Gr¨obner 基 G 只包含线性多项式，即 $G = \{1\}$ 或 $G = \{x_1 - a_1, \ldots, x_n - a_n\}$（见命题 2.2）。当然，计算完整 Gr¨obner 基通常代价高昂，因为其复杂度为 (1)；而已知属于理想 $J + L$ 的线性多项式，有助于在不增加次数的情况下消去变量，这一点可由命题 2.6 得知。因此，合理的做法是以相当低的次数 D 计算 $G =$ GrobnerBasis(H, D)，以实现快速计算，并利用我们可能在 G 中找到的线性多项式，从 $J + L$ 的生成元中消去变量。如果 $G = \{1\}$ 或 $G = \{x_1 - a_1, \ldots, x_n - a_n\}$，则无需进一步计算。否则，这种预处理通常会简化计算 $V_{\mathbb{F}}(J)$ 的任务。我们将其形式化为算法 3.1。

在算法 GBElimLin中，我们假设集合 G_1中的线性多项式已完全相互约化，也就是说，它们是在 Macaulay 矩阵中以简化行阶梯形得到的。过程 Reduce 是 Gröbner 基理论中的完全约化算法（例如，参见 [2]）。如果满足 max{deg(g) | $g \in G$} ≤ 1，即 $G = \{1\}$ 或 $G = \{x_1 - a_1, \ldots, x_n - a_n\}$，那么 GBElimLin 会立即给出 $V_{\mathbb{F}}(J) = V(J + L)$，因为此时 D 就是求解次数。否则，根据命题 2.6，理想 $J' + L' = \langle G \rangle$ ($G =$GBElimLin(H, D)) 的解集与 $V(J + L)$ 直接相关。在这种情况下，我们将 NRV > 0 定义为出现在 G中的变量数。请注意，为了计算可能解的所有坐标，我们还应输出过程 GBElimLin所得的线性方程集合 G_1 。我们不这样做，因为在接下来要介绍的求解算法中，我们只需在 GBElimLin处调用一次该过程，并使用这个集合。

由于不完全 Gr¨obner 基计算 GrobnerBasis(H, D) 是在 GBElimLin 中进行的，并且假设 H的完全 Gr¨obner 基是完全线性的，因此线性生成元集 G_1 通常非空。因此，

算法 3.1 GBElimLin

输入： 理想 $J+L$ (#$V_{\mathbb{F}}(J) \leq 1$) 的生成集 H 以及一个整数 $D>0$;

输出： $J+L$ 的一个格罗布纳基，或一个理想的生成集通过线性方程消去部分变量得到，来自 $J+L$。

$G :=$ GrobnerBasis(H, D);
若 max{deg(g) $\mid g \in G\} \leq 1$ ，则
　返回 G;
结束如果;
$G_1 := \{g \in G \mid$ 的次数（g）$= 1\}$;
$G_2 := G \setminus G_1$;
$S := \{x_i^q - x_i \mid x_i$ 出现在 $G\}$中;
$G :=$ Reduce（G_2, $G_1 \cup S$）;
$S := \{x_i^q - x_i \mid x_i$ 出现在 $G\}$中;
$G := G \cup S$;
返回 G;

由于消去过程 Reduce($G_2, G_1 \cup S$)，剩余变量 NRV 的数量通常少于 H 中的变量数量。请注意，利用线性方程消去变量不会改变 G_2 中非线性生成元的次数，甚至可能降低其次数，这是由 S中的域方程导致的。因此，计算解集 $V_{\mathbb{F}}(J')$比计算$V_{\mathbb{F}}(J)$更容易。这种简化是以执行 GBElimLin为代价实现的；当 D 的次数较低时，这一操作的成本可能相当低。例如，当$D = \max\{\deg(g) \mid g \in H\}$ 且 q 是一个较小的整数时，就会出现这种情况。请注意，在此情况下，过程 GrobnerBasis(H, D)本质上是 Gr¨obner 基理论中的一种互约化，而过程 GBElimLin实际上与基于线性代数的算法 ElimLin [4, 13]一致。

现在固定一个界 $0 \leq B \leq n$ ，并假设在变量数小于或等于 B 的情况下，可以在合理的时间内计算出一组完整的 Gröbner 基。若变量数 n 相当大（例如密码学中出现的多项式系统），我们就无法立即应用 GBElimLin(H, D)。接着，我们向 $J+L$ 的生成集 H 中加入一组对适量适当变量的求值，例如

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

并计算 $G =$ GBElimLin($H \cup E, D$)。如前所述，我们假设 H 的完整 Gr¨obner 基，因此也是 $H \cup E$ 的完整 Gr¨obner 基，是线性的。更确切地说，如果 G' 是 $H \cup E$ 的完整 Gr¨obner 基，那么 $G' = \{1\}$ 或 $G' = \{x_1 - a_1, \ldots, x_n - a_n\}$。虽然 D 通常低于 $H \cup E$ 的求解次数，但在 GBElimLin 中计算出的 $H \cup E$ 的不完整 Gr¨obner 基 G'' 会趋近于线性基 G'，并在 $G'' \neq \{1\}$ 时包含 E 。换言之，除了 E 中的那些多项式之外，通常还会计算出一些新的线性多项式，也就是说，NRV 通常小于 $n-k$。对于非随机方程组，这一点尤为明显，例如源自特定（流）密码密码分析的方程组。

这提示我们应适当调整数量 k，该数量决定了在域 $\mathbb{F} = \mathrm{GF}(q)$ 上对 k 个变量进行穷举求值所对应的指数复杂度 q^k。如果 NRV $\leq B$，我们就继续计算 GrobnerBasis(G)。否则，我们将评估集 E 扩展为

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

并计算 $G' = \mathrm{GBElimLin}(H \cup E', D)$。通过迭代上述步骤，每次增加少量待求值的变量，即可得到解集 $V_{\mathbb{F}}(J)$ ($\#V_{\mathbb{F}}(J) \leq 1$)，直到所有考虑的求值均已穷尽，或 GBElimLin or GrobnerBasis (NRV $\leq B$) 的输出是满足 $\max\{\deg(g) \mid g \in G\} = 1$ 的 Gr¨obner 基 G。算法 3.3 实现了这一多步策略，并妥善管理对不同数量变量进行的穷举求值。一个重要的子程序是算法 3.2。

算法 3.2 StepSolve

输入：　理想 $J + L$ ($\#V_{\mathbb{F}}(J) \leq 1$) 的一个生成集 H，一个子集 $W \subset \mathbb{F}^k$（$1 \leq k \leq n$）和三个整数 $k \leq l \leq n$、$0 \leq B \leq n$、$D > 0$;

输出：要么是一对（解，G），其中解是一个字符串，且 $G \neq \{1\}$ 是 $J + L$ 的格罗布纳基，或是一个理想的格罗布纳基该理想由消去其中一些变量（通配集，W'），其中通配集是一个字符串，且 W' 是一个 $\mathbb{F}^l$的子集。　$J + L$ 得到；要么是这一对

如果 $k \neq l$ 则
　$W := W \times \mathbb{F}^{l-k}$;
结束如果;

$W' := \emptyset$;
对于所有$(a_1, \ldots, a_l) \in W$执行
　$E := \{x_1 - a_1, \ldots, x_l - a_l\}$;
　$G := \mathrm{GBElimLin}(H \cup E, D)$;
　如果 $\max\{\deg(g) \mid g \in G\} = 1$ 则
　　打印 E;
　　返回 (解, G);
　结束 if;
　如果 $G \neq \{1\}$ 则
　　NRV: $=$ G中的变量数;
　　如果 NRV $\leq B$ 则
　　$G := \mathrm{GrobnerBasis}(G)$;
　　如果 $\max\{\deg(g) \mid g \in G\} = 1$ 则
　　打印 E;
　　返回 (解, G);
　　结束 if;
　　else
　　$W' := W' \cup \{(a_1, \ldots, a_l)\}$;
　　结束 if;
　结束 if;
结束 for;
返回（通配集，W'）;

注意， StepSolve 可能会输出一个 Gr¨obner 基 $G \neq \{1\}$，它属于通过 GBElimLin 从输入理想 $J + L$ 中消去部分变量后得到的理想。根据命题 2.6，只需通过 procedure GBElimLin($(H \cup E, D)$ 中计算出的线性方程扩展 G 中的坐标，即可得到 $J + L$ 的实际 Gr¨obner 基 $G' = \{x_1 - a_1, \ldots, x_n - a_n\}$，或者等价地得到 $V_{\mathbb{F}}(J) = \{(a_1, \ldots, a_n)\}$；其中 $E = \{x_1 - a_1, \ldots, x_l - a_l\}$ 由 StepSolve 打印。确实，只有在这一特殊情形下，这些方程才是必要的。

算法 3.3 MultiSolve

输入： 理想 $J + L$ ($V_{\mathbb{F}}(J) \leq 1$) 的一个生成集 H，一个整数序列
整数 $1 \leq k_1 < \ldots < k_r \leq n$ 和两个整数 $0 \leq B \leq n$、$D > 0$;
输出：要么是 $J + L$ 的 Gr¨obner 基，要么是一个理想的 Gr¨obner 基
通过线性方程消去部分变量而得到，来自
理想 $J + L$。

对于所有 $(a_1, \ldots, a_{k_1}) \in \mathbb{F}^{k_1}$ 执行
$W := \{(a_1, \ldots, a_{k_1})\}$;
$i := 1$;
当 $W \neq \emptyset$ 时执行
$(A_1, A_2) :=$ StepSolve(H, W, k_i, D, B);
如果 $A_1 =$解 则
返回 A_2;
结束如果;
$W := A_2$;
$i := i + 1$;
结束 while;
结束 for;
返回 $\{1\}$;

算法 MultiSolve 在以下条件下会产生所需的输出：对于每个 $(a_1, \ldots, a_{k_1}) \in \mathbb{F}^{k_1}$，在 StepSolve 处调用最终步骤 k_r后，当 $A_1 =$ 是通配集时， A_2 为空集。事实上，我们假设$1 \leq k_r \leq n$ 是满足这一条件的最小整数。当然，这样的整数必定存在，因为当 $k_r = n$ 时，我们可以对所有变量进行赋值。注意，我们的求解算法只需在StepSolve 的每次迭代中，以及针对每个猜测 $(a_1, \ldots, a_{k_1})$，存储通配集 W。如果找到一个解，算法便会立即终止，避免进一步进行无用的计算。

我们指出，在只有一次迭代的情况下，即对于 $r = 1$ 和$k_r = k_1$, MultiSolve 是对标准猜测并确定（或混合）策略的推广。与标准策略唯一的区别在于， MultiSolve 分两步计算所有赋值 $(a_1, \ldots, a_k) \in \mathbb{F}^k$ ($k = k_r = k_1$) 的 Gr¨obner 基：首先执行 GBElimLin ，然后执行 GrobnerBasis。如果 NRV($\leq B$) 远小于 $n - k$，这种方法可以减少计算时间，尤其是对于基于 Macaulay 矩阵和线性代数的 Gr¨obner 基算法。事实上，对标准策略的这一改进已出现在算法 ElimLin [4, 13]中。

我们提到，在许多计算机代数系统中，例如 Magma，基于生成足够数量线性方程的 GrobnerBasis 算法中集成了重启机制。要在我们的 MultiSolve

算法中使用它，则应能够输入参数 B 和D ，以便在达到次数 D 但 NRV $> B$时停止可能无法完成的计算。在这种情况下，还应提供指令，以增加穷举评估的变量数量。在下一节中，我们将说明，用于 $r > 1$ 的算法 MultiSolve 确实优于标准策略，因为它减少了计算解集 $V_{\mathbb{F}}(J)$所需的 Gr¨obner 基的总数。

请注意，如果 $B = 0$ 算法 MultiSolve 仅通过应用 GBElimLin来计算输出，也就是说，通过在次数达到 D时停止的 Gr¨obner 基计算，那么对于足够大的最终步骤 $k_r \leq n$，所有所得方程组的求解次数都会变得小于或等于 D，因此这是可行的。

在应用 MultiSolve之前，先对理想 $J + L$的生成集 H 运行 GBElimLin 会很有帮助。事实上，当 H 的变量数量很大时，这通常是不可行的。不过，如果 H 中给出了线性多项式，一个好的做法是先利用这些生成元消去变量。我们指出，除了整数 D 和 B的选择之外，算法 MultiSolve 优化中的一个关键问题是对子集的选择

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

在下一节中，我们将说明，步骤 k_i 的最优选择是令$k_{i-1} = k_i - 1$。因此，关键问题是找出对子集 $\{x_1, \ldots, x_{k_r}\}$的选择，使其对 Mul-tiSolve的总运行时间产生最有利的影响。我们建议，在搜索要赋值的最优变量子集时，可以将适当大型测试集 $T \subset \mathbb{F}^{k_r}$上对 $\{x_1, \ldots, x_{k_r}\}$进行赋值时由 GBElimLin 得到的 NRV 数值的平均值用作目标函数。当然，如果所有变量组成的集合所含的这类子集数量巨大，就无法通过暴力破解来进行这种优化。在这种情况下，我们可以在数量有限的子集中进行搜索，这些子集因给定方程组的特定形式而显得较为理想。设计更有效的优化策略仍是一个开放问题，我们将留待今后研究。

4. MultiSolve complexity

本节分析算法 MultiSolve 的复杂度，实质上对应于对子程序 GBElimLin 和GrobnerBasis的调用次数。事实上，对于参数 $D > 0$ 和 $0 \leq B \leq n$的适当取值，这两个过程都能在合理的时间内执行。

我们将对多项式代数$R = \mathbb{F}[x_1, \ldots, x_n]$ ($\mathbb{F} = \mathsf{GF}(q)$)的前 k 个变量进行赋值称为 k-猜测；也就是说，进行猜测（$a_1, .., a_k$） $\in \mathbb{F}^k$ 意味着我们正在对由以下集合生成的理想取模

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

如果过程 GBElimLin($H \cup E, D$)（H 是理想 $J + L$的生成集）输出一组生成元 G，使得$\max\{\deg(g) \mid g \in G\} > 1$，且其对应的NRV严格大于 B，则称该猜测为野性猜测。否则，我们称该猜测为驯服猜测。请注意，后者包括 GBElimLin输出完整的Gr¨obner基的情况，即$\max\{\deg(g) \mid g \in G\} \leq 1$ ，因为 $V_{\mathbb{F}}(J) \leq 1$。

记 $p_B(k)$ 为集合 $\mathbb{F}^k$ 中所有 k 猜测里野生 k 猜测的概率，也就是说， $p_B(k)q^k$ 是算法 GBElimLin 在其中确定的野生 k 猜测的数量

在集合 $\mathbb{F}^k$ 中确定的数量。实际应用中，这种概率是在规模足够大的某个随机测试集 $T \subset \mathbb{F}^k$ 上估算的。

我们将要介绍的关于 MultiSolve 复杂度的结果，均基于这样一个安全假设：将一个驯服的 k-猜测扩展为一个 l-猜测（$k < l$）时，我们总能再次得到一个驯服的 l-猜测。这是因为，通过赋值减少理想中的变量数量，也会降低其求解次数，而由 GBElimLin 计算得到的不完全 Gröbner 基会趋近于完全 Gröbner 基，后者是完全线性的。第一个推论是，映射 $k \mapsto p_B(k)$ 是一个递减函数。

命题 4.1。设 $1 \leq k_1 \leq \ldots \leq k_r \leq n$。有 $1 \geq p_B(k_1) \geq \ldots \geq p_B(k_r) \geq 0$，其中 $p_B(n) = 0$。

证明。设 $1 \leq k \leq l \leq n$。扩展野生 k-猜测的所有 l-猜测的数量为

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

由于将一个 wild l-猜测限制后，总能得到符合上述假设的 wild k-猜测，因此集合 $\mathbb{F}^l$ 中的所有 wild l-猜测，都是通过扩展某个 wild k-猜测得到的。这意味着 $p(k)q^l \geq p(l)q^l$，因此 $p(k) \geq p(l)$。请注意，wild k-猜测的某些扩展可能会变成 tamed l-猜测。特别地，显然 $p_B(n) = 0$，因为我们已经知道R中所有变量的赋值。□

从现在开始，我们记作

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

给定步骤序列 $1 \leq k_1 \leq \ldots \leq k_r = k''$，我们希望计算算法 MultiSolve 中所考虑的猜测总数，该数等于对子程序 GBElimLin的调用次数。我们证明，该数仅取决于概率 $p_B(k_1) \geq \ldots \geq p_B(k_{r-1})$，这些概率可以在不实际执行 MultiSolve的情况下，在测试集上进行估计。

命题 4.2. 算法 MultiSolve 在步骤 $1 \leq k_1 \leq \ldots \leq k_r = k''$ 中的猜测次数为

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

证明。为简化记号，设 $k < l < m < \ldots$ 为算法MultiSolve的步骤。在第一步 k 中，我们需要考虑所有 k 猜测，其数量为 q^k。在第二步 l 中，我们需要将所有野生 k 猜测扩展为 l 猜测，其数量为 $p_B(k)q^k$。因此，此类扩展的数量为

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

对于第三步 m，在所有扩展了野生 k-猜测的 l-猜测子集中，我们现在必须考虑将野生 l-猜测扩展为 m-猜测。由于限制一个野生 l-猜测总会得到一个野生 k-猜测，因此这个数量恰好等于集合 $\mathbb{F}^l$ 中所有野生 l-猜测的数量，即 $p_B(l)q^l$。那么，扩展野生 l-猜测（这些猜测扩展了野生 k-猜测）的 m-猜测数量为

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

我们推导出，直到第三步为止，算法中考虑的猜测总数为 MultiSolve

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

对算法的所有步骤进行迭代，可得复杂度为 C_1。 □

现在我们来考虑子程序 GrobnerBasis 在算法 MultiSolve中执行的次数；通常，这对其总运行时间贡献最大。根据定义，该次数就是算法中所考虑的驯化猜测的基数。

命题 4.3。算法 MultiSolve 中的驯化猜测数量为 s

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

证明。由上述结果可知，算法 Mul-tiSolve 中的野性猜测数量仅取决于概率序列 $1 \geq p_B(k_1) \geq \ldots \geq p_B(k_{r-1}) \geq p_B(k_r) = 0$。该总数为

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

我们得出，驯化猜测的总数为 $C_2 = C_1 - C$。 □

现在我们可以探讨，是否存在一个能使主要复杂度公式 C_2 达到最优的子集 $\{k_1, \ldots, k_r\} \subset \{1, \ldots, n\}$。回顾一下，在该公式中，最后一步固定为 $k_r = k'' = \min\{1 \leq k \leq n \mid p_B(k) = 0\}$。换言之，步骤 k'' 是满足以下条件的最小整数：GBElimLin$(H \cup E, D)$ (H 是 $J + L$ 的生成集，且 $E = \{x_1 - a_1, \ldots, x_{k''} - a_{k''}\}$) 对所有猜测 $(a_1, \ldots, a_{k''}) \in \mathbb{F}^{k''}$ 均获得 NRV $\leq B$。于是我们得到以下结果。

定理 4.4. 复杂度 C_2 的最小值在 $k_i = i (1 \leq i \leq k'')$ 时取得，也就是说，此时步骤数最多。

证明。要证明该命题成立，只需说明每次在两个步骤之间或所有步骤之前添加一个新步骤时，总和 C_2 都会减小。我们分别讨论这两种情况。

情况 1。设 $1 \leq k < l < m \leq k''$ 是算法 MultiSolve中的三个连续步骤。换言之，我们考虑可能添加在两个步骤之间的步骤 l。如果这三个步骤都出现，公式 C_2 包含以下两个加项

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

否则，如果缺少步骤 l，那么 C_2中只有一个对应的加项，即

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

注意，在这两种子情况下， C_2 中的其余加项都相同。现在，由于

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

并且 $(p_B(k) - p_B(l))q^m \geq (p_B(k) - p_B(l))q^l$，因为 $p_B(k) \geq p_B(l)$ 且 $m > l$，我们得出

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

情形 2. 设 $1 \leq k < l \leq k''$，并假设步骤 k 或 l 中有一个是第一步。换句话说，我们将 k 视为可能在所有步骤之前添加的一步。如果 k 是第一步，则公式 C_2 包含以下各项

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

否则，如果步骤 l 是第一步，也就是说，缺少步骤 k ，那么 C_2 中只有一个对应项，即

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

与情况 1 类似，可知

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

其中 $(1-p_B(k))q^l \geq (1-p_B(k))q^k$ ，因为 $1 \geq p_B(k)$ 且 $l > k$。因此我们得出

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

□

请注意，上述结果特别表明，与复杂度 C_2 相关的标准猜测并确定策略——即对单步 k'' 应用 MultiSolve ——实际上是最差的策略。至于第一步，设为 k'，上述结果表明 $k'=1$ ；但在实践中，对于给定参数 $D>0$， k' 是使得对所有 $(a_1,\ldots,a_{k'}) \in \mathbb{F}^{k'}$ 都能在合理时间内执行 $\mathrm{GBElimLin}(H \cup E, D)$ ($E=\{x_1-a_1,\ldots,x_{k'}-a_{k'}\}$) 的最小整数。对于具有大量变量的多项式系统，例如密码学中出现的此类系统，第一步 $1 \leq k' \leq n$ 将是一个不太小的整数。我们将对应于 C_2 最小值的策略称为完整多步策略，其复杂度公式如下

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

其中 $p_B(k'')=0$ ，并且按照约定， $p_B(k'-1)=1$。

当然，要得到运行时间，上述复杂度公式中的每一项都应乘以相应的平均耗时。事实上，相应的总运行时间如下：

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

其中， $\sigma(k-1,k)$ 是将 wild $(k-1)$-猜测扩展为 k-猜测时， GBElimLin 的平均运行时间； $\tau(k-1,k)$ 是将 wild $(k-1)$-猜测扩展为 tamed k-猜测时， GroebnerBasis 的平均运行时间。请注意，对于第一步 k'，我们采用的耗时为 $\sigma(k'-1,k')$ 和 $\tau(k'-1,k')$；按照约定，它们分别对应所有 k'-猜测和所有 tamed k'-猜测。因此， MultiSolve 的总执行时间为 $T=T_1+T_2$。

我们指出，在实验研究中（将在后续章节详述），我们观察到概率 $p_B(k)$ ($k' \leq k \leq k''$) 似乎是关于多元多项式方程组的可靠统计数据。因此，可以放心地用这些数据来估算通过多步算法 MultiSolve求解多项式系统的复杂度，也可以通过调整其主要参数，即整数 D、 B 以及用于求值的子集

个 k'' 变量，来将其调至最优复杂度。还请注意，在算法 MultiSolve 中，可以用任何其他有限域上的多项式系统求解器替换 GrobnerBasis 求解器，例如 XL 求解器、SAT 求解器等。在 MultiSolve中，唯一必须进行（不完全）Gr¨obner 基计算的步骤，是子程序 GBElimLin；该步骤利用变量取值生成额外的线性方程。另一种方法是使用线性代数算法 ElimLin [4, 13]。我们还要指出，概率 $p_B(k)$，以及由此得到的多步复杂度 C_1、C_2，是通过对某个测试集 $T \subset \mathbb{F}^k$执行 GBElimLin，针对所有 $k' \leq k \leq k''$ 得出的。

最后我们注意到，如果 $V_{\mathbb{F}}(J) \neq \emptyset$，并且我们可以估计，平均而言，解可通过一次经过调整（正确）的 l 猜测找到，且 $l \leq k < k''$，那么即使 $p_B(k) \neq 0$，我们基本上也可以将 k 视为 MultiSolve 的最后一步。这通常意味着在这种平均情况下，复杂度 C_1、C_2 会显著降低。例如，当我们使用包含许多不同密钥的测试集，研究源自对某个密码系统进行代数密码分析的多项式方程组时，就可能得到这样的估计。

为了说明算法 MultiSolve在密码学领域具有实际应用价值，接下来的章节将介绍并攻击著名的流密码 Trivium [15]。具体而言，我们将在近期论文 [31, 32] 提出的“差分流密码”概念所提供的一般框架下，介绍这一密码方案及其密码分析。

5. Stream ciphers

流密码本质上是伪随机函数的实际实现（例如，参见 [1, 30, 34]）。从宏观上看，流密码接收两个固定长度的随机字符串作为输入，通常称为种子和初始化向量，并生成一个任意长度、看似随机的字符串，称为密钥流。这些字符串中的字符通常是有限域的元素，最常见的是比特。要构造对称密码，可以用密钥流加密明文流，或通过简单的加法或减法解密密文流。在这种情况下，种子和初始化向量分别是该密码的密钥和随机数。如果流密码与真正的随机函数不可区分，则认为它是安全的。

已知的大多数流密码构造都依赖于反馈移位寄存器（简称 FSR）。反馈移位寄存器是由一组存储单元组成的数组，每个单元存储给定有限域 $\mathbb{F} = \mathrm{GF}(q)$中的一个元素，并通过某个函数 f进行更新。更准确地说，在一次更新过程中，各单元的值向左移位，而最右侧单元的新值则是函数 f 作用于同一寄存器中某个给定子集的单元当前值（也可能包括其他反馈移位寄存器的单元当前值）所得的结果。对于有限域，函数 f 总能转换为多元多项式，因此称为更新多项式。如果多项式 f 是线性的，则称该反馈移位寄存器为线性反馈移位寄存器；否则称为非线性反馈移位寄存器。

一种流密码 $\mathcal{C}$可以通过组合若干反馈移位寄存器 $\mathcal{F}_1$、.....、$\mathcal{F}_n$、它们各自对应的更新多项式 f_1、.....、f_n，以及一个系数取自 $\mathbb{F}$ 的额外多元多项式g 来构造，该多项式称为密钥流多项式。事实上，$\mathcal{C}$ 的密钥流是通过将多项式 g 作用于 $\mathcal{F}_i$中各寄存器的当前值而得到的。具体而言，反馈移位寄存器的存储单元最初填入

种子、初始化向量以及可能存在的常数字符串中的元素。随后， $\mathcal{F}_i$ 并行更新（ $h+u$）次，其中 h 是预期的密钥流长度， u 是预热阶段的更新次数（包括初始化）。该阶段在反馈移位寄存器初始化与密钥流输出之间留出一定间隔，以防止基于已知密钥流中若干元素而对流密码实施攻击。

描述由反馈移位寄存器得到的流密码，一种自然的语言是显式差分方程；正如 [32] 中首次指出的那样，此类流密码被称为差分流密码。为便于阅读，在给出正式定义之前，我们先回顾一些记号（另见 [25]）。

给定有限域 $\mathbb{F} = \mathrm{GF}(q)$ 和 $n \in \mathbb{N}^* = \mathbb{N} \setminus \{0\}$，我们用 R 表示变量无限集 $X = \bigcup_{t\in\mathbb{N}}\{x_1(t) \ldots, x_n(t)\}$ 上的多项式代数 $\mathbb{F}[X]$。代数自同态 $\sigma\colon R \to R$ 定义为令 $\sigma(x_i(t)) = x_i(t+1)$ $(1 \le i \le n, t \ge 0)$，称为 R 的移位映射。对于任意 $r_1, \ldots, r_n \in \mathbb{N}$，以下变量子集

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

定义了一个（有限生成的）子代数 $R = \mathbb{F}[X] \subset R$。

定义 5.1. 差分流密码 $\mathcal{C}$ 是一个由（代数）常微分显式差分方程构成的方程组，其形式为

$$\begin{cases} x_1(r_1+t) &= \sigma^t(f_1), \\ &\vdots \\ x_n(r_n+t) &= \sigma^t(f_n), \end{cases} \qquad (t \in \mathbb{N}) \tag{7}$$

以及一个多项式 $g \in R$，其中 $r_1, \ldots, r_n \in \mathbb{N}$ 和 $f_1, \ldots, f_n \in R$。

方程组 (7) 的 $\mathbb{F}$-解是其所有显式差分方程的解，也就是说，是一个由映射构成的 n 元组 $(a_1, \ldots, a_n)$，这些映射从 $\mathbb{N}$ 映射到 $\mathbb{F}$，并且满足：给定 $v(t) = (a_1(t) \ldots, a_1(r_1-1+t) \ldots, a_n(t) \ldots, a_n(r_n-1+t))$，对每个整数 $t \ge 0$，均有 $a_i(r_i+t) = f_i(v(t))$。我们称 $v(t)$ 为 $(a_1, \ldots, a_n)$ 的 t-状态，特别是 $v(0)$ 为初始状态。初始状态一旦固定，方程组 (7) 就有唯一的 $\mathbb{F}$-解 [32, 定理 2.4]。

注 5.2. 由显式差分系统(7)和多项式 $g \in R$ 定义的差分流密码 $\mathcal{C}$，可以实现为由一组$\mathcal{F}_1, \ldots, \mathcal{F}_n$反馈移位寄存器构成的流密码，这些反馈移位寄存器具有更新多项式 $f_1, \ldots, f_n$和密钥流多项式g。特别地，变量 $x_i(t)$（$1 \le i \le n$）对应于 $\mathcal{F}_i$的寄存器在时钟 $t \ge 0$时最左侧单元的符号值。如果 $(a_1, \ldots, a_n)$ 是(7)的$\mathbb{F}$-解，则其初始状态由种子、初始化向量以及可能存在的常数字符串组成。如果 $v(t) \in \mathbb{F}^r$ $(r = r_1 + \ldots + r_n)$ 是 $(a_1, \ldots, a_n)$ 的 t-状态，则称函数 $b\colon \mathbb{N} \to \mathbb{F}$ 满足对所有 $t \ge 0$ 均有 $b(t) = g(v(t))$，为 $(a_1, \ldots, a_n)$ 的密钥流。按照设计，密钥流的第 j 位可以设为 $b(u+j-1)$，其中 u 为该流密码的预热更新次数。反之，由某些反馈移位寄存器构成的流密码也可以形式化地描述为差分流密码。

6. 代数攻击

如前所述，安全的流密码 $\mathcal{C}$ 可用于构造 CPA 安全的对称加密方案 $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$，其中 CPA 是

选择明文攻击（参见 [30]）。具体而言，密钥生成算法 Gen 接收安全参数 $\lambda \in \mathbb{N}$ 作为输入，并返回一个随机种子 s 作为密钥。加密算法 Enc 在输入种子 s 和消息 m 后，均匀采样一个初始化向量 IV，并输出密文 $c = (\mathrm{IV}\ m + \mathrm{str}(|m|))$，其中 $\mathrm{str}(|m|)$ 表示 $\mathcal{C}$ 在输入 s 和 IV 时生成的密钥流，并在 $|m|$ 个元素后截断（此处 m 是由有限域 $\mathbb{F}$中的元素组成的字符串，$|m|$ 表示其长度）。然后，通过减去密钥流 $\mathrm{str}(|m|)$ 来执行解密。

选择明文攻击安全性通过一个安全游戏来评估，在该游戏中，攻击者可以访问加密预言机。因此，攻击者可以获得数量为多项式级（关于安全参数 λ）的字符串 $(\mathrm{IV}, \mathrm{str}(|m|))$，其中种子 s 固定，而 IV 和 m 可变。随后，攻击者可以利用这些信息发起密钥恢复攻击，试图推断有关秘密种子 s的信息。

因此，如果 $\mathcal{C}$ 是差分流密码，攻击者可以获得某个字符串 $(b(u) \ldots, b(u + h - 1))$，其中 $h \in \mathbb{N}$是关于 λ的多项式，b是给定的系统 (7) 的一个 $\mathbb{F}$-解 $(a_1, \ldots, a_n)$ 的密钥流。这意味着已知理想的生成元

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

我们将 $V_{\mathbb{F}}(J_{u,\ h})$ 表示为理想 $J_{u,\ h}$ 的生成元的同时 $\mathbb{F}$-解集合。给定子集 $S = \{x_1(r_1) - f_1, \ldots, x_n(r_n) - f_n\} \subset R$ 以及理想

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

我们考虑 $V_{\mathbb{F}}(I + J_{u,h}) = V_{\mathbb{F}}(I) \cap V_{\mathbb{F}}(J_{u,h})$，其中 $V_{\mathbb{F}}(I)$ 是差分系统 (7) 的所有 $\mathbb{F}$-解的集合。由于函数 b 是给定 $\mathbb{F}$-解 $(a_1, \ldots, a_n) \in V_{\mathbb{F}}(I)$ 的密钥流，因此 $(a_1, \ldots, a_n) \in V_{\mathbb{F}}(I + J_{u,h}) \neq \emptyset$。事实上，对于实际的流密码，有 $\#(V_{\mathbb{F}}(I + J_{u,h})) = 1$，其中 $h \in \mathbb{N}$ 足够大。换言之，(7) 有唯一一个与足够长的密钥流相容的解。

记 $V_{\mathbb{F}}(I + J_{u,h}) \subset \mathbb{F}^r$ 为 $\mathbb{F}$-解 $(a_1, \ldots, a_n) \in V_{\mathbb{F}}(I + J_{u,h})$ 的初始状态集合。以下结果本质上是命题 2.6 的推论。

定理 6.1。 [32, 定理 5。4] 令 $\mathrm{T}\colon\ R \to R$ 为代数自同态，对所有 $1 \le i \le n$ 定义如下

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

此外，定义理想

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

那么，有 $V_{\mathbb{F}}(I + J_{u,\ h}) = V_{\mathbb{F}}(J'_{u,\ h})$。

在一个合理的假设下，即对于足够多的 h 个密钥流元素，存在唯一一个与它们相容的 $\mathbb{F}$-解 $(a_1, \ldots, a_n)$，任何

旨在确定 $(a_1, \ldots, a_n)$ 的初始状态，即 $V_{\mathbb{F}}(J'_{u,\ h})$ 的攻击，被称为代数（密钥恢复）攻击。注意，存在伪解的情况，即 $\#V_{\mathbb{F}}(J'_{u,\ h}) > 1$，可以通过计算 Gr¨obner 基来识别，具体而言是通过线性维数

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

其中，$L \subset R$ 是对应于多项式代数 R 的域方程理想。实际上，回想一下，$V_{\mathbb{F}}(J'_{u,\ h}) = V(J'_{u,\ h} + L)$。要计算这个解集，可以根据命题 2.2 使用 Gr¨obner 基，或者使用其他类型的求解器。不过，如果更新多项式 f_i 是非线性的，那么当时钟 t取值较大时，多项式 $\bar{\mathrm{T}}^t(g)$ 可能具有很高的次数。由于在 $J'_{u,\ h}$ 的定义中存在 $t \geq u$，因此，当流密码 $\mathcal{C}$ 的预热更新数量 u 较大时，就会出现这种情况，而这通常正是如此。高次数必然会影响 $J'_{u,\ h} + L$ 的 Gr¨obner 基计算，因此，较长的预热阶段也是一种针对代数攻击的良好安全策略。

注意，初始状态通常还包含种子、初始向量以及一些常量元素，因此，攻击初始状态似乎是可取的，因为其中实际未知的项少于寄存器的总长度。然而，对于特殊类别的可逆差分流密码，任意内部状态都能唯一确定初始状态，因此，攻击密钥流开始输出时的时钟 u 处的状态更为方便。实际上，在这种情况下，我们可以假设 $u = 0$ 并计算 $V_{\mathbb{F}}(J'_{0,\ h})$，其中 $J'_{0,\ h}$ 的生成元次数远低于 $J'_{u,\ h}$的生成元次数。当然，在这种情况下，所考虑初始状态的所有项都是完全未知的。

定义 6.2. 如果定义 6.1 中的代数自同态 T 实际上是一个自同构，则称显式差分方程组 (7) 是可逆的。相应地，由显式差分方程组定义的差分流密码 $\mathcal{C}$ 称为可逆差分流密码。

注意，代数自同态 T: $R \to R$ 对应一个多项式映射 T: $\mathbb{F}^r \to \mathbb{F}^r$ ($r = r_1 + \ldots + r_n$)。若 $v(t) \in \mathbb{F}^r$ 是方程组 (7) 的一个 t-状态解（$a_1, \ldots, a_n$），则对于所有时钟 $t \geq 0$，都有 $\mathrm{T}(v(t)) = v(t+1)$。我们称 T 为显式差分系统 (7) 的状态转移自同态，称 T 为其状态转移映射。若 T 是自同构，则映射 T 也是可逆的，并且有 $\mathrm{T}^{-1}(v(t+1)) = v(t)$。这意味着，对于可逆差分流密码，我们可以从任意内部状态 $v(t) = \mathrm{T}^t(v(0))$ 唯一地求出初始状态 $v(0)$。

要证明代数自同构的可逆性并计算其逆映射，可以应用以下基于格罗布纳基理论的一般结果。关于多项式代数自同构的全面参考资料，请参阅著作 [40]。

定理 6.3. 设 $\mathbb{K}$ 为任意域，$X = \{x_1, \ldots, x_r\}$ 和 $X' = \{x'_1, \ldots, x'_r\}$ 为两个不相交的变量集。定义多项式代数 $P = \mathbb{K}[X]$、$P' = \mathbb{K}[X']$ 和 $Q = \mathbb{K}[X \cup X']$。考虑代数自同态 $\varphi\colon P \to P$，使得 $x_1 \mapsto g_1, \ldots, x_r \mapsto g_r (g_i \in P)$，以及由集合 $\{x'_1 - g_1, \ldots, x'_r - g_r\}$ 生成的相应理想 $J \subset Q$。此外，在多项式代数 Q 上赋予乘积单项式序，使得 $X \succ X'$。那么，映射 φ 是 P 的自同构，当且仅当 J 的约化 Gr ¨obner 基具有如下形式：$\{x_1 - g'_1, \ldots, x_r - g'_r\}$

其中 $g'_i \in P'$，对所有 $1 \le i \le r$ 均成立。在此情况下，若 $\varphi'\colon P' \to P'$ 是满足 $x'_1 \mapsto g'_1, \ldots, x'_r \mapsto g'_r$ 的代数自同态，且 $\xi\colon P \to P'$ 是同构$x_1 \mapsto x'_1, \ldots, x_r \mapsto x'_r$，则有 $\xi\,\varphi^{-1} = \varphi'\,\xi$。

例如，流密码 Trivium满足可逆性，我们将通过对其进行代数攻击来测试算法 MultiSolve。请注意，可逆显式差分系统的逆系统这一概念可以自然地引入 [32, 定义 3.6]。此外，可逆方程组（具有独立子系统）可用于定义分组密码 [32, 定义 5.6]。

7. Trivium

流密码 Trivium 由 De Canni`ere 和 Preneel 于 2005年提出。该密码被提交至 eSTREAM竞赛，并入选最终组合。我们回顾一下，eSTREAM 是一个欧洲项目，旨在找出新型、安全且高效的流密码。尽管密码分析工作广泛，对 Trivium 已知的最佳攻击仍显著慢于穷举攻击。 Trivium 是一种差分流密码，其对应的显式差分系统仅由基域$\mathbb{F} = \mathrm{GF}(2)$上的三个二次方程组成，即

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

其密钥流多项式是如下齐次线性多项式

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

因此， t-状态是一个由 $288 = 93 + 84 + 111$ 比特组成的字符串，其中 $t \ge 0$ 任意。预热更新的次数为 $u = 4 \cdot 288 = 1152$。种子（即私钥）和初始化向量均为 80 比特字符串，合起来构成初始状态的 160 比特。剩余的 128 比特为常数。

若 $R = \mathbb{F}[x(0)\ldots, x(92)\ y(0)\ldots, y(83)\ z(0)\ldots, z(110)]$, 状态转移 自同态 $\mathrm{T}\colon R \to R$，则 Trivium 的状态转移如下

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

根据定理 6.3，我们得到 T 实际上是一个自同构，其逆映射 $\mathrm{T}^{-1}\colon \bar{R} \to \bar{R}$ 定义如下

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

流密码 Trivium 的可逆性因此使得我们能够对其 u 状态发起代数攻击，并使用算法 MultiSolve 对其进行了分析。事实上，变量数量过多（即 288 个），使得我们无法通过一次格罗布纳基计算求解相应的多项式系统。相反，我们的多步策略将这一求解任务划分为许多子问题，其数量可以轻松估算，并且每个子问题中的变量数量都可以任意限定。

8. A multistep attack on Trivium

本节主要向读者介绍第 4 节中 MultiSolve 复杂度分析的实际应用。为此，我们针对流密码 Trivium 实施了一种代数攻击，并能够估算其计算成本。这种密码分析是第 6 节所述的、针对可逆差分流密码内部状态的代数攻击的一个实例。回顾一下，对于 Trivium，我们考虑多项式代数 $R = \mathbb{F}[x(0)\ldots,x(92)\ y(0)\ldots,y(83)\ z(0)\ldots,z(110)]$ ($\mathbb{F} = \mathrm{GF}(2)$) 以及相应的域方程理想 $L \subset R$。我们将 MultiSolve 应用于理想 $J'_{0,\ h} + L \subset R$ 的生成集 H，其中

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

这里 $\bar{\mathrm{T}}: \bar{R} \to \bar{R}$ 是 Trivium 的状态转移自同构 (10)，$g \in R$ 是其密钥流多项式 (9)，而 $b(t)$ ($0 \le t < h$) 是我们假设在攻击中已知的那部分密钥流。攻击的目标是计算 $V_{\mathbb{F}}(J'_{0,\ h} + L)$，也就是 Trivium 在密钥流初始时钟时的内部状态（见第 6 节）。

在我们的攻击中，密钥流长度设为 $h = 240$，因为这是能够在我们计算的所有用于分析 MultiSolve 的多项式系统中至多提供一个解的最少数据量，同时也能使生成集 H 中的多项式次数相对较低 (≤ 5)。

一开始值得注意的观察是，65 个初始值 $x(0), y(0), z(0)$ 这些值成为独立的线性多项式，因此我们可以立即利用它们消去 66 个变量。我们考虑的剩余 222 个变量集合为

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

这个集合仍然过大，无法实际执行 GBElimLin，因此我们决定选定一个由 116 个变量组成的集合 V，逐步对其赋值。为定义集合 V，我们首先注意到，由于 Trivium 的定义方程，X' 中有三个子集，每个子集均含 73 个变量；对这些变量赋值后，待求解方程组中会出现许多线性多项式。这些子集如下：

$V_0 = \{x(0)\ y(0)\ z(0)\ x(3)\ y(3)\ z(3)\ x(6)\ y(6)\ z(6)\ x(9)\ y(9)\ z(9)\ x(12)\ y(12)\ z(12),x(15)\ y(15)\ z(15)\ x(18)\ y(18)\ z(18)\ x(21)\ y(21)\ z(21)\ x(24)\ y(24)\ z(24)\ x(27)\ y(27)\ z(27),x(30)\ y(30)\ z(30)\ x(33)\ y(33)\ z(33)\ x(36)\ y(36)\ z(36)\ x(39)\ y(39)\ z(39)\ x(42)\ y(42)\ z(42),x(45)\ y(45)\ x(48)\ y(48)\ x(51)\ y(51)\ x(54)\ y(54)\ x(57)\ y(57)\ x(60)\ y(60)\ x(63)\ y(63)\ x(66),y(66)\ x(69)\ y(69)\ x(72)\ y(72)\ x(75)\ y(75)\ x(78)\ y(78)\ x(81)\ x(84)\ x(87)\ x(90)\}$;

$V_1 = \{x(1)\ y(1)\ z(1)\ x(4)\ y(4)\ z(4)\ x(7)\ y(7)\ z(7)\ x(10)\ y(10)\ z(10)\ x(13)\ y(13)\ z(13),x(16)\ y(16)\ z(16)\ x(19)\ y(19)\ z(19)\ x(22)\ y(22)\ z(22)\ x(25)\ y(25)\ z(25)\ x(28)\ y(28)\ z(28),x(31)\ y(31)\ z(31)\ x(34)\ y(34)\ z(34)\ x(37)\ y(37)\ z(37)\ x(40)\ y(40)\ z(40)\ x(43)\ y(43)\ z(43),x(46)\ y(46)\ x(49)\ y(49)\ x(52)\ y(52)\ x(55)\ y(55)\ x(58)\ y(58)\ x(61)\ y(61)\ x(64)\ y(64)\ x(67),$

$y(67)\ x(70)\ y(70)\ x(73)\ y(73)\ x(76)\ y(76)\ x(79)\ y(79)\ x(82)\ x(85)\ x(88)\ x(91)\}$;

$V_2 = \{x(2)\ y(2)\ z(2)\ x(5)\ y(5)\ z(5)\ x(8)\ y(8)\ z(8)\ x(11)\ y(11)\ z(11)\ x(14)\ y(14)\ z(14),x$
$(17)\ y(17)\ z(17)\ x(20)\ y(20)\ z(20)\ x(23)\ y(23)\ z(23)\ x(26)\ y(26)\ z(26)\ x(29)\ y(29)\ z(29),x$
$(32)\ y(32)\ z(32)\ x(35)\ y(35)\ z(35)\ x(38)\ y(38)\ z(38)\ x(41)\ y(41)\ z(41)\ x(44)\ y(44)\ z(44),x$
$(47)\ y(47)\ x(50)\ y(50)\ x(53)\ y(53)\ x(56)\ y(56)\ x(59)\ y(59)\ x(62)\ y(62)\ x(65)\ y(65)\ x(68),y$
$(68)\ x(71)\ y(71)\ x(74)\ y(74)\ x(77)\ y(77)\ x(80)\ y(80)\ x(83)\ x(86)\ x(89)\ x(92)\};$

因此， V集合通过将上述集合之一与另一个由 43 个变量组成的子集结合而成。使用 GBElimLin，我们针对不同密钥和赋值构成的测试集，搜索 NRV 平均值最低的集合 V。一些实验表明，可以定义 $V = V_2 \cup V_0'$，其中

$$V_0' = \{x(3), y(3), z(3), x(6), y(6), z(6), x(9), y(9), z(9), x(12), y(12), z(12), x(15), y(15), z(15),$$
$$x(18), y(18), z(18), x(21), y(21), z(21), x(24), y(24), z(24), x(27), y(27), z(27), x(30), y(30), z(30),$$
$$x(33), y(33), z(33), x(36), y(36), z(36), x(39), y(39), z(39), x(42), y(42), z(42), y(45)\}.$$

对于我们的多步策略，我们首先对从集合 V_0' 中移除最后 10 个变量后得到的 106 个变量进行赋值，然后在每个步骤中再对一个变量赋值。我们用相应的已赋值变量数量来标记这些步骤，例如 k，因此在实验中固定 $106 \leq k \leq 116$。我们应用 GBElimLin，参数 D 设为最大输入次数，该次数通常等于 3。这样可使 GBElimLin的平均运行时间相当短，例如在我们的实验中， $k = 106$ 为 4.5 秒。

在测试过程中，我们在一台具有以下硬件配置的服务器上运行代码：

- CPU：2 x AMD EPYC(TM) 7742；
- CPU核心/线程数：2 x 64 核心/2 x 128 线程；
- 可达到的最高 CPU 频率：3.4GHz；
- L3缓存：256Mb；
- RAM：2 TB。

在这台服务器上，我们安装了 Linux 发行版 Ubuntu 22.04 LTS 作为操作系统，并安装了 Magma 版本 2.27——这是一款用于求解代数和数论问题的软件包 [10]。我们将 Magma 设置为在执行并行线性代数运算时使用 16 个线程。我们的主要测试集包含 2^{16} 项不同的测试，即针对从相同数量的随机初始状态开始生成的 2^4 条密钥流，对变量进行 2^{12} 次随机猜测。我们称这个集合为随机测试集。我们还使用一个包含 2^{16} 项不同正确猜测的测试集，这些猜测对应于 2^{16} 个随机初始状态，我们称之为正确测试集。请注意，这里的初始状态指的是 Trivium 在密钥流初始时钟时的内部状态。

算法的复杂度 MultiSolve严格取决于参数B，该参数定义了驯化猜测，也就是说，定义了实际计算的 Gr¨obner 基。在我们的实验中，我们固定 $32 \leq B \leq 38$。选择这一设置的一个原因是，我们发现 $p_{32}(116) = 0$，也就是说，对于 $B = 32$, $k'' = 116$ 是可能的最后一步，其中 116 是所考虑的最大步数。此外，我们发现，NRV $\leq B = 38$至多只需几分钟，就能通过 Magma 计算 GrobnerBasis ；对于规模相当大的测试集而言，这对我们来说是“合理的时间”。实际上，我们测试中通常遇到的求解次数至多为 5。当变量数大于 38 时，我们还观察到，使用参数“HFE”执行 GrobnerBasis 时会出现一些内部错误。事实上，我们在所有计算中都采用了这一选项，因为它通常能加快计算速度。这主要是因为，在该参数设置下， Magma 会对稠密 Macaulay 矩阵采用并行计算的 F4算法

。因此，为了确保计算完全成功且速度更快，我们也选择设置 $B \leq 38$。

用于计算 MultiSolve 复杂度的主要统计量是，在所考虑的区间内， B 和 k 对应的概率 $p_B(k)$。为此，我们实际上在服务器上运行了两个相互独立的随机测试集，没有发现所得两组统计量之间存在显著差异。请注意，这两个测试集各包含 2^{12} 次随机猜测，针对 2^4 个随机初始状态。

Table 1. 在我们的随机测试集上估计的概率 $p_B(k)$。

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

表 1 给出了 B每个取值对应的可能最终步骤 k'' ；这里考虑的是最坏情形，即与初始状态对应的正确猜测在这些步骤时变得受控。

Table 2. 在我们的随机测试集上估计的最终步骤 k'' （最坏情形）。

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

请注意，上述最后几步实质上给出了单步策略的复杂度 $C = 2^{k''}$ ，也就是标准的猜测并确定策略或混合策略。假设完整多步策略的第一步 k' 对所有B都设为 $k' = 106$ ，我们可以将其与算法 MultiSolve 的主要复杂度 C 进行比较，即

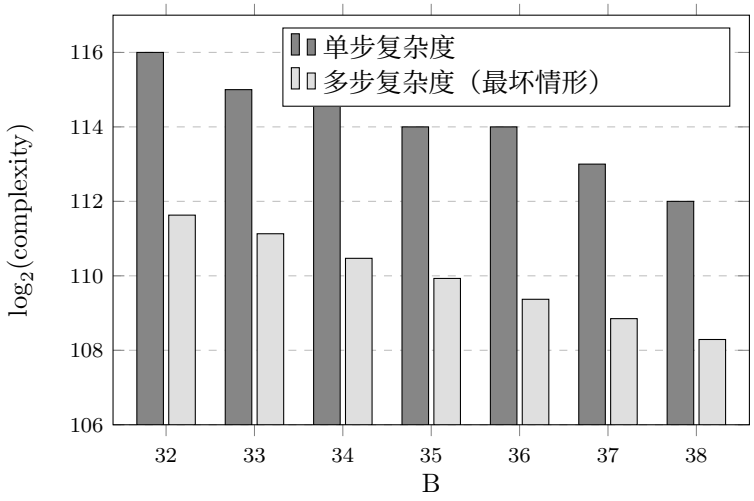
$$C_2 = \sum_{k' \leq k \leq k''} (p_B(k-1) - p_B(k))2^k.$$

回顾一下，按照约定 $p_B(k'-1) = 1$ ，定理 4.4 意味着 $C > C_2$。表 3 和图 1 对比了我们针对 Trivium 的代数攻击中一步策略与多步策略。

Table 3. 单步复杂度与多步复杂度（最坏情形）。

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

Figure 1. 单步复杂度与多步复杂度（最坏情形）



为了求得 MultiSolve 在平均情况下的复杂度，我们还需要第二项统计量，即正确测试集情况下的概率 $p_B(k)$。为避免混淆，我们将以此方式得到的概率记为 $\bar{p}_B(k)$。对于以下统计，我们使用两个相互独立的测试集，每个测试集包含 2^{16} 个不同的初始状态。

Table 4. 在正确的测试集上估算的概率 $\bar{p}_B(k)$。

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

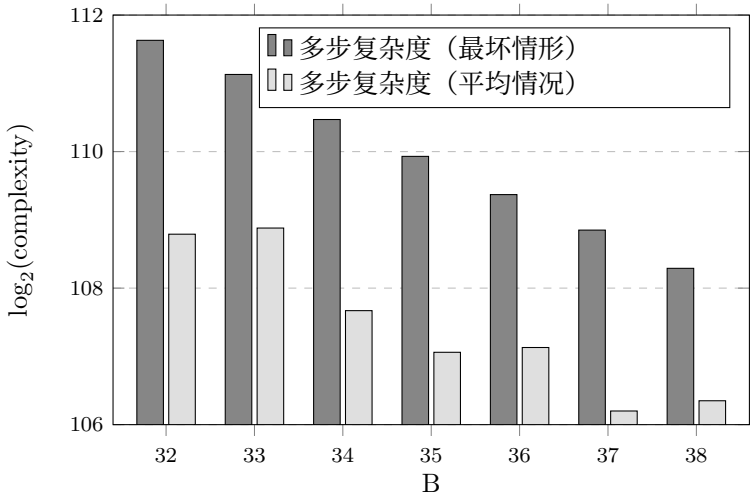
我们利用上述统计数据确定在哪一步 k'' 概率为 $\bar{p}_B(k'') < 0.5$，也就是说，至少一半的正确猜测会在某一步被驯服 $l \leq k'' < k''$。换言之，这表明算法 MultiSolve 以大于或等于 0.5 的概率在最终步骤 $l \leq k''$停止。

通过考察 k'' 和 l 作为该算法的最后几个步骤 MultiSolve，下面的表6和图2对最坏情况复杂度进行了比较

_TABLE 5. 在我们的正确测试集上估算的最终步骤数（平均情况）。

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

FIGURE 2. 多步复杂度（最坏情形）与多步复杂度（平均情况）的对比



C_2 与平均情形复杂度

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

TABLE 6. 多步复杂度（最坏情形）与多步复杂度（平均情形）。

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

¯我们强调，我们所达到的最佳复杂度的对数，即对于 $B = 37$，$\log_2(C_2) = 106.2$，非常接近我们为优化所选的已赋值变量最小数量 $k' = 106$ 。这是在 $p_{37}(106) = 0.5$ 的情况下实现的，也就是说，第 106 步中有一半的 Gr¨obner 基因未被计算，因为 NRV > 37。使用我们在 MAGMA 编写的代码并在我们的服务器上运行，对于这一平均情形复杂度，我们估计由 MULTISOLVE 对 TRIVIUM 发起代数攻击约需 2^{112} 秒。该估算按 $T = T_1 + T_2$ 得出（见公式 (6)），其中仅计算 Gr¨obner 基的运行时间为 $T_2 = 2^{111.5}$ 秒。这清楚地表明，在 GBELIMLIN 处调用 T_1 的运行时间对总时间无关紧要。

9. Conclusions and further directions

本文表明，无论在理论上还是实践中，多步策略都是估算并降低求解有限域上多项式方程组复杂度的可行方法。尽管整洁流密码 Trivium 在我们基于这一求解策略发起代数攻击后仍然安全，但与以往同类攻击相比，我们显著降低了所需复杂度 $2^{106.2}$。

事实上，所提方法的实验才刚刚开始。我们相信，进一步优化多步策略的多个参数，并随着求解算法不断进步，这项技术将成为评估密码系统安全等级的更有效工具。当然，多步策略是一种通用算法，在密码学领域之外也可能派上用场。

作为这条研究路线上的进一步方向，我们建议将剩余变量数量的限制推广为涉及其他关键指标的界限，这些指标可能表明多项式系统难以求解，例如方程的次数、方程的稠密度、求解次数的理论值等。当然，这些选择在很大程度上取决于所要使用的多项式系统求解器，因为多步策略可以采用其中任何一种。

总之，我们认为，有限域上多项式系统复杂度的统计赋值这一概念前景广阔，值得进一步研究。

10. Acknowledgements

首先，我们谨向 Lorenzo D'Ambrosio 致谢，感谢他富有成效的讨论和建议。此外，我们衷心感谢ReCaS-Bari （巴里大学、INFN）提供的高性能计算资源。

我们谨向匿名审稿人致以诚挚谢意，感谢他们对稿件进行了全面细致的审阅。我们感谢他们提出的所有真知灼见和建议，这些意见显著提升了论文的整体可读性。

References

[1] Armknecht, F., Ars, G., 基于 Gr¨obner 基的流密码代数攻击。Gr¨obner 基、编码与密码学。施普林格，柏林，2009。[2] Adams, W.W.; Loustaunau, P., Gr¨obner 基导论。数学研究生研究丛书，3。美国数学会，普罗维登斯，罗得岛州，1994。[3] Amadori, A.; Pintore, F.; Sala, M., 素域椭圆曲线离散对数问题。《有限域及其应用》，51 (2018)，168–182。[4] Bard, G.V., 代数密码分析。施普林格，Dordrecht，2009。[5] Bardet, M.; Faug`ere, J.-C.; Salvy, B.; Spaenlehauer, P.-J., 求解二次布尔系统的复杂度。《复杂性杂志》29 (2013)，第 1 期，53–75。[6] Bardet, M.; Faug`ere, J.-C.; Salvy, B.; Yang, B.-Y., 半正则多项式系统正则度的渐近行为。《代数几何中的有效方法》- MEGA 2005，1–14，2005年。[7] Bettale, L.; Faug`ere, J.-C.; Perret, L., 求解有限域上多变量系统的混合方法。《数学密码学杂志》3 (2009)，第 3 期，177–197。[8] Bettale, L.; Faug`ere, J.-C., Perret, L., 求解有限域上的多项式系统：混合方法的改进分析。国际符号与代数计算研讨会 - ISSAC 2012，67–74，2012。

[9] Borghoff, J.; Knudsen, L.R.; Matusiewicz, K., 爬山算法与 Trivium。密码学选题。SAC 2010，《计算机科学讲义》，6544。施普林格，柏林，2011。[10] Bosma, W.; Cannon, J.J.; Fieker, C.; Steel, A.（编），Magma 函数手册，第 2.27 版（2022），6359 页。[11] Caminata, A.; Gorla E., 求解多元多项式系统及交换代数中的一个不变量。《有限域算术》，《计算机科学讲义》，12542，3 2021 –36，施普林格，尚贝里，。[12] Cianfriglia, M.; Onofri, E.; Onofri, S.; Pedicini, M., 立方攻击十四年。《应用代数、工程、通信与计算》，2023，41 页。[13] Courtois, N.; Bard, G.V., 数据加密标准的代数密码分析。《密码学与编码》，《计算机科学讲义》，4887，152–169，施普林格，柏林，2007。[14] Courtois, N.; Klimov, A.; Patarin, J.; Shamir, A., 求解超定多元多项式方程组的高效算法。《密码学进展》- EUROCRYPT 2000，《计算机科学讲义》，1807，392–407，施普林格，柏林，2000。[15] De Canni`ere, C.; Preneel, B., Trivium 规范。eSTREAM - ECRYPT 流密码项目，https://www.ecrypt.eu.org/stream/triviumpf.html。[16] Dinur, I.; Shamir, A., 对可调黑盒多项式的立方攻击。《密码学进展》- EUROCRYPT 2009，《计算机科学讲义》，5479，278–299，施普林格，柏林，2009。[17] Dub`e, T.W., 多项式理想与 Gr¨obner 基的结构，SIAM 计算杂志，19 (1990)，750–773。[18] Faug`ere, J.C., 一种计算 Gr¨obner 基（F4）的新型高效算法。《代数几何中的有效方法》- MEGA 1998。《纯粹与应用代数杂志》，139 (1999)，第 1-3 期，61–88。[19] Faug`ere, J.C., 一种无需化简为零即可计算 Gr¨obner 基的新型高效算法（F5 2002，75 2002）。国际符号与代数计算研讨会 - ISSAC –83，ACM，纽约，。[20] Faug`ere, J. C.; Gianni, P.; Lazard, D.; Mora, T., 通过序变换高效计算零维 Gr¨obner 基。《符号计算杂志》16 (1993)，329–344。[21] Faug`ere, J.-C.; Joux, A., 使用 Gr¨obner 基对隐藏域方程（HFE）密码系统进行代数密码分析。《密码学进展》- CRYPTO 2003，《计算机科学讲义》，2729，44–60，施普林格，Belin，2003。[22] Gao, X.-S.; Huang, Z., 有限域方程求解的特征集算法。《符号计算杂志》47 (2012)，第 6 期，655–679。[23] Garey, M.-R.; Johnson, D.-S., 计算机与难解性：NP 完全性理论指南。Freeman，1979。[24] Gerdt, V.P.; Blinkov, Y.A., 多项式理想的对合基。用于应用问题的代数方程组与微分方程组的化简。《数学与计算机模拟》45 (1998)，第 5-6 期，519–541。[25] Gerdt, V.; La Scala, R., 偏差分多项式代数的诺特商与对称理想的 Gr¨obner 基。《代数杂志》423 (2015)，1233–1261。[26] Ghorpade, S.R., 有限域上零点定理的一个注记。《代数与代数几何研究》，23–32，《当代数学》，738，美国数学会，普罗维登斯，罗得岛州，2019。[27] Hashemi, A.; Lazard, D., 零维 Gr¨obner 基与多项式系统求解的更精确复杂度界，《国际代数与计算杂志》21 (2011)，703–713。[28] He, J.; Hu, K.; Preneel, B.; Wang, M., 扩展立方攻击：改进的大型超多项式恢复方法。《密码学进展》- ASIACRYPT 2022。《计算机科学讲义》，13794。施普林格，尚贝里，2022。[29] Huang, Z.; Lin, D., 使用特征集方法攻击 Bivium 和 Trivium。《密码学进展》- AFRICACRYPT 2011，《计算机科学讲义》，6737，77–91，施普林格，柏林，2011。[30] Katz, J.; Lindell, Y., 现代密码学导论。第二版。Chapman & Hall/CRC 密码学与网络安全。CRC 出版社，博卡拉顿，佛罗里达州，2015。[31] La Scala, R.; Polese S.; Tiwari S.K.; Visconti A., 对蓝牙 E0 流密码的代数攻击。《有限域及其应用》84 (2022)，论文编号 102102，29 页。[32] La Scala, R.; Tiwari S.K., 流密码/分组密码、差分方程与代数攻击。《符号计算杂志》109 (2022)，177–198。

[33] Marek, V.W.，《可满足性数学导论》，Chapman & Hall/CRC 信息学研究丛书。CRC 出版社，博卡拉顿，佛罗里达州，2009年。[34] Mascia, C.; Piccione, E.; Sala, M.，《针对流密码的代数攻击及其在非线性滤波生成器和 WG 伪随机数生成器中的应用》。arXiv:2112.12268，2022年，16页，即将在《Adv. Math. Commun.》发表。[35] Maximov, A.; Biryukov, A.，《对 Trivium 的两种平凡攻击》。密码学精选领域——SAC 2007。计算机科学讲义，4876。施普林格，柏林，2007。[36] Orsini, E.; Sala, M.，《通过综合征簇纠正错误和擦除》。纯粹与应用代数杂志 200 (2005年)，第 1-2 期，191–226 页。[37] Raddum, H.，《关于 Trivium 的密码分析结果》。eSTREAM——ECRYPT 流密码项目，报告 2006/039（2006）。[38] Rajchel-Mieldzio´c, G.，《量子映射与设计》，博士论文，波兰科学院理论物理中心，2022。[39] Ramos-Calderer, S.; Bravo-Prieto, C.; Lin, R.; Bellini, E.; Manzano, M., Aaraj, N., Latorre, J.I.，《使用量子退火求解布尔多元方程组》，《物理评论研究》4(1)，2022。[40] van den Essen, A.，《多项式自同构与雅可比猜想》。《数学进展》，190。Birkh¨auser 出版社，巴塞尔，2000。[41] Wu, W.T.，《关于初等几何中的判定问题与定理证明机械化》，《中国科学》21（1978年），第 2 期，159–172 页。另见：自动定理证明（丹佛，科罗拉多州，1983），213–234 页，《当代数学》，29，美国数学会，普罗维登斯，罗得岛州，1984。

* Dipartimento di Matematica, Università degli Studi di Bari “Aldo Moro”, ViaOrabona 4, 70125 Bari, Italy*Email address*: roberto.lascala@uniba.it

** Dipartimento di Matematica, Università degli Studi di Trento, Via Sommarive 14, 38123 Povo, Italy*Email address*: federico.pintore@unitn.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