

Provided for non-commercial research and educational use. Not for reproduction, distribution or commercial use.
--

Serdica

Mathematical Journal

Сердика

Математическо списание

The attached copy is furnished for non-commercial research and education use only.

Authors are permitted to post this version of the article to their personal websites or institutional repositories and to share with other researchers in the form of electronic reprints.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to third party websites are prohibited.

For further information on
Serdica Mathematical Journal
which is the new series of
Serdica Bulgaricae Mathematicae Publicationes
visit the website of the journal <http://serdica.math.bas.bg>
or contact: Editorial Office
Serdica Mathematical Journal
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Telephone: (+359-2)9792818
e-mail: serdica@math.bas.bg

ALGEBRAIC CRYPTANALYSIS OF SYMMETRIC CIPHERS: MODELING AND MULTISTEP SOLVING STRATEGIES

Roberto La Scala, Sharwan K. Tiwari

Communicated by A. Giambruno

ABSTRACT. This paper presents a systematic study of algebraic cryptanalysis for symmetric ciphers. It explains how encryption algorithms can be expressed as systems of polynomial equations over finite fields and analyzed using algebraic and logical solvers. The study begins by detailing block and stream ciphers through the lens of difference algebra, interpreting state updates and key additions as polynomial mappings. It then outlines the main solving strategies used in algebraic cryptanalysis, starting from Gröbner-basis and SAT approaches and progressing to hybrid solving. It next focuses on multistep and oracle-based strategies, which adjust variable guessing according to the hardness of each instance. These strategies extend classical hybrid attacks and make the solving process more efficient. Four case studies on Bluetooth E0, BIVIAM, TRIVIAM, and ARADI illustrate these ideas. Each example shows how algebraic modeling and adaptive solving reveal structural properties and solvability thresholds intrinsic to the considered cipher. The results show that multistep reasoning reduces computational effort compared with the static hybrid solving and provides a unified way to understand algebraic complexity in both stream and block ciphers.

<https://doi.org/10.55630/serdica.2025.51.419-442>

2020 *Mathematics Subject Classification*: Primary: 94A60; Secondary: 68W30, 13P10.

Key words: Algebraic cryptanalysis, Gröbner bases, hybrid and multistep strategies, stream and block ciphers.

1. Introduction. In the seminal work *Communication Theory of Secrecy Systems*, Shannon observed that breaking a well-designed encryption scheme should require an effort comparable to solving a large system of complex, nonlinear multivariate polynomial equations [31]. This idea remains one of the central principles in cryptosystem design.

From a mathematical viewpoint, solving systems of multivariate polynomial equations is among the most difficult computational problems. Given polynomials $f_1, \dots, f_m$ in n variables over a finite field $\text{GF}(q)$, the task is to find all common solutions of

$$f_1(x_1, \dots, x_n) = f_2(x_1, \dots, x_n) = \dots = f_m(x_1, \dots, x_n) = 0.$$

This multivariate polynomial (MP) problem is NP-hard [18]. When $q = 2$, it coincides with the Boolean satisfiability (SAT) problem. The complexity of generic algorithms such as exhaustive search grow exponentially in n , and even advanced symbolic methods such as Faugère’s F4 and F5 algorithms [16, 17] or XL and ElimLin procedures [11, 12, 13] also have exponential complexity in the general case. Hence, Shannon’s link between cryptographic strength and the hardness of nonlinear solving remains valid. However, polynomial systems derived from cryptographic constructions are highly structured rather than random. Algebraic cryptanalysis exploits these structures to simplify the solving process.

In algebraic cryptanalysis, each internal operation of a cipher such as substitution (confusion), diffusion, key addition, or feedback is represented by polynomial equations over $\text{GF}(2)$ or a small extension field. An expanded cipher therefore corresponds to a structured system that connects plaintext, ciphertext (or keystream bits), internal state variables, and key bits. The aim of algebraic cryptanalysis is not to solve arbitrary systems, but to use algebraic structure to identify efficient solving strategies and to estimate the computational cost [2, 12].

Several families of algorithms are used to attack such systems. Symbolic methods, including Gröbner bases [1] and linearization techniques [11], exploit algebraic structure and can detect contradictions at low degree. Logical approaches convert the equations into conjunctive normal form and use SAT or SMT solvers equipped with conflict learning and branching heuristics [27, 33]. Optimization-oriented approaches, such as quantum annealing, have also been explored for solving multivariate polynomial systems over finite fields [28]. Each approach has distinct advantages: algebraic methods perform well for inconsistent systems, SAT solvers are effective when solutions exist, and optimization-based solvers are promising for large, low-degree systems.

A practical improvement over direct solving is the *hybrid* or *guess-and-determine* strategy. Here, a subset of variables is fixed by exhaustive guessing,

and each reduced system is solved using an algebraic or logical solver. If the average time to solve one instance is τ and k variables are fixed, the expected cost is about $q^k \tau$. Hybrid methods decompose a large system into many smaller ones and work well when most partial assignments can be discarded quickly.

In the static hybrid approach the number of guessed variables k is fixed in advance. In practice, solvability varies across instances: some become easy with few guesses, while others remain difficult. This motivates *multistep strategies*, which adjust the number of guessed variables dynamically [24]. Starting from a small initial set, the algorithm alternates between partial evaluation and lightweight algebraic preprocessing, for example truncated Gröbner computation and low-degree eliminations. If the resulting system appears solvable, a full solve is attempted; otherwise, further variables are fixed and the process continues. This adaptive approach reduces the number of full solves by identifying simpler instances early.

A further generalization introduces *oracle functions*, which automatically decide whether the current system should be solved or further specialized. An oracle can follow an analytic rule or be learned empirically. This view unifies classical hybrid and multistep methods under a single principle and provides systematic control over branching depth and solving effort [25].

The multistep and oracle-based approaches are both practical and insightful. Practically, they can outperform static hybrid attacks by exploiting non-uniform solvability among instances. Analytically, they reveal thresholds defined by the number of guessed bits or the solving degree, beyond which polynomial systems shift from difficult to tractable. Such thresholds offer a concrete measure of a cipher's algebraic resistance and often provide more precise insights than asymptotic estimates.

Many symmetric ciphers, both stream and block, can be expressed as systems of algebraic difference equations. In this form, the internal state evolves over discrete time through a polynomial update rule, and the output is given by another map. Difference algebra [10, 29] provides a natural framework for this description and is closely related to the theory of linear and nonlinear feedback shift registers, which have long been used in pseudorandom generators. This unified algebraic perspective allows stream and block ciphers to be modeled and analyzed using the same mathematical framework [22].

In this manuscript we present algebraic cryptanalysis as a general methodology for the study of symmetric-key primitives. We emphasize the role of algebraic modeling, the trade-offs among solver families, and the transition from static hybrid attacks to adaptive multistep and oracle-based solving strategies.

Through representative examples on stream and block ciphers, we illustrate how these approaches operate in practice and how the resulting solvability patterns help characterize the algebraic resistance of modern cipher designs.

2. Preliminaries and notation. Let $\text{GF}(q)$ be the finite field with q elements, and let $\text{GF}(2)$ denote the binary field used in most symmetric-key constructions. We describe the multistep and hybrid oracle framework in the general setting of $\text{GF}(q)$, since the theory applies to arbitrary finite fields, but all case studies here arise from symmetric key ciphers over the Boolean field $\text{GF}(2)$. For any positive integer n , we work in the polynomial ring

$$\text{GF}(q)[x_1, \dots, x_n],$$

equipped with the degree-reverse-lexicographic (degrevlex) monomial order. Every Boolean function has a unique *algebraic normal form* (ANF), which is a sum of monomials with coefficients in $\text{GF}(2)$. Addition in $\text{GF}(2)$ corresponds to XOR, and multiplication corresponds to logical AND.

Given polynomials $f_1, \dots, f_m \in \text{GF}(q)[x_1, \dots, x_n]$, the ideal they generate is denoted by $I = \langle f_1, \dots, f_m \rangle$, and its set of common zeros in $\text{GF}(q)^n$ is

$$V(I) = \{x \in \text{GF}(q)^n : f_i(x) = 0 \text{ for all } i\}.$$

To restrict all solutions to the base field, the ideal is usually enlarged by the field equations

$$x_i^q - x_i = 0, \text{ for all variables } x_i \text{ (and } x_i^2 - x = 0 \text{ when } q = 2).$$

A *Gröbner basis* of I with respect to a chosen monomial order is a generating set whose leading terms generate the leading-term ideal [1]. Algorithms such as Buchberger's method or Faugère's F4 and F5 compute Gröbner bases by systematically combining and reducing polynomial multiples. In the matrix-based implementations (F4/F5), these reductions are performed through Gaussian elimination on large sparse matrices known as *Macaulay matrices*, whose rows correspond to multiples of the input polynomials up to a certain total degree. The highest degree reached during this process is called the *solving degree* d^* . It determines the size of the largest Macaulay matrix and therefore provides a practical measure of algebraic complexity. For systems that come from symmetric ciphers, d^* is sometimes small, which explains why Gröbner basis attacks can be efficient [2, 16, 17].

Because symmetric ciphers update an internal state over rounds or clock cycles, it is convenient to index the state variables by discrete time and describe

the design as an algebraic recursion. This view belongs to *difference algebra* [10, 29], the discrete analogue of differential algebra, and it is closely related to the study of linear and nonlinear feedback shift registers. We represent the time-indexed state as

$$X(t) = (x_1(t), \dots, x_n(t)) \in \text{GF}(2)^n,$$

and its evolution is described by polynomial maps

$$X(t+1) = F(X(t), K), \quad Y(t) = G(X(t), K),$$

where $K = (k_1, \dots, k_m)$ denotes the key variables, $Y(t)$ represents the round or keystream output, and F, G are polynomial functions over $\text{GF}(2)$. The shift operator σ acts by $\sigma(x_i(t)) = x_i(t+1)$, and repeated application of σ unfolds multiple rounds or clock steps. This representation, originally developed in [22], provides a unified algebraic model for both stream and block ciphers. A detailed presentation of this difference algebra based modeling approach appears in Section 5 of [22].

We adopt the following modeling conventions. All equations are expressed in the Boolean polynomial ring

$$\text{GF}(2)[x_1, \dots, x_n, k_1, \dots, k_m],$$

where x_i are the state variables and k_j are the key variables. The plaintext or initialization vector P and the ciphertext or keystream C are treated as known constants in a known-plaintext or known-keystream setting. The field equations $x_i^2 - x_i = 0$ are always included to ensure that all solutions are binary.

The notation and framework introduced here will be used in the following sections. The next section describes how symmetric ciphers are represented within this framework through algebraic models, outlining the main components such as linear layers, key additions, diffusion steps, and nonlinear operations (e.g., S-boxes and feedback functions), which later serve as inputs for Gröbner basis, SAT/SMT, hybrid, and multistep solving strategies.

3. Algebraic modeling of symmetric ciphers. The algebraic description of a cipher formalizes its internal operation as a system of polynomial equations over a finite field, typically $\text{GF}(2)$ or a small extension field in practical designs. Once each component is expressed in algebraic form, the concatenation of rounds or clock steps produces a structured system of polynomial equations that link plaintext, ciphertext, internal state, and key variables. The quality of this algebraic model largely determines the efficiency of the subsequent solving

process for the cryptanalytic purpose. From a cryptanalytic viewpoint, these models are built in order to recover the secret key or internal state from observed plaintext-ciphertext pairs or keystream bits, so the difficulty of solving the resulting polynomial system directly measures the resistance of the cipher to algebraic attacks.

At a high level, algebraic modeling proceeds through a sequence of steps. Starting from the cipher specification, one identifies all state and key variables, represents each operation as a polynomial mapping, and collects the resulting relations into a single polynomial (sparse) system. Building on the notation introduced in Section 2, the equations are written in the Boolean ring

$$R = \text{GF}(2)[P, X^{(0)}, X^{(1)}, \dots, X^{(r)}, K, C],$$

where P is the known plaintext or initialization vector, C is the known ciphertext or keystream bits, K is the set of key variables, and $X^{(t)} = \{x_1(t), \dots, x_n(t)\}$ represents the internal state after t rounds or clock cycles. Unrolling the cipher for r rounds creates a separate copy of the state variables for each round, and all of them are linked by the round equations that define the cipher's behaviour.

Modeling of the linear components is straightforward. Operations such as XOR, bit permutation, affine transformation, and key addition are expressed as linear equations over $\text{GF}(2)$. For example, a key addition step between state and key bits is modeled as

$$x_i^{(t+1)} = x_i^{(t)} + k_i,$$

and a linear diffusion or bit-permutation layer is represented as

$$X^{(t+1)} = A X^{(t)},$$

where A is a binary matrix describing the mixing pattern of the cipher. These equations directly capture the structure of linear and affine layers.

Modeling of nonlinear components such as substitution boxes (S-boxes) is a central aspect of algebraic representation. An S-box is defined by its truth table, from which one can derive either an explicit algebraic normal form (ANF) for each output bit or an implicit quadratic encoding that lowers the algebraic degree [2].

Example 1 (ARADI S-box). *The 4×4 ARADI S-box, defined by the output permutation*

$$(0, 1, 2, 3, 4, 13, 15, 6, 8, 11, 5, 14, 12, 7, 10, 9),$$

maps input bits (x_0, x_1, x_2, x_3) to output bits (y_0, y_1, y_2, y_3) . From its truth table, the explicit algebraic normal forms are

$$\begin{aligned} y_0 &= x_0x_2x_3 + x_0x_2 + x_0 + x_1x_2 + x_1x_3, & y_1 &= x_0x_2 + x_1, \\ y_2 &= x_0x_1x_2 + x_0x_2 + x_0x_3 + x_2, & y_3 &= x_0x_2 + x_1x_2 + x_3. \end{aligned}$$

This explicit representation has maximum algebraic degree 3. Alternatively, the same mapping can be expressed implicitly in variables x_i and y_i with 21 quadratic equations. This implicit encoding increases the number of equations but reduces the degree to 2, which can be advantageous for algebraic and hybrid solvers.

The next source of nonlinearity is modular addition. It is modeled by introducing carry variables that ensure bitwise consistency while keeping the algebraic degree low. For n -bit words

$$x = (x_{n-1}, \dots, x_0), \quad y = (y_{n-1}, \dots, y_0), \quad z = (x + y) \bmod 2^n,$$

the bitwise relations are

$$\begin{aligned} z_i &= x_i + y_i + c_i, \\ c_{i+1} &= x_iy_i + x_ic_i + y_ic_i, \quad i = 0, 1, \dots, n-2, \end{aligned}$$

where c_i are the carry bits with $c_0 = 0$. All operations are over $\text{GF}(2)$. This provides a compact quadratic encoding of modular addition suitable for algebraic solvers.

Another source of nonlinearity is feedback, where the next state depends on previous bits. In stream ciphers, such relations are modeled as

$$x_i^{(t+1)} = x_{i+3}^{(t)} + x_i^{(t)}x_{i+1}^{(t)},$$

with all operations over $\text{GF}(2)$. Bitwise Boolean operations such as AND and OR are represented directly as low-degree polynomials, for instance

$$x_i \text{ AND } x_j \Rightarrow x_ix_j, \quad x_i \text{ OR } x_j \Rightarrow x_i + x_j + x_ix_j.$$

These forms preserve the logical structure of feedback and mixing layers while keeping degree small.

Combining the modeled components, the round or clock transition of a cipher can be expressed compactly as

$$X(t+1) = F(X(t), K), \quad Y(t) = G(X(t), K),$$

where F and G are polynomial maps over $\text{GF}(2)$ describing the state update and output relation. This formulation corresponds to the algebraic difference representation introduced in Section 2 and detailed in [22]. The shift operator $\sigma(X(t)) = X(t+1)$ unfolds successive rounds or clock steps, linking all state copies through polynomial recursions. When unfolded for r rounds or clock steps, this recursion yields the system used in algebraic cryptanalysis, algebraically equivalent to the difference formulation since both define ideals with the same set of $\text{GF}(2)$ -solutions [22].

This representation provides a common algebraic framework for both stream and block ciphers. In stream ciphers, $Y(t)$ gives the keystream bits at each clock, while in block ciphers the unrolling of rounds serves the same purpose as discrete-time expansion. The resulting systems are sparse, low in degree, and often overdetermined, making them suitable for symbolic and hybrid elimination methods [22]. The formal connection with difference algebra [10, 19, 22, 26, 29] clarifies how polynomial recursions describe state evolution and output relations, even though dedicated solvers for difference systems are not yet available. Modeling choices such as explicit or implicit S-box encoding and carry-bit formulations for modular addition directly influence the number of variables, the algebraic degree, and thus the overall solving complexity.

In summary, algebraic modeling translates a cipher into a structured system of polynomial equations that accurately reproduces its round function. This formulation provides the mathematical foundation for the solver frameworks discussed in the following sections.

4. Solving strategies for polynomial systems. Once a cipher is represented as a system of polynomial equations, the central task in algebraic cryptanalysis is to solve this system as efficiently as possible. For general or random multivariate polynomial systems, the known algorithms grow exponentially with the number of variables [18]. In algebraic cryptanalysis, however, suitable modeling often yields structured systems of low algebraic degree and high sparsity, which can be tackled by dedicated solving approaches. The main families of such solving approaches are algebraic (symbolic), logical (satisfiability-based), and hybrid strategies.

Symbolic approaches work directly on polynomial equations through algebraic elimination. The best-known methods are those for computing Gröbner bases, beginning with Buchberger's algorithm and including Faugère's efficient F4 and F5 variants [16, 17]. These algorithms multiply and reduce input polynomials under a chosen monomial order and perform Gaussian elimination on the

resulting Macaulay matrices. Their cost is governed by the *solving degree* d^* , the highest total degree reached during the computation. For dense random systems d^* may grow quickly, but for structured systems derived from ciphers it usually stays low, which explains the efficiency of Gröbner basis attacks [2]. Alternative algebraic methods such as the XL (eXtended Linearization) algorithm and the ElimLin procedure [11, 12, 13] expand equations up to a bounded degree, linearize the resulting monomials, and then eliminate variables through Gaussian elimination.

Logical approaches translate polynomial systems into a Boolean satisfiability problem. Each equation over $\text{GF}(2)$ is written as a set of clauses in conjunctive normal form, and SAT or SMT solvers search for assignments that satisfy all clauses [27, 33]. These solvers use conflict-driven clause learning, restarts, and heuristic variable selection to explore the search space effectively. In the algebraic analysis of stream ciphers such as Bluetooth E0, SAT solvers and Binary Decision Diagrams (BDDs) have served as useful complements to algebraic methods in practical cryptanalysis [30, 33].

Hybrid or *guess-and-determine* strategies combine exhaustive guessing with algebraic solving [8, 11]. A subset of variables is fixed to specific values, and each reduced system is solved separately. If the average time to solve one instance is τ and k variables are fixed, the expected cost is about $q^k \tau$. This approach, introduced by Courtois et al. [11], balances algebraic and combinatorial effort. Hybrid strategies can be strengthened through partial elimination and incremental reduction, for example by computing truncated Gröbner bases at low degree to extract new linear relations and eliminate additional variables.

Optimization-based approaches, such as quantum annealing, have also been investigated for solving systems of Boolean equations [28]. Although these approaches are still experimental, they offer an alternative way of viewing polynomial solving in cryptanalysis and may inspire future algorithmic designs. Recent studies also explore using machine learning to guide variable selection, estimate solvability, or tune branching heuristics within algebraic and hybrid frameworks.

No single solving method performs best for all ciphers. Algebraic methods (Gröbner bases) are effective when the solving degree is low and few solutions correspond to the correct key or internal state (i.e., valid cryptanalytic solutions), while SAT solvers perform better when many assignments satisfy the equations. Hybrid and adaptive frameworks combine the strengths of both, balancing algebraic reasoning with search-based exploration. The next section introduces the adaptive multistep and oracle-based frameworks, which extend the hybrid idea by adjusting the level of specialization to the hardness of each instance.

5. Multistep and Oracle-based strategies. Hybrid solving can be made adaptive by varying the number of fixed variables according to feedback from the solver. In classical hybrids, this number k is chosen in advance, which may lead to excessive branching when k is too small or wasted computation when it is too large. The multistep strategy [24] formalizes this adaptive idea and provides a recursive framework that balances algebraic elimination with combinatorial branching.

Starting from an initial system $J = \{f_1, \dots, f_m\} \subset \text{GF}(q)[x_1, \dots, x_n]$, a small subset of variables is fixed and a truncated Gröbner basis computation is carried out up to a low degree d_1 . This preprocessing often reveals new linear relations that reduce the number of active variables. If the number of remaining variables after this elimination falls below a chosen bound B (a “tameness” threshold), the system is treated as small enough for a full solution attempt; otherwise, another variable is fixed and the process continues. The recursion forms a tree of partial evaluations whose depth equals the number of variable assignments. The total number of nodes in this tree gives a direct estimate of the overall complexity. A key result shown in [24] is that the multistep strategy with the maximum number of steps, where variables are added one by one, is optimal with respect to the total number of subsystems that must be solved, while the single-step hybrid is the least efficient case. This corresponds to the formal definition of the multistep strategy in Section 3 of [24] and to the optimality result stated in Theorem 4.4 of that paper.

Formally, let $R = \text{GF}(q)[x_1, \dots, x_n]$ and define the field-equation ideal

$$\mathcal{F} = \langle x_1^q - x_1, \dots, x_n^q - x_n \rangle.$$

The augmented system is $H = J + \mathcal{F}$. For a threshold degree d_1 , let $\text{GROBNER}(H, d_1)$ denote a truncated Gröbner basis computation that processes all *S-polynomials* of degree at most d_1 . If d_1 exceeds the true solving degree, $\text{GROBNER}(H, d_1)$ coincides with the full basis; otherwise it provides partial information, often revealing new linear relations. This low-degree elimination step, written as GBELIMLIN , returns a pair (L, G) , where L is the set of discovered linear equations and G the reduced residual system. When $\max\deg(G) \leq 1$, the system is already solved, making this step an inexpensive reduction in practice.

To attempt a bounded solve within a time limit, the routine GROBNERSAFE is used. Given a degree bound d_2 and timeout τ , run $\text{GROBNER}(H, d_2)$ within that limit. If the computation finishes and the result satisfies $\max\deg(G) \leq 1$, the case is declared **tame**; otherwise it is **wild**. The returned tag $\text{status} \in \{\text{tame}, \text{wild}\}$ records this result. Together, the two components GBELIMLIN and

GROBNERSAFE form the symbolic core of the multistep recursion. Intuitively, a case is **tame** when the reduced system is small enough (typically with at most B active variables) to be completely solved within the time bounds, whereas a **wild** case remains too large or too hard and therefore triggers further branching.

For branching, let $\text{GF}(q) = \{c_1, \dots, c_q\}$ and define, for $0 \leq k \leq n$ and $1 \leq i_1, \dots, i_k \leq q$,

$$E(i_1, \dots, i_k) = \{x_1 - c_{i_1}, \dots, x_k - c_{i_k}\}, \quad J(i_1, \dots, i_k) = J + \langle E(i_1, \dots, i_k) \rangle.$$

If a direct computation of $V_{\text{GF}(q)}(J(i_1, \dots, i_k))$ is infeasible within the time limit, the next variable is fixed and the recursion continues:

$$V_{\text{GF}(q)}(J(i_1, \dots, i_k)) = \bigcup_{i=1}^q V_{\text{GF}(q)}(J(i_1, \dots, i_k, i)).$$

This procedure defines a tree search where each level corresponds to one additional variable assignment.

The oracle-based formulation in [25] generalizes the multistep idea by introducing a decision function

$$\text{ORACLE}(G, k) \in \{\mathbf{tame}, \mathbf{wild}\},$$

which decides, at recursion depth k , whether to attempt a bounded solve or to continue branching. The oracle mechanism is introduced in Section 2 and 3 of [25]. The classical hybrid method is recovered by setting $\text{ORACLE}(G, k) = \mathbf{wild}$ for $k < B$ and **tame** otherwise. The oracle can depend on algebraic features of G after linear elimination such as the number of remaining variables, rank of the linear subsystem, or density of quadratic terms and may also incorporate data-driven heuristics learned from previous runs.

A depth-first implementation is convenient for both analysis and practice, because each recursive call either fixes one more variable or launches a bounded solver with degree d_2 and time limit τ . The conceptual structure of this recursion is summarized below.

Termination is always guaranteed, because once all n variables are fixed the augmented ideal becomes trivial. In practical cryptanalysis the recursion usually stops much earlier, at the point where the bounded solver first succeeds. This adaptive stopping rule is what gives the strategy its efficiency on cipher-derived systems.

The recursion defines a q -ary tree in which each internal node represents a branching decision and each leaf corresponds to a computed tame case or an

Algorithm 1 MULTISOLVE($i_1, \dots, i_k$) (depth-first)

```

1:  $(L, G) \leftarrow \text{GBELIMLIN}(H \cup E(i_1, \dots, i_k), d_1)$ 
2: if  $\text{maxdeg}(G) \leq 1$  then return  $L \cup G$ 
3: end if
4: if  $\text{ORACLE}(G, k) = \text{tame}$  then
5:    $(\text{status}, G) \leftarrow \text{GBSAFE}(G, d_2, \tau)$ 
6:   if  $\text{status} = \text{tame}$  and  $\text{maxdeg}(G) \leq 1$  then return  $L \cup G$ 
7:   end if
8: end if
9: for  $i = 1$  to  $q$  do
10:   $G' \leftarrow \text{MULTISOLVE}(i_1, \dots, i_k, i)$ 
11:  if  $\text{maxdeg}(G') = 1$  then return  $G'$ 
12:  end if
13: end for
14: return  $\{1\}$ 

```

immediate linear termination. If the tree has L leaves, the total and internal node counts are

$$N = \frac{qL - 1}{q - 1}, \quad M = \frac{L - 1}{q - 1},$$

so that for Boolean systems ($q = 2$) one has $N = 2L - 1$ and $M = L - 1$. An oracle is called *accurate* if no computed wild cases occur during the execution of MULTISOLVE. It is called *perfect* if, in addition, it minimizes the number of computed tame cases. In other words, an oracle is accurate if every predicted tame case corresponds to a computed tame case, and it is perfect if the prediction and computation always agree. When the oracle is accurate, the runtime is essentially proportional to L , since each leaf corresponds to one completed solve.

For an integer $B > 0$, the classical hybrid rule corresponds to the accurate choice

$$\text{ORACLEH}_B(G, k) = \begin{cases} \text{wild}, & k < B \text{ (guessed variables),} \\ \text{tame}, & \text{otherwise,} \end{cases}$$

which produces exactly q^B leaves. Here B plays the role of a depth bound on the number of guessed variables, the oracle never declares a case as **tame** (and thus never launches a bounded solve) before k reaches B . More generally, if the depths of computed tame cases lie between B and C , then ORACLEH with parameter C provides a conservative upper bound on both tame and wild counts for any oracle.

Two limit cases describe the overall behaviour. When $\text{ORACLET} \equiv \text{tame}$, the bounded solver is invoked at every node (always-try), giving N solver calls. With a perfect oracle, solver calls occur only at the leaves, giving L calls. Their ratio,

$$\frac{N}{L} = \frac{qL - 1}{L(q - 1)} = 1 + \frac{1}{q - 1} \left(1 - \frac{1}{L}\right),$$

shows that for $q = 2$ the overhead of the always-try rule is below a factor of two and decreases as L grows.

In practical calibration, one first probes a small sample with the always-try rule to estimate the typical depths of computed tame cases, then adopts the threshold oracle ORACLEH with parameter C for a conservative bound or with parameter B for a near-optimal trade-off [24, 25].

The multistep and oracle-based frameworks thus represent a shift from static to dynamic hybrid solving. They combine algebraic elimination and combinatorial branching within a unified recursive structure and provide a quantitative link between the algebraic properties of a system and its solving complexity. Empirical studies on modern stream and block ciphers such as TRIVIUM and ARADI [24, 25] show that the multistep approach can outperform the classical hybrid strategy, requiring fewer complete solving instances and revealing consistent ranges of variable guesses where bounded solving tends to succeed.

6. Case studies in algebraic cryptanalysis. This section illustrates the complete algebraic modeling and solving framework through four representative symmetric ciphers: Bluetooth E0, TRIVIUM , BIVIUUM , and ARADI . For each cipher this section describes the algebraic modelling, discusses how its structure affects the solving degree and guessing strategy, and outlines the corresponding hybrid or multistep attack procedure. Each subsection follows the same pattern. It first gives the algebraic recurrence, then describes the polynomial system and the solving method.

6.1. Algebraic Attack on E0. The Bluetooth stream cipher E0 consists of four independent linear feedback shift registers (LFSRs) of lengths 25, 31, 33, and 39, combined through a 4-bit finite-state machine (FSM) [4]. Let the register sequences be x , y , z , and u , and let the FSM delay elements be c and d . For $t \geq 0$, the LFSR recurrences are given by the linear difference equations over $\text{GF}(2)$

$$\begin{aligned} x(t + 25) &= x(t) + x(t + 5) + x(t + 13) + x(t + 17), \\ y(t + 31) &= y(t) + y(t + 7) + y(t + 15) + y(t + 19), \end{aligned}$$

$$\begin{aligned} z(t+33) &= z(t) + z(t+5) + z(t+9) + z(t+29), \\ u(t+39) &= u(t) + u(t+3) + u(t+11) + u(t+35). \end{aligned}$$

Let the LFSR outputs at clock t be $b_0(t)$, $b_1(t)$, $b_2(t)$, $b_3(t) = x(t+1)$, $y(t+7)$, $z(t+1)$, $u(t+7)$. The FSM stores $(d(t), c(t))$ and $(d(t+1), c(t+1))$, representing its lower and upper delay elements. At each clock, the lower delay element $(d(t), c(t))$ stores the previous upper value $(d(t+1), c(t+1))$, and the new upper element $(d(t+2), c(t+2))$ is computed from two nonlinear Boolean functions $g_0(t)$ and $g_1(t)$ written in algebraic normal form (ANF). Writing

$$\begin{aligned} \sigma_1(t) &= b_0(t) + b_1(t) + b_2(t) + b_3(t), & \sigma_2(t) &= \sum_{0 \leq i < j \leq 3} b_i(t)b_j(t), \\ \sigma_3(t) &= \sum_{0 \leq i < j < k \leq 3} b_i(t)b_j(t)b_k(t), & \sigma_4(t) &= b_0(t)b_1(t)b_2(t)b_3(t). \end{aligned}$$

the functions g_0 and g_1 are

$$\begin{aligned} g_0(t) &= \sigma_1(t)c(t+1) + \sigma_2(t) + c(t+1) + d(t+1) + c(t) + d(t), \\ g_1(t) &= \sigma_4(t) + d(t+1)\sigma_2(t) + c(t+1)d(t+1)\sigma_1(t) + c(t+1)\sigma_3(t) + d(t+1) + c(t). \end{aligned}$$

Hence, all nonlinearity in the ANF model is concentrated in g_0 and g_1 , while the LFSR recurrences remain linear. The keystream bit at clock t is

$$f(t) = x(t+1) + y(t+7) + z(t+1) + u(t+7) + c(t+1),$$

and f denotes the corresponding keystream polynomial.

According to the Bluetooth specification, the cipher is reinitialized after $T = 200$ clocks [4]. The explicit difference system describing E0 is invertible; the inverse system can be computed symbolically, for instance by a Gröbner basis computation. This property allows an attacker to target any internal state instead of only the initial one, and it is convenient to work with the state at the clock where the keystream output begins [23].

For algebraic cryptanalysis, the linear LFSR recurrences are used to eliminate linear variables, leaving only the nonlinear FSM equations and the keystream constraints. Fixing a clock bound B that produces a keystream of size K (in practice $K \in [51, 63]$ in experiments), the reduced polynomial system has the variable set

$$\{x(0:24), y(0:30), z(0:32), u(0:38), c(0:B), d(0:B)\}.$$

This system is sparse and has maximum algebraic degree 4.

The attack follows a guess-and-determine strategy that guesses 83 state variables, including a subset of 14 that serve as a powerful early filter for inconsistent instances. The detailed derivation and evaluation of this filter appear in Section 7 of [23]. Reference [23] develops a Gröbner basis attack on E0 using the difference stream cipher framework of [22] and a classical guess-and-determine strategy, without multistep or oracle-based solving. In the present paper it serves as a complementary case study, it illustrates how difference algebraic modeling and Grobner/SAT/BDD comparisons behave on a short keystream stream cipher, whereas the later works [24, 25] apply the multistep and oracle-based strategies to Trivium and Aradi. Each specialized instance is solved by computing a DeRevLex Gröbner basis of the system augmented with field equations, and the number of GF(2)-solutions is read from the quotient-algebra dimension. Empirically, once $K \geq 59$, most specialized instances are inconsistent (yielding degree-0 Gröbner bases), and at $K = 63$ the average number of spurious keys per guess is essentially zero. This behaviour helps explaining why Gröbner solving tends to be faster than SAT solving on these inconsistent low-degree systems.

Table 1. Solver comparison for the E0 short keystream attack. Each value shows the average per-instance time (milliseconds) over 2^{17} guesses and 8 keys on Server (Intel Core i9-10900 @ 2.80 GHz, 64 GB RAM, Debian)

K	GB avg (ms)	GB min/max (ms)	SAT avg (ms)	SAT min/max (ms)
51	31	1–411	196	105–1007
53	34	2–480	220	121–876
55	41	2–522	230	134–638
57	52	3–620	245	143–645
59	64	3–799	283	161–777
61	79	3–1115	300	174–732
63	96	4–1287	326	191–862

For $K = 60$, the average Gröbner runtime per guess is about 60 ms, giving a sequential cost of roughly 2^{79} seconds across the full guess space. All Gröbner runs with $K = 63$ remain below 0.5 GB of RAM. BDD based solvers show rapidly increasing memory use (tens to hundreds of gigabytes) and much longer runtimes as K increases, while Gröbner and SAT solvers (`slimgb` in SINGULAR [32], [33]) scale more smoothly. The total exhaustive complexity is approximately 2^{83} seconds, improving on earlier BDD based short keystream estimates of about 2^{86} [23, 21]. The approach is straightforward to parallelize across the

guess space.

6.2. Multistep algebraic attack on Trivium. TRIVIUM is a 288-bit stream cipher built from three nonlinear feedback shift registers (NLFSRs) of lengths 93, 84, and 111 [14], denoted

$$x = (x_0, \dots, x_{92}), \quad y = (y_0, \dots, y_{83}), \quad z = (z_0, \dots, z_{110}).$$

At clock t , the register contents are $x(i+t), y(i+t), z(i+t) \in \text{GF}(2)$. Each register is shifted by one position and updated through quadratic recurrences,

$$\begin{aligned} x(93+t) &= z(t) + x(24+t) + z(45+t) + z(1+t)z(2+t), \\ y(84+t) &= x(t) + y(6+t) + x(27+t) + x(1+t)x(2+t), \\ z(111+t) &= y(t) + y(15+t) + z(24+t) + y(1+t)y(2+t), \end{aligned}$$

where all additions are in $\text{GF}(2)$. The keystream bit at clock t is

$$f(t) = x(t) + x(27+t) + y(t) + y(15+t) + z(t) + z(45+t).$$

TRIVIUM is initialized with an 80-bit key and an 80-bit initialization vector (IV), while the remaining 128 bits are fixed constants. The cipher runs $4 \times 288 = 1152$ initialization clocks before producing keystream bits.

The above relations define an explicit difference-algebraic model:

$$X(t) = (x(t), y(t), z(t)), \quad X(t+1) = F(X(t)), \quad f(t) = G(X(t)).$$

Unrolling this model over T clocks gives a sparse quadratic system in $\text{GF}(2)[x, y, z]$ with about $288T$ variables. Although this number is large, the equations are very sparse, enabling effective elimination by truncated Gröbner passes or hybrid solving. In the experimental setup of [24], the system is built using $K = 240$ keystream bits, which is the smallest amount of data that yields a unique solution in all tests while keeping the solving degree at most 5. Among these equations, 66 are independent linear equations and can be used directly to remove 66 variables, leaving a reduced system of 222 unknowns before the MULTISOLVE procedure is applied.

A direct Gröbner or SAT solution of the full unrolled system is infeasible. The classical hybrid approach fixes k state variables and solves the remaining system completely. In contrast, the *multistep* strategy (MULTISOLVE) applies the guess-and-determine idea iteratively. After each truncated elimination phase GBELIMLIN(d_1), the oracle function ORACLE(G, k) evaluates the number of remaining variables (NRV). If this number falls below a threshold B , a bounded

Gröbner computation $\text{GROBNERSAFE}(d_2, \tau)$ is executed; otherwise, one more variable is fixed and the process continues recursively. This oracle-guided rule directly implements the theoretical control principle introduced in the previous section.

Formally, if $p_B(k)$ denotes the probability that the reduced system after k variable assignments still exceeds the bound B , the expected number of Gröbner computations in the multistep strategy is

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

where k' and k'' are the first and last iteration depths. The single-step hybrid complexity is $C = 2^{k''}$. Here C_2 represents the worst-case complexity of the multistep framework, while $\bar{C}_2$ denotes the average-case (empirical) complexity.

The threshold parameter $B \in [32, 38]$ controls the oracle decision based on the number of remaining variables (NRV). It is chosen according to the experimentally measured probability function $p_B(k)$, which gives the chance that the reduced system after k variable assignments still exceeds the bound B . For example, $p_{32}(116) \approx 0$ indicates that $k'' = 116$ is the last feasible depth for $B = 32$, while $p_B(k' - 1) = 1$ guarantees that all relevant branches are explored. Systems with $\text{NRV} \leq 38$ can be solved in MAGMA [9] within a few minutes using the “HFE” option (parallel F4 implementation). In all tests, the solving degree does not exceed 5.

Table 2. Estimated base-2 logarithmic complexities of the hybrid and multistep (MULTISOLVE) strategies for TRIVIUM. Each column corresponds to NRV bound $B \in \{32, \dots, 38\}$

Strategy	Metric	32	33	34	35	36	37	38
HYBRID	$\log_2(C)$	116	115	115	114	114	113	112
MULTISOLVE	$\log_2(C_2)$	111.63	111.13	110.47	109.93	109.37	108.85	108.29
MULTISOLVE	$\log_2(\bar{C}_2)$	108.79	108.88	107.67	107.06	107.13	106.20	106.35

For $B = 37$, the empirical complexity of MULTISOLVE is $\log_2(\bar{C}_2) \approx 106.2$, which corresponds to about $2^{106.2}$ Gröbner computations. The solving degree remains below 5 in all cases, showing that the multistep strategy effectively limits algebraic growth and keeps the structure low-degree even for large unrolled systems. Across all tested cases, the multistep method reduces the number of

Gröbner computations compared with the classical hybrid approach. For TRIVIUM, it decreases the expected solving cost from about 2^{115} to roughly $2^{106.2}$ without loss of correctness, demonstrating the advantage of iterative Gröbner based reasoning on large sparse Boolean systems. Detailed timing data and computed probabilities can be found in [24].

Algebraic attack on BIVIUM. BIVIUM is a two-register reduction of TRIVIUM obtained by removing one NLFSR. Its explicit difference model, with clock index t , is

$$\begin{aligned}x(93+t) &= y(t) + y(15+t) + x(24+t) + y(1+t)y(2+t), \\y(84+t) &= x(t) + y(6+t) + x(27+t) + x(1+t)x(2+t),\end{aligned}$$

and the keystream polynomial is

$$f(t) = x(t) + x(27+t) + y(t) + y(15+t).$$

Each state at time t contains $93+84 = 177$ bits, and the cipher performs $4 \times 177 = 708$ warm-up clocks before generating keystream.

In the experiments of [22], the algebraic system was built using a keystream window $K \in [180, 195]$ (best trade-off near $K \approx 190$). For $K = 180$, the system empirically admits a unique solution over $\text{GF}(2)$. By guessing 38 carefully chosen variables, about 61 additional variables become evaluated automatically, leaving a small residual system solvable by Gröbner or SAT methods in fractions of a second.

Table 3. Solver timings for BIVIUM over random (key, IV) pairs. Each interval shows the 90% confidence range of per-instance times (Intel i7-8700, 6 cores/12 threads, 32 GB RAM)

# ks bits	slimgb (ms)	std (ms)	MiniSat (s)	CrMiniSat (s)
180	(160, 195)	(326, 397)	(9.33, 84.03)	(9.61, 40.39)
185	(159, 170)	(332, 367)	(6.88, 60.24)	(7.33, 28.41)
190	(119, 134)	(353, 411)	(6.94, 63.98)	(9.59, 38.26)
195	(123, 138)	(342, 397)	(6.55, 67.57)	(7.69, 25.26)

The results indicate that Gröbner basis solving (**slimgb**, and **std** in SINGULAR [32]) is faster than SAT solvers (MiniSat, CryptoMiniSat [33]) on these UNSAT-dominated instances. The best performance occurs near $K \approx 190$, where the average runtime is $0.12 \cdot 2^{37} \text{ s} \approx 2^{34} \text{ s}$ [22], improving the earlier 2^{39} s for algebraic attacks [15].

6.3. Algebraic attack on Aradi. ARADI is a 16-round substitution–permutation network (SPN) operating on 128-bit blocks under a 256-bit key [20]. The internal state consists of four 32-bit words (W, X, Y, Z) . Each round applies round key (RK) addition, a column-wise Toffoli layer π , and a word-wise linear layer Λ_i with $i \bmod 4$; a final key addition completes encryption. The design and algebraic structure of these Toffoli and linear layers have been studied in [5, 6, 7], showing that specific composition patterns influence degree growth and potential algebraic distinguishers. The round transformation is

$$(W, X, Y, Z) \xrightarrow{\oplus RK^i} (\bar{W}, \bar{X}, \bar{Y}, \bar{Z}) \xrightarrow{\pi} (\tilde{W}, \tilde{X}, \tilde{Y}, \tilde{Z}) \xrightarrow{\Lambda_i} (W', X', Y', Z'),$$

where (W', X', Y', Z') serve as inputs to the next round.

Key addition introduces 128 linear relations per round. The Toffoli layer π acts independently on each bit column. For each bit position $j \in \{0, \dots, 31\}$, the word bits update as

$$X \leftarrow X + (W \cdot Y), \quad Z \leftarrow Z + (X \cdot Y), \quad Y \leftarrow Y + (W \cdot Z), \quad W \leftarrow W + (X \cdot Z),$$

with all operations over $\text{GF}(2)$. Thus π can be modeled as a 4-bit S-box per column, encoded by 21 quadratic implicit equations linking the 4 input and 4 output bits. Each round therefore contributes $32 \times 21 = 672$ quadratic equations from the Toffoli layer.

The linear layer Λ_i acts word-wise. Writing each 32-bit word as upper and lower 16-bit halves (u, l) and denoting by S_{16}^m a left rotation by m bits, it updates

$$(u, l) \mapsto (u + S_{16}^{a_i}(u) + S_{16}^{c_i}(l), \quad l + S_{16}^{a_i}(l) + S_{16}^{b_i}(u)),$$

where the offsets (a_i, b_i, c_i) depend on $i \bmod 4$ [20].

A Boolean model over $\text{GF}(2)$ is built in

$$R = \text{GF}(2)[k_0, \dots, k_{255}, p_0, \dots, p_{127}, s_w^{(1)}, s_x^{(1)}, s_y^{(1)}, s_z^{(1)}, \dots, s_w^{(16)}, s_x^{(16)}, s_y^{(16)}, s_z^{(16)}],$$

where k_i are key bits, p_j are known plaintext bits, and $s_{\bullet}^{(r)}$ denote the 4-bit S-box outputs across 16 rounds. Using one known plaintext–ciphertext pair and field equations produces a sparse system with about 2,432 variables (256 key bits, 128 plaintext bits, and 16×128 auxiliary bits) and 10,880 equations (16 rounds $\times$ 672 quadratics from π plus 128 linear output constraints).

The multistep solver MULTISOLVE is applied to the full 16-round model. Here the linear-elimination phase is disabled, and a bounded Gröbner run of degree 2 is executed with a 60s timeout per attempt. An always-try rule invokes

the bounded solver at every node. Key bits are guessed in a fixed depth-first order by dividing the 256-bit key into eight 32-bit blocks. Different block orders behave similarly; the reversed-block order performs slightly better. Robustness tests over 2^{16} random (key, plaintext) pairs show a clear transition: once 251 key bits are assigned, almost all instances solve at degree 2; occasional timeouts at 251 resolve completely when 252 bits are guessed.

Table 4. Multistep configuration for ARADI. Measurements on Ubuntu 22.04 (AMD EPYC 7763, 64 cores) using MAGMA v2.27.5 (sparse F4 implementation)

# guessed key bits	GrobnerSafe time (s)	Std. dev. (s)
251	34.63	1.89
252	24.96	1.76

Interpreting the average runtimes ($\log_2 34.63 \approx 5.11$, $\log_2 24.96 \approx 4.64$) gives an overall complexity of about 2^{256} seconds for the multistep strategy, compared with $2^{256.5}$ for a fixed-depth hybrid at 252 guesses. A sharp threshold is observed between 250 and 251 guesses: at 251, nearly all instances solve at degree 2 within tens of seconds, whereas at 250 a degree-2 bound is usually insufficient. While this does not constitute a practical break, since single encryptions are far faster than one second, experiments show a sharp drop in solving difficulty around this boundary. Further details and timing data appear in [25].

7. Conclusion. Algebraic cryptanalysis links the security of a cipher with the difficulty of solving polynomial systems over finite fields, which can be analyzed in a unified way once the cipher is expressed in algebraic form. Symbolic, logical, and hybrid approaches then appear as different ways of exploring the same structured search space. The move from static hybrid attacks to adaptive multistep and oracle-based methods represents a practical advance, reducing unnecessary computation by adjusting the number of guessed variables according to the observed hardness of each subsystem. The case studies on Bluetooth E0, BIVIUM, TRIVIUM, and ARADI show that modern symmetric ciphers can be modeled and analyzed within a single mathematical framework based on difference algebra and polynomial solving. The observed thresholds in solving degree and guessing depth offer practical indicators of algebraic resistance. Future work may include designing improved oracle functions, possibly using machine learning, and extending these ideas to post-quantum and lightweight cryptographic primitives.

REFERENCES

- [1] W. W. ADAMS, P. LOUSTAUNAU. An introduction to Gröbner bases. Grad. Stud. Math., vol. **3**. Providence, RI, American Mathematical Society, 1994.
- [2] G. V. BARD. Algebraic Cryptanalysis. Dordrecht, Springer, 2009.
- [3] L. BETTALE, J.-C. FAUGÈRE, L. PERRET. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.* **3**, 3 (2009), 177–197.
- [4] Bluetooth Special Interest Group (SIG). Bluetooth Core Specification, Revision 5.2, 2019. Available at: <https://www.bluetooth.com/specifications/bluetooth-core-specification>.
- [5] E. BELLINI, M. RACHIDI, R. ROHIT, S. K. TIWARI. On the structural properties of Toffoli gate composition in ARADI: implications for algebraic distinguishers. Lecture Notes in Comput. Sci., vol. **15826**. Cham, Springer, 2025, 400–425.
- [6] E. BELLINI, M. RACHIDI, R. ROHIT, S. K. TIWARI. Mind the Composition of Toffoli Gates: Structural Algebraic Distinguishers of ARADI. Cryptology ePrint Archive, Paper 2024/1559, 2024, <https://eprint.iacr.org/2024/1559>.
- [7] E. BELLINI, M. FORMENTI, D. GÉRAULT et al. CLAASping ARADI: Automated Analysis of the ARADI Block Cipher. In: Progress in Cryptology – INDOCRYPT 2024, Part II. Lecture Notes in Comput. Sci., vol **15496**. Cham, Springer, 2025, 90–113, https://doi.org/10.1007/978-3-031-80311-6_5.
- [8] L. BETTALE, J.-C. FAUGÈRE, L. PERRET. Solving polynomial systems over finite fields: improved analysis of the hybrid approach. ISSAC 2012—Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation. New York, Association for Computing Machinery (ACM), 2012, 67–74.
- [9] W. BOSMA, J. J. CANNON, C. PLAYOUST. The Magma algebra system. I. The user language. *J. Symbolic Comput.* **24**, 3–4 (1997), 235–265.
- [10] R. M. COHN. *Difference Algebra*. New York–London–Sydney, Interscience Publishers, John Wiley & Sons, 1965.

- [11] N. COURTOIS, A. KLIMOV, J. PATARIN, A. SHAMIR. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. *Advances in Cryptology—EUROCRYPT 2000* (Bruges). *Lecture Notes in Comput. Sci.*, Vol. **1807**. Berlin, Springer-Verlag, 2000, 392–407.
- [12] N. T. COURTOIS, G. V. BARD. Algebraic Cryptanalysis of the data encryption standard. *Cryptography and Coding, Lecture Notes in Comput. Sci.*, Vol. **4887**. Berlin, Springer, 2007, 152–169.
- [13] N. T. COURTOIS, J. PIEPRZYK. Cryptanalysis of block ciphers with overdefined systems of equations. *Advances in cryptology—ASIACRYPT 2002*. *Lecture Notes in Comput. Sci.*, Vol. **2501**. Berlin, Springer-Verlag, 2002, 267–287.
- [14] C. DE CANNIÈRE, B. PRENEEL. Trivium Specifications. eSTREAM – ECRYPT Stream Cipher Project. Available at: <https://www.ecrypt.eu.org/stream/triviumpf.html>.
- [15] T. EIBACH, G. VÖLKELE, E. PILZ. Optimising Gröbner Bases on Bivium. *Math. Comput. Sci.* **3**, 2 (2010), 159–172.
- [16] J.-C. FAUGÈRE. A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra* **139**, 1–3 (1999), 61–88.
- [17] J.-C. FAUGÈRE. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. New York, Association for Computing Machinery (ACM), 2002, 75–83.
- [18] M. R. GAREY, D. S. JOHNSON. *Computers and intractability: A guide to the theory of NP-completeness*. Ser. Books Math. Sci. San Francisco, CA, W. H. Freeman and Co., 1979.
- [19] V. GERDT, R. LA SCALA. Noetherian quotients of the algebra of partial difference polynomials and Gröbner bases of symmetric ideals. *J. Algebra* **423** (2015), 1233–1261.
- [20] P. GREENE, M. MOTLEY, B. WEEKS. ARADI and LLAMA: Low-Latency Cryptography for Memory Encryption. *Cryptology ePrint Archive*, Paper 2024/1240, 2024, <https://eprint.iacr.org/2024/1240>
- [21] A. KLEIN. *Stream Ciphers*. London, Springer, 2013.

- [22] R. LA SCALA, S. K. TIWARI. Stream/block ciphers, difference equations and algebraic attacks. *J. Symbolic Comput.* **109** (2022), 177–198.
- [23] R. LA SCALA, S. POLESE, S. K. TIWARI, A. VISCONTI. An algebraic attack to the Bluetooth stream cipher E0. *Finite Fields Appl.* **84** (2022), Paper No. 102102, 29 pp.
- [24] R. LA SCALA, F. PINTORE, S. K. TIWARI, A. VISCONTI. A multistep strategy for polynomial system solving over finite fields and a new algebraic attack on the stream cipher Trivium. *Finite Fields Appl.* **98** (2024), Paper No. 102452, 33 pp.
- [25] R. LA SCALA, S. K. TIWARI. Oracle-Based Multistep Strategy for Solving Polynomial Systems Over Finite Fields and Algebraic Cryptanalysis of the Aradi Cipher. Cryptology ePrint Archive, Paper 2025/1097, 2025, <https://eprint.iacr.org/2025/1097>
- [26] R. LA SCALA, A. N. ZUBKOV. Donkin–Koppinen filtration for general linear supergroups. *Algebr. Represent. Theory* **15**, 5 (2012), 883–899.
- [27] F. MASSACCI, L. MARRARO. Logical cryptanalysis as a SAT problem. *J. Automat. Reason.* **24**, 1–2 (2000), 165–203.
- [28] S. RAMOS-CALDERER, C. BRAVO-PRIETO, R. LIN ET AL. Solving systems of Boolean multivariate equations with quantum annealing. *Physical Review Research*, **4**, 1 (2022), paper No. 013096, 9 pp. <https://doi.org/10.1103/PhysRevResearch.4.013096>.
- [29] J. F. RITT, H. W. RAUDENBUSH. Ideal theory and algebraic difference equations. *Trans. Amer. Math. Soc.* **46** (1939), 445–452.
- [30] Y. SHAKED, A. WOOL. Cryptanalysis of the Bluetooth E0 Cipher Using OBDDs. In: Information Security. ISC 2006. Lecture Notes in Comput. Sci., vol **4176**. Berlin, Heidelberg, Springer, 2006, 187–202, https://doi.org/10.1007/11836810_14.
- [31] C. E. SHANNON. Communication theory of secrecy systems. *Bell System Tech. J.* **28** (1949), 656–715.
- [32] W. DECKER, G.-M. GREUEL, G. PFISTER, H. SCHÖNEMANN. Singular 4-1-2 — A computer algebra system for polynomial computations, 2019, <http://www.singular.uni-kl.de>.

- [33] M. Soos M. CryptoMiniSat 5: An Advanced SAT Solver, 2021. Available at: <https://www.msoos.org/cryptominisat5>.

Roberto La Scala
Dipartimento di Matematica
Università degli Studi di Bari “Aldo Moro”
Via Orabona 4, 70125 Bari, Italy
e-mail: roberto.lascalea@uniba.it

Sharwan K. Tiwari
Cryptography Research Centre
Technology Innovation Institute
Abu Dhabi, United Arab Emirates
e-mail: sharwan.tiwari@tii.ae

Received October 12, 2025

Accepted December 11, 2025