

Comparative Evaluation of Open-Source Implementations of Gröbner Basis Computation Algorithms for Small-Scale AES Polynomial Systems over GF(2)

Pavel Holý | Supervisor: Mgr. Martin Jureček, Ph.D.
Faculty of Information Technology, Czech Technical University in Prague

1. Why benchmark algebraic cryptanalysis?

AES is a widely used encryption algorithm that protects data with a secret key. Algebraic cryptanalysis seeks to recover that key by solving polynomial equations derived from known plaintext and ciphertext. A **Gröbner basis** transforms these equations into a form suitable for key recovery. But this computation can be expensive: **which open-source implementation should we use?**

We study **small-scale AES**: simplified variants with fewer rounds and a smaller state, retaining AES-like algebraic structure while making experiments feasible.



2. A reusable benchmarking framework

We built a **modular Python framework** that generates input systems, runs implementations under controlled resource limits, and collects **runtime, peak memory and CPU usage**, alongside success rates.

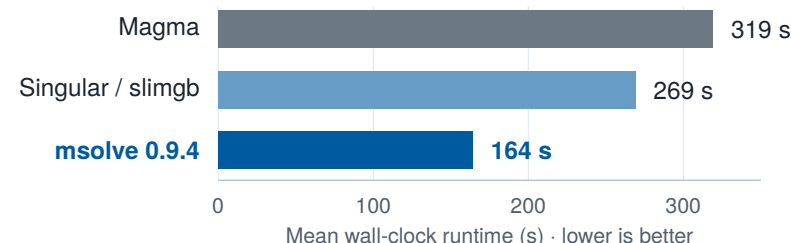
- **20 open-source implementations** evaluated against **Magma**, a state-of-the-art commercial system for Gröbner basis computation.
- **Four variants, ten keys each: 40 systems** over GF(2), including field equations.
- **One-hour timeout**; a dual-Xeon server with 24 physical cores and 754 GiB RAM.

3. What the work contributes

- **Evidence for solver selection**: identifying competitive open-source implementations for algebraic cryptanalysis.
- **A reusable experiment pipeline** for evaluating further implementations and input families.
- **Faster open-source computation** through two targeted improvements to msolve.

4. Open source competes with Magma

In the hardest experiment, **only Singular/slimgb and msolve outperformed Magma in mean runtime**. msolve was the fastest, with **49% lower mean runtime than Magma**. All three recovered **10 out of 10 keys**.

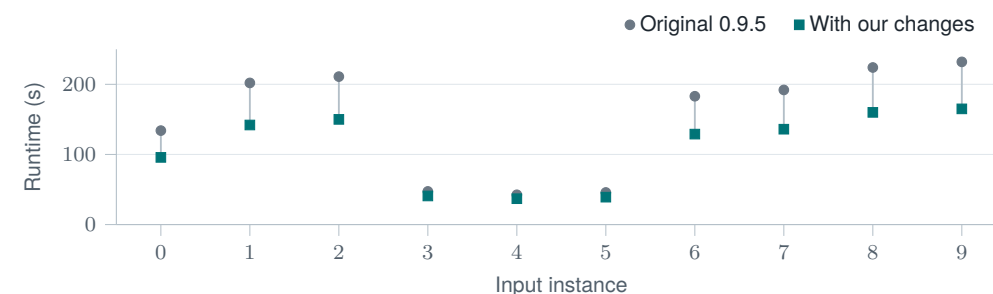


Hard: 2 rounds, 32-bit key, 8 plaintext-ciphertext pairs per key. Means over 10 instances.

Mean CPU use: msolve 13.4 cores; Magma and slimgb one core each. Rankings depend on the input and hardware.

5. Making the fastest implementation faster

We optimized **msolve's F4 symbolic preprocessing**: **cache leading-monomial metadata** instead of repeatedly looking it up, and **filter reducer candidates by degree** before expensive checks.



24% lower runtime on average; up to 30%. Every instance improved, with essentially unchanged peak memory.

Same ten Hard inputs; original vs. modified msolve 0.9.5. The 24% figure is the mean of per-instance relative reductions.