A crossbred algorithm for solving Boolean polynomial systems

eprint.iacr.org/2017/372

(by Antoine Joux and Vanessa Vitse)

Tung Chou

Multivariate polynomial systems

- Field $\mathbf{F}_q$
- n variables: $x_1, \ldots, x_n$
- m polynomials: $f_1, \ldots, f_m \in \mathbf{F}_q[x_1, \ldots, x_n]$
- Degree $d = Max(\{\mathrm{Deg}(f_i)\})$
- Monomials: $x_1^{e_1} \ldots x_n^{e_n}$

Multivariate polynomial systems

- Example of $(q, m, n, d) = (7, 3, 3, 4)$:

$$f_1 = x_1 x_2^3 + 6x_2 x_3 + x_1 x_3 + 2x_1 + 3$$
$$f_2 = x_1 x_2 x_3 + 4x_2 x_3^2 + x_1 x_3 + x_1 + 5$$
$$f_3 = 6x_1^4 + x_2 x_3^2 + 3x_1 x_3 + x_2 + x_3$$

- Example of $(q, m, n, d) = (2, 2, 3, 3)$:

$$f_1 = x_1 x_2 + x_1 x_3 + x_2 x_3 + x_2 + x_3$$
$$f_2 = x_1 x_2 x_3 + x_1 x_3 + x_2 x_3 + x_1 + 1$$

- The task is to find a solution for x_i's s.t. $f_j = 0\ \forall j$.

System solving vs. Cryptography

- NP-Hard even when $d = 2$ (quadratic) and $q = 2$ (Boolean)
- Foundation of Multivariate Cryptography.
- Breaking crypto systems can be reduced to system solving

  $\Rightarrow$ Algebraic attacks

Exhaustive search

- Highly parallelizable
- Extremely memory efficient
- Complexity: $O(poly \cdot 2^n)$
- Presumably the best algorithm when n is not too large:

  $(d, n) = (2, 48)$ can be solved in 21 minutes using 1 GPU.

eXtended Linearization (XL)

- Extend: multiply f_i's by all monomials of degree $D - d$
- View the extended system as a matrix and "solve" it

  $\Rightarrow$ Macaulay matrix

$$
\begin{array}{c}
\begin{array}{ccccccccccc}
X_1X_2 & X_1X_3 & X_1X_4 & X_1 & X_2X_3 & X_2X_4 & X_2 & X_3X_4 & X_3 & X_4 & 1
\end{array}\\
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0\\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1\\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Moderately parallelizable
- Works on sparse matrix: memory-efficient
- Works well for random, $m \gg n$ (small-D) systems

Gröbner basis solvers

- Buchberger's algorithm and it variants (F4, F5, etc)
- Adaptively extend and linearize polynomials
- Not parallelizable?
- Polynomials are much denser: not memory-efficient
- Some algebraic attacks have been carried out using GB solvers
- Appear to be quite efficient when memory fits

Exhaustive search + XL

Determine the parameter $k < n$, then

- Enumerate all 2^{n-k} solutions for $x_{k+1}, \ldots, x_n$.
- For each guess, solve the resulting system with XL
- Reducing number of variables results in smaller D

XL + exhaustive search

Determine parameters D, k, then

- Compute the degree-D Macaulay matrix
- (1) Eliminate all monomials that contain $x_1, \ldots, x_k$.
- (2) Solve the resulting system of $n - k$ variables with exhaustive search.

XL + exhaustive search

- Macaulay matrix:

$$\begin{array}{c}
\begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_1X_4 & X_1 & X_2X_3 & X_2X_4 & X_2 & X_3X_4 & X_3 & X_4 & 1 \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{array}$$

- After reduction:

$$\begin{array}{c}
\begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_1X_4 & X_1 & X_2X_3 & X_2X_4 & X_2 & X_3X_4 & X_3 & X_4 & 1 \end{array} \\
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}$$

XL + exhaustive search + linear algebra (Joux–Vitse)

Determine parameters D, k, then

- (1) Eliminate all monomials of degree > 1 in $x_1, \ldots x_k$.
- (2) Enumerate all 2^{n-k} solutions for $x_{k+1}, \ldots, x_n$.
- (3) Solve the resulting **LINEAR** systems in $x_1, \ldots, x_k$.

XL + exhaustive search + linear algebra (Joux–Vitse)

- After reduction:

$$\begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_2X_3 & X_1X_4 & X_2X_4 & X_3X_4 & X_1 & X_2 & X_3 & X_4 & 1 \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

- Last 3 equations:

$$\begin{cases} (X_4+1)X_1 + X_2 + X_3 + 1 = 0 \\ (X_4+1)X_2 = 0 \\ X_1 + X_2 + (X_4+1)X_3 + 1 = 0 \end{cases}$$

Fukuoka MQ challenge

- www.mqchallenge.org
- Joux–Vitse results

Number of Vars ($m = 2n$)	External hybridation h	Parameters $(D, n-h-k)$	Max CPU (estimate)	Real CPU (rounded)
67	9	$(4, 36)$	6 200 h	3 100 h
68	9	$(4, 37)$	11 200 h	4 200 h
69	9	$(4, 38)$	15 400 h	15 400 h
70	13	$(4, 34)$	33 000 h	16 400 h
71	13	$(4, 35)$	60 000 h	13 200 h
72	13	$(4, 36)$	110 000 h	71 800 h
73	13	$(4, 37)$	190 000 h	14 300 h
74	12	$(4, 39)$	360 000 h	8 100 h

- D is much smaller than XL degree
- $(m, n) = (148, 74)$ was solved with 448 CPU cores.
- $(m, n) = (132, 66)$ was solved in 8 days with 128 FPGAs using exhaustive search.

Open problems

- Complexity of the Joux–Vitse algorithm?
- Can the idea be used for attacking real systems?