Concrete Bit-Operation Cost of XL

For Solving Multivariate Quadratic Systems Using Wiedemann and Berlekamp–Massey

Hülya Evkan Ruben Niederhagen

Motivation

Security estimates should be comparable.

- XL/Wiedemann estimates are often given at different abstraction levels.
- Some count field operations; some include bit-level arithmetic.
- Hidden constants can change concrete security estimates.

Goal: a unified bit-operation model for XL-based attacks.

Buchberger's Algorithm

Iteratively eliminate the lexicographically leading terms and perform standard elimination.

Example

$$F = \begin{pmatrix} x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 \\ x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 \\ x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 \\ x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 \end{pmatrix}$$

Find x for $F(x) = (0, 0, 0, 0)$.

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \quad (1)$$

$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \quad (2)$$

$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \quad (3)$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \quad (4)$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$

$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$

$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_2 + 1 = 0 \qquad (2) + (3) = \tag{5}$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$

$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$

$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_2 + 1 = 0 \qquad (2) + (3) = \tag{5}$$

$$x_2 + x_1 + 1 = 0 \qquad (4) + (5) = \tag{6}$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$
$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$
$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_2 + 1 = 0 \quad (2) + (3) = \tag{5}$$
$$x_2 + x_1 + 1 = 0 \quad (4) + (5) = \tag{6}$$
$$x_3 + x_2 = 0 \quad (1) + (4) = \tag{7}$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$
$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$
$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_2 + 1 = 0 \quad (2) + (3) = \tag{5}$$
$$x_2 + x_1 + 1 = 0 \quad (4) + (5) = \tag{6}$$
$$x_3 + x_2 = 0 \quad (1) + (4) = \tag{7}$$
$$x_3x_2x_1 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 + x_3 = 0 \quad x_3(1) + (2) = \tag{8}$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$
$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$
$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_2 + 1 = 0 \quad (2) + (3) = \tag{5}$$
$$x_2 + x_1 + 1 = 0 \quad (4) + (5) = \tag{6}$$
$$x_3 + x_2 = 0 \quad (1) + (4) = \tag{7}$$
$$x_3x_2x_1 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 + x_3 = 0 \quad x_3(1) + (2) = \tag{8}$$
$$x_3x_2x_1 + x_4x_1 + x_3x_2 + x_3x_1 + x_2 + 1 = 0 \quad x_3(4) + (3) = \tag{9}$$
$$x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + 1 = 0 \quad (8) + (9) = \tag{10}$$
$$x_4 + x_3 + x_2 + 1 = 0 \quad x_1(7) + (10) = \tag{11}$$
$$x_4 = 1 \quad (7) + (11) = \tag{12}$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \quad (1)$$

$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \quad (2)$$

$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \quad (3)$$

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \quad (4)$$

$$x_2 + x_1 + 1 = 0 \quad (6)$$

$$x_3 + x_2 = 0 \quad (7)$$

$$x_4 = 1 \quad (12)$$

Buchberger's Algorithm

Example

$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \tag{1}$$
$$x_4x_3 + x_4x_1 + x_3x_2 + x_2x_1 + x_4 = 0 \tag{2}$$
$$x_4x_3 + x_4x_1 + x_3x_1 + x_2 + 1 = 0 \tag{3}$$
$$x_3x_2 + x_3x_1 + x_2x_1 + x_4 + x_1 = 0 \tag{4}$$
$$x_2 + x_1 + 1 = 0 \tag{6}$$
$$x_3 + x_2 = 0 \tag{7}$$
$$x_4 = 1 \tag{12}$$
$$x_4x_3x_1 + x_4x_3 + x_2x_1 + x_4 + x_3 + x_1 = 0 \qquad x_3(3) + (4) = \tag{13}$$
$$x_4x_3x_1 + x_3x_2x_1 + x_3x_1 + x_2x_1 + x_4 + x_3 + x_2 + x_1 = 0 \qquad (1) + x_3(2) = \tag{14}$$
$$x_2 = 0 \qquad (14) + (13) + (9) + x_4(7) + x_4(6) + x_2(7) + (12) = \tag{15}$$
$$x_3 = 0 \qquad (7) + (15) = \tag{16}$$
$$x_1 = 1 \qquad (6) + (15) = \tag{17}$$

XL in one slide

Given quadratic equations

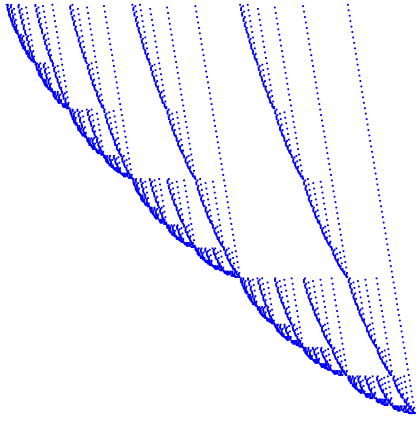
$$f_1, \ldots, f_m \in K[x_1, \ldots, x_n],$$

XL multiplies them by monomials to obtain equations up to degree D.

$$uf_i, \qquad \deg(uf_i) \leq D$$

This gives a Macaulay matrix A_D. Now solve:

$$A_D y = b.$$



Solving the polynomial system becomes a sparse linear-algebra problem.

Wiedemann + Berlekamp–Massey

Wiedemann solves the sparse system through matrix–vector products:

$$b,\ A_D b,\ A_D^2 b, \ldots$$

A scalar sequence is extracted:

$$s_k = u^\top A_D^k b.$$

Berlekamp–Massey recovers the linear recurrence
and gives an annihilating polynomial $\mu()$ with $\mu(A_D)b = 0$.

Wiedemann + Berlekamp–Massey

Wiedemann solves the sparse system through matrix–vector products:

$$b,\ A_D b,\ A_D^2 b, \ldots$$

A scalar sequence is extracted:

$$s_k = u^\top A_D^k b.$$

Berlekamp–Massey recovers the linear recurrence
and gives an annihilating polynomial $\mu()$ with $\mu(A_D)b = 0$.

If $\mu_0 \neq 0$, this gives:

$$A_D \left(-\mu_0^{-1} (\mu_d A_D^{d-1} b + \mu_{d-1} A_D^{d-2} b + \cdots + \mu_1 b) \right) = b.$$

Wiedemann + Berlekamp–Massey

Wiedemann solves the sparse system through matrix–vector products:

$$b,\ A_D b,\ A_D^2 b, \ldots$$

A scalar sequence is extracted:

$$s_k = u^\top A_D^k b.$$

Berlekamp–Massey recovers the linear recurrence
and gives an annihilating polynomial $\mu()$ with $\mu(A_D)b = 0$.

If $\mu_0 \neq 0$, this gives:

$$A_D \left(-\mu_0^{-1} \left(\mu_d A_D^{d-1} b + \mu_{d-1} A_D^{d-2} b + \cdots + \mu_1 b \right) \right) = b.$$

Dominant cost: repeated sparse matrix–vector multiplication.

CryptAttackTester — CAT

- C++ framework to implement algorithm as circuit.
- Field arithmetic is implemented using bit operations.
- The final cost is a bit-operation count.
- Framework “runs” the circuit and counts bit operations.

$\Rightarrow$ Define cost formulas for circuit, verify for small parameters.

The cost in bit operations directly relates to bit-security.

CAT and random memory access

CAT has no RAM primitive: data access is part of the circuit.

Potential weakness: random RAM access may be expensive in CAT.

- Runtime-dependent lookup becomes selection circuitry.
- This may be pessimistic for memory-heavy algorithms.

This does not affect our XL–Wiedemann circuit.

All “memory addresses” in the dominant Wiedemann computation are known at circuit-generation time.

XL in CAT: three implementation variants

Construct a Wiedemann XL circuit for a specific problem size.

- **Baseline:** Macaulay matrix entries are ordinary field elements.
- **Const:** the matrix is fixed; multiplication by constants is cheaper.
- **Const+bucket:** fixed coefficients allow static bucket assignment.

Bucketing is useful only because the access pattern is static.

From field operations to bit operations

Separate the combinatorics from the field implementation.

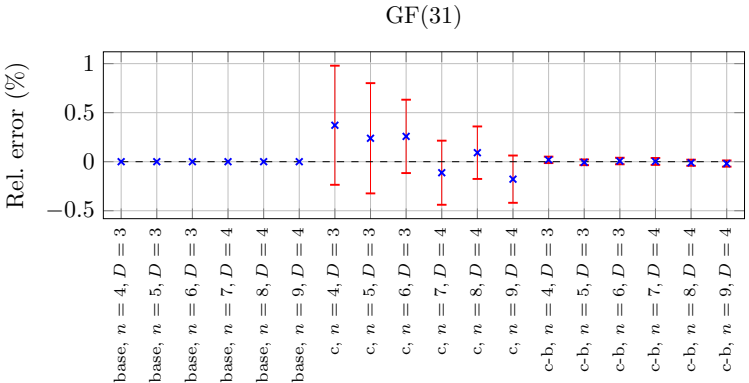
XL/Wiedemann structure $\Rightarrow$ A, S, M, I

$$Aa_q + Ss_q + Mm_q + Ii_q$$

Field	q	add a_q	sub s_q	mul m_q	inv i_q	mul const μ_q	bucket mul const $\mu_q q/(q-2)$
GF(2)	2	1	1	1	0	0	–
GF(31)	31	41	46	207	1449	153.74	164.34
GF(256)	256	8	8	132	752	38.75	39.06

Formula validation

Do the formulas predict the circuits?



- Baseline: exact agreement.
- Optimized variants: small data-dependent deviations.
- Mean errors remain centered near zero.

Comparison with published Wiedemann–XL estimates

Same attack assumptions ⇒ very similar estimates

Scheme	q	n	m	WXL	base	const	c+b	D	k
MQOM2-L1-gf256	256	48	48	157	158.0	156.5	155.2	22	3
MQOM2-L3-gf256	256	72	72	217.5	218.6	217.0	215.2	28	5
MQOM2-L5-gf256	256	96	96	280.4	281.4	279.8	277.8	33	8
uov-Ip	256	112	44	145	146.9	145.3	144.2	20	3
uov-III	256	184	72	218	218.6	217.0	215.2	28	5
uov-V	256	244	96	278	281.4	279.8	277.8	33	8
QR-UOV I (31, 165, 60, 3)	31	75	60	163	171.3	171.0	168.8	20	7
QR-UOV III (31, 246, 87, 3)	31	111	87	226	234.7	234.4	232.2	28	9
QR-UOV V (31, 324, 114, 3)	31	156	114	286	294.4	294.0	291.8	31	15

Comparison with published Wiedemann–XL estimates

Same attack assumptions $\Rightarrow$ very similar estimates

- **MQOM, UOV:** very close to our model; we make essentially the same algorithmic assumptions concrete.
- **QR-UOV:** lower published estimate because bit-operation costs of field arithmetic are not included.

Takeaways

- We give concrete bit-operation formulas for XL with Wiedemann and Berlekamp–Massey.
- The formulas are validated against executable CAT circuits.
- Published MQOM/UOV estimates are reproduced closely because they use similar algorithmic assumptions.

A common cost model makes security estimates comparable.

Operation Count Formulas

	A	M	M_{fixed}
baseline	$(2\,R_q - 1)Z_q$	$(2\,R_q - 1)Z_q$	0
const	$(2\,R_q - 1)Z_q$	0	$(2\,R_q - 1)Z_q$
const+bucket	$\left(R_q(q-2) + \frac{Z_q(q-1)}{q}\right)(2\,R_q - 1)$	0	$(2\,R_q - 1)R_q(q-2)$

	A	S	M	I
Berlekamp–Massey	$\frac{3\,(R_q+1)R_q}{2}$	$\frac{(3\,R_q+5)R_q}{2}$	$\frac{(9\,R_q+13)R_q}{2}$	0
normalization	0	0	$R_q + 1$	1
evaluation	$R_q n$	0	$R_q n$	0

Operation Count Formulas

GF(31), baseline model.

$$C_{31}^{\text{base}} = \frac{2169}{2} R_{31}^2 + 496\, R_{31} Z_{31} + 28\, R_{31}\ell + 248\, R_{31} n + \frac{3557}{2} R_{31} - 248\, Z_{31} + 1656$$

GF(31), const+bucket model.

$$C_{31}^{\text{buck}} = 58\, R_{31}^2 \mu_{31} \frac{31}{29} + \frac{6925}{2} R_{31}^2 + \frac{2460}{31} R_{31} Z_{31} + 28\, R_{31}\ell - 29\, R_{31}\mu_{31}\frac{31}{29} + 248\, R_{31} n + \frac{1179}{2} R_{31} - \frac{1230}{31} Z_{31} + 1656$$