Analysis of QUAD

Owen (Chia-Hsin) Chen, National Taiwan University

March 27, FSE 2007, Luxembourg

Work at Academia Sinica supervised by Dr. Bo-Yin Yang
Jointly with Drs. Dan Bernstein and Jiun-Ming Chen

$\texttt{QUAD}(q, n, r)$, a Family of Stream Ciphers

State: n-tuple $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in K^n$, $K = \mathrm{GF}(q)$

Update: $\mathbf{x} \leftarrow (Q_1(\mathbf{x}), Q_2(\mathbf{x}), \ldots, Q_n(\mathbf{x}))$. Here each Q_j is a randomly chosen, public quadratic polynomial

Output: r-tuple $(P_1(\mathbf{x}), P_2(\mathbf{x}), \ldots, P_r(\mathbf{x}))$ before updating (again, each P_j is a random, public quadratic polynomial)

At Eurocrypt 2006, Berbain-Gilbert-Patarin reported speeds for $\texttt{QUAD}(2, 160, 160)$, $\texttt{QUAD}(16, 40, 40)$, and $\texttt{QUAD}(256, 20, 20)$.

A graphical Depiction

$$
\begin{array}{ccccccc}
\mathbf{x}_0 & \longrightarrow & \mathbf{x}_1 = \mathbf{Q}(\mathbf{x}_0) & \longrightarrow & \mathbf{x}_2 = \mathbf{Q}(\mathbf{x}_1) & \longrightarrow & \mathbf{x}_3 = \mathbf{Q}(\mathbf{x}_2) & \longrightarrow & \cdots \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \\
\mathbf{y}_0 = \mathbf{P}(\mathbf{x}_0) & & \mathbf{y}_1 = \mathbf{P}(\mathbf{x}_1) & & \mathbf{y}_2 = \mathbf{P}(\mathbf{x}_2) & & \mathbf{y}_3 = \mathbf{P}(\mathbf{x}_3) & & \cdots
\end{array}
$$

Typically q is a power of 2, allowing each output vector $\mathbf{y}_i \in \mathrm{GF}(q)^r$ to encrypt the next $r \lg q$ bits of plaintext in a straightforward way.

`QUAD`, "Provably Secure"?

- Security Theorem: Breaking `QUAD` implies the capability to solve $n + r$ random quadratic equations in n variables.

- Generic $\mathcal{MQ}$ (Multivariate Quadratics) is an NP-hard problem.

- All known algorithms to solve such a generic quadratic polynomial system have average time complexity $2^{an+o(n)}$ when $r/n =$ constant; most also require exponential space.

Difficult Generically, *But* . . .

Following the position paper of Koblitz-Menezes ("Another look at Provable Security" J. of Crypto.) we would like to discuss the implications of the security proof.

- How tight is the security reduction?
- How difficult is the underlying problem?
- What is the best attack known today?
- Is the security reduction complete?

Instances and Provability

We would like to proposed the following classification of instances of families of cryptosystems covered by security reductions:

Broken: We can attack and break the instance.

Unprovable: We can solve the underlying hard problem.

Unproven: A putative feasible attack on the instance need not lead to an improvement on the solution of the underlying hard problem due to the looseness factor in the security reduction.

Proved: Security proof works as advertised **for this instance**.

Today's System-Solving

State-of-the-art algorithms to solve m generic polynomial equations in n $\mathrm{GF}(q)$-variables are all related in some way to Buchberger's algorithm for computing Gröbner Bases.

- XL, first proposed by Lazard and rediscovered by Courtois *et al.* Essence: an elimination on a Macaulay Matrix. Also the adjuncts
  - FXL ('F' for "fix") introduces guessing variables.
  - XL2, running the elimination on the highest monomials only and then repeatedly multiply by variables to raise degrees.
- $\mathbf{F_4}$ (now in MAGMA) and $\mathbf{F_5}$, of which XL2 is an inferior form.

Facts of Life for XL

$$\#\text{ monomials: } T \;=\; [t^D]\ \left((1-t^q)^n(1-t)^{-(n+1)}\right); \tag{1}$$

$$\#\text{ free monoms: } T-I \;\geq\; [t^D]\ \left(\frac{(1-t^q)^n}{(1-t)^{n+1}}\prod_{i=1}^{m}\left(\frac{1-t^{d_i}}{1-t^{qd_i}}\right)\right). \tag{2}$$

Here $\deg p_i := d_i$, $[u]s :=$ coefficient of u in expansion of s. We expect a solution at $D_{XL} = \min\{D : \text{RHS of Eq. 2} \leq 0\}$. If the (p_i) is q-semi-regular (true almost always), Eq. 2 is $=$ as long as its RHS remains positive.

$$T = \binom{n+D}{D}, \quad T - I = [t^D]\ \left((1-t)^{m-n-1}\,(1+t)^m\right)$$

is the reduced case for large fields ($q > D$). $C_{XL} \approx 3kT^2(c_0 + c_1 \lg T)$ using a modified Wiedemann algorithm (k is average number of terms per equation).

XL with Homogenous Wiedemann

1. **Create the extended Macaulay matrix of the system to a certain degree D_{XL}:** Multiply each equation of degree d_i by all monomials up to degree $D_{XL} - d_i$ and take the matrix of coefficients.
2. **Randomly delete some rows then add some columns to form a square system, $A\mathbf{x} = 0$ where $\dim A = \beta T + (1-\beta)R$. Usually $\beta = 1$ works. Keep the same density of terms.**
3. **Apply the homogeneous version of Wiedemann's method to solve for $\mathbf{x}$:**
   (a) Set $k = 0$ and $g_0(z) = 1$, and take a random $\mathbf{b}$.
   (b) Choose a random $\mathbf{u}_{k+1}$ [usually the $(k+1)$-st unit vector].
   (c) Find the sequence $\mathbf{u}_{k+1}A^i\mathbf{b}$ starting from $i = 0$ and going up to $2N - 1$.
   (d) Apply g_k as a difference operator to this sequence, and **run the Berlekamp-Massey algorithm over $\mathrm{GF}(q)$ on the result** to find the minimal polynomial f_{k+1}.
   (e) Set $g_{k+1} := f_{k+1}g_k$ and $k := k + 1$. If $\deg(g_k) < N$ and $k < n$, go to (b).
4. **Compute the solution $\mathbf{x}$ using the minpoly $f(z) = g_k(z) = c_m z^m + c_{m-1}z^{m-1} + \cdots + c_\ell z^\ell$: Take another random $\mathbf{b}$. Start from $\mathbf{x} = (c_m A^{m-\ell} + c_{m-1}A^{m-\ell-1} + \cdots + c_\ell 1)\mathbf{b}$, continuing to multiply by A until we find a solution to $Ax = 0$.**
5. If the nullity $\ell > 1$ repeat the check below at every point of an affine subspace (q points if $\ell = 2$).
6. Obtain the solution from the last few elements of $\mathbf{x}$ and check its correctness.

$\texttt{QUAD}(256, 20, 20)$ Unprovable from $\mathcal{MQ}$

- Is 20 $\mathbf{GF(256)}$ variables in 40 equations hard to solve?
- We say no! Generic XL solves this in 2^{45} cycles, only a few hours on a decent computer.
- The technical details are: cycles per multiplication on a P4 ≈ 12 (3 L1 cache loads); $D_{XL} = 5$ and $T = 53130$. Max number of terms per equation is $k \lesssim 231$, so $C_{XL} \approx 9 \times 10^{12} \lesssim 2^{45}$.
- *Hence no security is provable [nor claimed by orig.* ***QUAD*** *paper] from $\mathcal{MQ}$ (20 vars, 40 eqs) over* $\mathrm{GF}(256)$.

Direct Attack

- Can $\texttt{QUAD}(256, 20, 20)$ be a cipher that is acceptably secure without being provable? We say no, and estimate 2^{63} cycles for a direct attack that breaks $\texttt{QUAD}(256, 20, 20)$.

- Often we can acquire some cipher stream via known plaintext. This attack only uses **two blocks (2^9 bits)** of output.

- Let the instance be $\mathbf{x}_{j+1} = Q(\mathbf{x}_j)$, $\mathbf{y}_j = P(\mathbf{x}_j)$ with $P, Q : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^n$. With (WLOG) $\mathbf{y}_0$ and $\mathbf{y}_1$, we solve for $\mathbf{x}_0$ via

$$P(\mathbf{x}_0) = \mathbf{y}_0,\ P(Q(\mathbf{x}_0)) = \mathbf{y}_1.$$

20 quadratics, 20 quartics over $\mathrm{GF}(256)$

- 2^{63} mults upper bound, real value should be more like $\lesssim 2^{60}$.
- Significant parameters are:
  - degree $D_{XL} = 10$,
  - #monomials $T = \binom{30}{10} = 30045015$,
  - #initial equations is $R = 20 \times \binom{28}{8} + 20 \times \binom{26}{6} = 66766700$,
  - total # terms in those equations is $\tau := kR = 20\binom{28}{8}\binom{22}{2} + 20\binom{26}{6}\binom{24}{4} = 63287924700.$

  Should be doable on a machine or cluster with 384GB of memory.

Testing Attack vs. $\texttt{QUAD}(256, n, n)$

n	9	10	11	12	13	14	15
D	7	7	7	8	8	8	8
C_{XL}	$2.29 \cdot 10^2$	$7.55 \cdot 10^2$	$2.30 \cdot 10^3$	$5.12 \cdot 10^4$	$1.54 \cdot 10^5$	$4.39 \cdot 10^5$	$1.17 \cdot 10^6$
lgC_{XL}	7.84	9.56	$1.12 \cdot 10$	$1.56 \cdot 10$	$1.72 \cdot 10$	$1.87 \cdot 10$	$2.02 \cdot 10$
T	$1.14 \cdot 10^4$	$1.94 \cdot 10^4$	$3.28 \cdot 10^4$	$1.26 \cdot 10^5$	$2.03 \cdot 10^5$	$3.20 \cdot 10^5$	$4.90 \cdot 10^5$
aTm	120	147	177	245	288	335	385
clks	14.6	13.6	12.1	13.1	12.9	12.8	12.7

MS C++ 7; P-D 3.0GHz, 2GB DDR2-533, T: #monomials, aTm: average terms in a row, clks: number of clocks per multiplication.

- Serial Code on $i386$ requires three dependent L1 accesses per multiplication (3 cycles K8/Core, 4 cycles P4) plus change.
- Unrolling loops for x86-64 saves 20%–25% cycles a multiplication.
- 256-semi-regularity assumption fits empirical data up to $n = 15$.

$\texttt{QUAD}(16, 40, 40)$ Unprovable, but not Broken

- 80 eqs. in 40 $\mathbf{GF}(16)$ vars. estimated to $< 2^{72}$ cycles in XL.
- Technical data: $D_{XL} = 8$, $T = 377348994$, and $k \lesssim 861$.
- So $\texttt{QUAD}(16, 40, 40)$ can *never* be "provably secure" from $\mathcal{MQ}$ (40,80). But we don't know how to break it in 2^{80}.
- Direct solution takes $\lesssim 2^{95}$ mults (guesstimated at 2^{100} cycles) via XL-Wiedemann ($D_{XL} = 14$, $T = 3245372870670$).
- Data complexity is 10000 TB (only $\sim 2^{56}$ bits) for the matrix.

Why Only 2 Blocks?

- Practical answer: we test with degree-8 equations; doesn't help.
- Theoretical answer: the XL operating degree is

$$D_{XL} = \min\left\{ D : [t^D] \; \frac{\left((1-t^2)(1-t^4)\right)^n}{(1-t)^{n+1}} \; < 0 \right\},$$

Hence $w := D_{XL}/n \approx$ the smallest positive zero of $f_n(w) :=$

$$\oint \frac{(1-z^2)^n(1-z^4)^n}{(1-z)^{n+1} z^{wn+1}} dz = \oint \frac{dz}{z(1-z)} \left(\frac{(1+z)(1-z^4)}{z^w} \right)^n$$

Diminishing Returns (for large q)

In asymptotic analysis, $f_n(w) = \oint \frac{dz}{z(1-z)} \left(\frac{(1+z)(1-z^4)}{z^w}\right)^n$ can only vanish if the saddle point equation of the integral, letting the derivative of the expression between the paren be zero:

$$(w-5)z^4 + z^3 - z^2 + z - w = 0$$

has double roots (a "monkey saddle"), which happens when w is very close to 0.2 (actually ≈ 0.200157957).

Similar computations including degree-8 equations only make it $w \approx 0.1998$. Clearly not worth our time.

$\texttt{QUAD}(2, 160, 160)$: An Unproven Case

- $\texttt{QUAD}(2, 160, 160)$ takes $\approx 2^{180}$ multiplications to attack directly: just solve 160 equations in 160 variables using XL.

- For $n < 200$, the effect of using quartic and degree-8 equations (2nd, 3rd output blocks and beyond) is not discernible.

- Similar asymptotics as above shows that for large n they (eventually) make a big difference.

- The underlying $\mathcal{MQ}$ problem of 160 vars and 320 equations takes 2^{140} multiplications, which seems high enough, but . . .

Tightness of Reduction

- `QUAD` attack implies an $\mathcal{MQ}$ attack *with a loss of efficiency*.

- Specifically, if λr bits of output from $\texttt{QUAD}(2, n, r)$ can be distinguished from uniform with advantage ϵ in time T, then a random $\mathcal{MQ}$ system of $n + r$ equations in n variables over $\mathrm{GF}(2)$ can be solved with probability $2^{-3}\epsilon/\lambda$ in time

$$T' \leq \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda + 2)T_S + \log\left(\frac{2^7 n \lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n \lambda^2}{\epsilon^2} T_S$$

where $T_S :=$ time to run one block of $\texttt{QUAD}(2, n, r)$.

Proven and Unproved Cases for $q = 2$

The looseness factor is about $2^{10}n^2\lambda^3/\epsilon^3$. If $\epsilon = 0.01$, $n = r$, and $L = \lambda n = 2^{40}$, this factor is then $2^{150}/n$. The theorem cannot conclude $T \geq 2^{80}$ without assuming that $T' \geq 2^{230}/n$.

- $n = 160$ is hence Unproven (original `QUAD` paper states this).
- $n = 256$: *Proven* for $L = 2^{22}, \epsilon = 0.01$, $T' \approx 2^{205}$ (multiplications). In fact we only need $T' \geq 2^{168}$.
- $n = 350$: *Proven* for $L = 2^{40}, \epsilon = 0.01$, $T' \approx 2^{263}$ (multiplications). We only needed $T' \geq 2^{221}$.

A Note on $T^{2.376}$

- Often $T^{2.376}$ is used as the cost of eliminations.

- This discounts the huge constant that is expected from the Coppersmith-Winograd paper.

- We improve $T^{2.376}$ to T^2, using a sparse matrix algorithm, but there are still factors in front of T^2.

- This explains the gap in the analysis for $\texttt{QUAD}(2, 350, 350)$.

Conclusions and TODOs

- Generically $\mathcal{MQ}$ is believed to be exponential in n. Complexity of breaking `QUAD` would then also be of the form $2^{an+o(n)}$. But the coefficient a $(= a(q, r/n))$ can be surprisingly small.

- `QUAD` is clearly a worthwhile attempt and worth optimizing further.

- We need tighter reductions. At the moment, we are reducing from what seems to be a more difficult problem to an easier problem.

- Comparisons between ciphers w. provably secure parameters?

- Taking into account storage access delays and parallelism?

Thanks to

- Our gracious hosts and organizers
- Academia Sinica and TWISC (Taiwan Info. Security Center)
- Dr. Bo-Yin Yang, Prof. Dan Bernstein, Dr. Jiun-Ming Chen.
- Everyone for being here.

QUESTIONS??

Why Wiedemann and not Lanczos

The two should be more or less equivalent in modern forms. We chose Wiedemann over Lanczos because in the "naive" forms

- Because it is easier to program well. Lanczos requires multiplying by a sparse matrix in opposite directions.
- We don't need to use a random diagonal vector.
- We just had the code ready to use.

Why XL and not $\mathbf{F_5}$

- Theoretical: Working on the top degree monomials, for large fields XL2/$\mathbf{F_4}$/$\mathbf{F_5}$ play with one fewer variable. This may not offset dense vs. sparse matrix equation solving difference if $\omega > 2$.

- Practical: If the matrices of $\mathbf{F_4}/\mathbf{F_5}$ will eventually become moderately dense, we will run out of memory before time.

$m-n$	D_{XL}	D_{reg}	$n=9$	$n=10$	$n=11$	$n=12$	$n=13$
0	2^m	m	6.090	46.770	350.530	3322.630	sigmem
1	m	$\lceil \frac{m+1}{2} \rceil$	1.240	8.970	53.730	413.780	2538.870
2	$\lceil \frac{m+1}{2} \rceil$	$\lceil \frac{m+2-\sqrt{m+2}}{2} \rceil$	0.320	2.230	12.450	88.180	436.600

Test results given on P4-3.2G, 2GB RAM, MAGMA-2.12 with $\mathbf{F_4}$.

- Pragmatic: we don't have a copy of $\mathbf{F_5}$ to play with.

Basic XL at Degree D

Let $\mathcal{T}^{(D)} := \{\deg \leq D$ monomials$\}$, $T := |\mathcal{T}^{(D)}|$.

- EXTEND: first multiply each p_i of degree d_i by every monomial $\mathbf{x}^{\mathbf{b}} := x_1^{b_1} \cdots x_n^{b_n} \in \mathcal{T}^{(D-d_i)}$ to get equations $\mathcal{R}^{(D)}$.
- LINEARIZE: then reduce $\mathcal{R}^{(D)}$ as a linear system in all the $\mathbf{x}^{\mathbf{b}} \in \mathcal{T}^{(D)}$. We may be able to solve the system or to reduce down to a univariate equation (say in x_1).

$R := |\mathcal{R}^{(D)}|$ and I counts resp. equations and independent equations among $\mathcal{R}^{(D)}$.

Toy XL example over $\mathrm{GF}(7)$

$$\begin{array}{rrrrrrrrrrrr}
p_1: & x^2+ & 4y^2+ & z^2+ & 5xy+ & 2xz+ & 6yz+ & 5x+ & 3y+ & 5z+ & 1= & 0\\
p_2: & 3x^2+ & 2y^2+ & 3z^2+ & 4xy+ & 6xz+ & 2yz+ & 6x+ & 4y+ & 3z+ & 2= & 0\\
p_3: & 2x^2+ & 3y^2+ & 2z^2+ & 5xy+ & & 2yz+ & 4x+ & y+ & z+ & 4= & 0\\
p_4: & 6x^2+ & 3y^2+ & 3z^2+ & & 5xz+ & yz+ & & 5y+ & 2z+ & 2= & 0
\end{array}$$

Here $n = 3$, $m = 4$, we will use $D = 3$, and multiply every equation by $1, x, y, z$ to get $\left(\binom{4}{3}\right) = 20$ monomials (including 1) and $4 \times 4 = 16$ equations.

The Extended Macaulay Matrix

x^2y	x^2z	y^2x	xyz	z^2x	y^2z	z^2y	xy	xz	yz	x^3	x^2	x	y^3	y^2	y	z^3	z^2	z	1
0	0	0	0	0	0	0	5	2	6	0	1	5	0	4	3	0	1	5	1
0	0	0	0	0	0	0	4	6	2	0	3	6	0	2	4	0	3	3	2
0	0	0	0	0	0	0	5	0	2	0	2	4	0	3	1	0	2	1	4
0	0	0	0	0	0	0	0	5	1	0	6	0	0	3	5	0	3	2	2
5	2	4	6	1	0	0	3	5	0	1	5	1	0	0	0	0	0	0	0
1	0	5	2	0	6	1	5	0	5	0	0	0	4	3	1	0	0	0	0
0	1	0	5	2	4	6	0	5	3	0	0	0	0	0	0	1	5	1	0
4	6	2	2	3	0	0	4	3	0	3	6	2	0	0	0	0	0	0	0
3	0	4	6	0	2	3	6	0	3	0	0	0	2	4	2	0	0	0	0
0	3	0	4	6	2	2	0	6	4	0	0	0	0	0	0	3	3	2	0
5	0	3	2	2	0	0	1	1	0	2	4	4	0	0	0	0	0	0	0
2	0	5	0	0	2	2	4	0	1	0	0	0	3	1	4	0	0	0	0
0	2	0	5	0	3	2	0	4	1	0	0	0	0	0	0	2	1	4	0
0	5	3	1	3	0	0	5	2	0	6	0	2	0	0	0	0	0	0	0
6	0	0	5	0	1	3	0	0	2	0	0	0	3	5	2	0	0	0	0
0	6	0	0	5	3	1	0	0	5	0	0	0	0	0	0	3	2	2	0

The Result of Elimination

x^2y	x^2z	y^2x	xyz	z^2x	y^2z	z^2y	xy	xz	yz	x^3	x^2	x	y^3	y^2	y	z^3	z^2	z	1
5	2	4	6	1	0	0	3	5	0	1	5	1	0	0	0	0	0	0	0
0	1	0	5	4	6	1	3	6	5	4	6	4	4	3	1	0	0	0	0
0	0	3	6	0	3	4	1	2	6	0	5	6	2	5	4	0	0	0	0
0	0	0	1	0	2	3	4	5	3	0	2	1	2	4	2	0	0	0	0
0	0	0	0	5	5	5	4	6	5	3	1	3	3	4	6	1	5	1	0
0	0	0	0	0	5	3	2	4	0	0	1	4	1	2	1	0	2	6	0
0	0	0	0	0	0	6	4	2	0	5	1	5	6	5	6	1	0	0	0
0	0	0	0	0	0	0	5	0	2	0	2	4	0	3	1	0	2	1	4
0	0	0	0	0	0	0	0	5	1	0	6	0	0	3	5	0	3	2	2
0	0	0	0	0	0	0	0	0	2	0	4	0	0	3	0	0	2	4	2
0	0	0	0	0	0	0	0	0	0	6	0	6	3	1	0	4	1	6	1
0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	4	3	1
0	0	0	0	0	0	0	0	0	0	0	0	3	1	2	4	2	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	6	0	0	1	5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	3	6	1	5	5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	1	6

Operative Condition and Cost of XL

- XL solves a system if $T - I \leq \min(D, q-1)$.
- Other situations where XL also succeeds are called "pathological terminations". [Our example above is one.]
- Let $E(N, M)$:= the time complexity of elimination on N variables and M equations, then XL takes time $C_{\text{XL}} \approx E(T, R)$.
- Asymptotically $\lg E(T, R) \sim \omega \lg T$, where ω is "the order of matrix multiplication". An often-cited number is 2.376.