Algebraic Cryptanalysis: From Plug-and-Pray Experimental Approach to Constructive Optimization

Nicolas T. Courtois
University College London, UK
Topics

• AC, “Algebraization”, I/O equations method.
  – challenges: AES[Courtois-Pieprzyk], ECDLP[Diem].
• Two Philosophies, 1.+2.
  1. Classical approach: XL, XSL, predictions,
  2. Algebraic coding and optimization. NEW!

Overdefined heuristics, phase transitions
=> how “Degree of regularity” can be reduced
with help of redundancy and oracles!
Key Question

- Can do better than just **contemplate** these transitions?
- Can we explicitly **engineer** phase transitions to happen?

Toy examples:
- ElimLin on Simon block cipher: the force of an asymptotic
- ECC Coding: how redundancy leads to explicit I/O equations => explicitly constructed degree falls.
Planet Earth A.D. 2016

Mafia Economy
Manufacture of Toxic Waste
Debt Slaves
Solution

Travel to a Different Planet!
Which Planet?

1. A planet where a *crypto currency* is at the centre of a more inclusive economy

2. A planet where *quantum computers* break RSA, ECDLP etc…

3. A planet where *algebraic cryptanalysis* breaks AES, ECDLP etc…
Which Planet?

1. A planet with a crypto currency

2. A planet full of quantum computers

3. A planet full of algebraic cryptanalysts

=> all 3 planets have MORE jobs for crypto researchers…
Algebraic Cryptanalysis [Shannon]

Breaking a « good » cipher should require:

“as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type”

[Shannon, 1949]
This Talk

I will review some my research on Algebraic Cryptanalysis in the last 15+ years and try to focus on some “strategic” questions, key principles, big picture

False/Real difficulties.
- Some things are not necessarily a problem and CAN be solved/circumvented
- Some are really a problem and we hit the wall
False or Real Difficulty?

Not necessarily a problem:
1. Lack of algebraic structure, not clear how to even start doing any sort of “algebraic” attack.
2. Many rounds, large systems of equations with lots of variables.
3. NP hard problems – hard instances?

Real difficulties:
2. Bad equations topology / density / connectivity.
Challenges (1)

AES, cf. Courtois-Pieprzyk attack
Challenges (2)

ECDLP, cf. PKC 2016 paper:

Algebraic approaches for the Elliptic Curve Discrete Logarithm Problem over prime fields

Christophe Petit\textsuperscript{1}, Michiel Kosters\textsuperscript{2}, and Ange Messeng\textsuperscript{3}

Comment:

- Another "plug-and-pray" attack which just does not work => bad "equations topology"
Challenges (2)

ECDLP, cf. PKC 2016 paper:

\[ O(e^{2.0*n}) \]

Table 4. Prime case, \( p - 1 \) subgroups

<table>
<thead>
<tr>
<th>( n_1 )</th>
<th>( p )</th>
<th>( D_{av} )</th>
<th>( D_{av}^{corr} )</th>
<th>nbsols</th>
<th>Av. time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4.00</td>
<td>4.00</td>
<td>0.59</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>4.00</td>
<td>4.00</td>
<td>0.79</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>73</td>
<td>4.00</td>
<td>4.00</td>
<td>0.84</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>257</td>
<td>4.01</td>
<td>4.00</td>
<td>1.14</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>1153</td>
<td>4.48</td>
<td>4.00</td>
<td>1.34</td>
<td>0.02</td>
</tr>
<tr>
<td>6</td>
<td>4289</td>
<td>5.00</td>
<td>5.00</td>
<td>1.08</td>
<td>0.13</td>
</tr>
<tr>
<td>7</td>
<td>17921</td>
<td>5.36</td>
<td>5.00</td>
<td>0.99</td>
<td>1.14</td>
</tr>
<tr>
<td>8</td>
<td>65537</td>
<td>5.36</td>
<td>5.00</td>
<td>0.96</td>
<td>9.09</td>
</tr>
<tr>
<td>9</td>
<td>262657</td>
<td>5.78</td>
<td>5.00</td>
<td>1.06</td>
<td>59.87</td>
</tr>
<tr>
<td>10</td>
<td>1051649</td>
<td>6.36</td>
<td>6.00</td>
<td>0.96</td>
<td>454.79</td>
</tr>
<tr>
<td>11</td>
<td>4206593</td>
<td>6.29</td>
<td>6.00</td>
<td>0.76</td>
<td>4975.07</td>
</tr>
</tbody>
</table>
Algebraization: A Tool For Cryptanalysis
MQ Problem

Find a solution to a system of $m$ quadratic equations with $n$ variables over a field/ring.
Cryptography and MQ

Claim: 50 % of all applied cryptography depends on the hardness of MQ.

For example:
RSA is based on MQ with \( m=1 \) and \( n=1 \):
factoring \( N \) \( \iff \) solving \( x^2=C \mod N \).
MQ Problem

Multivariate Version

\[ n \text{ variables} \]
[..] Everybody in mathematics knows that going from one to several variables is an important jump that is accompanied by great difficulties and calls for completely new methods. [...]
MQ Problem over $\text{GF}(2)$

Find a solution (at least one), i.e. find $(x_0, ..., x_{n-1})$ such that:

\[
\begin{align*}
1 & = x_1 + x_0x_1 + x_0x_2 + \ldots \\
0 & = x_1x_2 + x_0x_3 + x_7 + \ldots \\
& \vdots
\end{align*}
\]
Dense MQ

Dense MQ is VERY hard. Best attacks $\approx 2^{0.8765n}$

Also a good candidate for PQ crypto.

$\Rightarrow$ Allows to build a provably secure stream cipher based on MQ directly!

C. Berbain, H. Gilbert, and J. Patarin:

QUAD: A Practical Stream Cipher with Provable Security, Eurocrypt 2005

• open problem: design a provably secure block cipher…
Schneier [Applied Cryptography book]

[...] Any algorithm that gets its security from the composition of polynomials over a finite field should be looked upon with scepticism, if not outright suspicion. [...] 

• Actually any cipher e.g. AES can be seen in this way… Including provably secure QUAD.

• ECDLP is also ‘based’ on hardness of solving polynomials over finite fields.
Algebraization:

Theorem:
Every function over finite fields is a polynomial function.

[can be proven as a corollary of Lagrange’s interpolation formula]

\[
P(X) = \sum_{i=1}^{t} Y_i \cdot \prod_{1 \leq j \leq t, j \neq i} \frac{X - X_j}{X_i - X_j}
\]

False over rings!
Better Method: I/O Degree:

Consider function $f : GF(2)^n \rightarrow GF(2)^m$, $f(x) = y$, with $x = (x_0, \ldots, x_{n-1})$, $y = (y_0, \ldots, y_{m-1})$.

Definition [The I/O degree] The I/O degree of $f$ is the smallest degree of the algebraic relation

$$g(x_0, \ldots, x_{n-1}; y_0, \ldots, y_{m-1}) = 0$$

that holds with certainty for every couple $(x, y)$ such that $y = f(x)$.

These can be used directly for algebraic coding.

[sometimes more equations are needed see the notion of “describing degree”]
Two Philosophies

Two major ways to approach the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Either we expand the number of monomials.
   - work in polynomial ideals, XL F5 etc…

2. Or we expand the number of variables
   - MC-efficient coding
   - algebraic coding
Two Philosophies

In both cases we have two quantities:

- \( R \) = number of equations
- \( T \) = number of monomials

Main idea: \( R \) grows FASTER than \( T \).
Two Philosophies

Two major ways to approach the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Either we expand the number of monomials.
   - work in polynomial ideals, XL F5 etc…

2. Or we expand the number of variables
   - MC-efficient coding
   - algebraic coding

previous research tends to show that this is a “bad idea”.
E.g. XL is preferred to Linearization [Eurocrypt 2000].
In a Way we also have

Two major ways to approach the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Plug-and-pray 😞
   – Build experiments, maybe it works?

2. More constructive!
   – More freedom for the attacker
   – Algebraic optimization problems

Find an “economical” way to expand the problem with redundancy so that the “degree of regularity” decreases the most.
Philosophy 2 Is Not Stupid

Two major ways to approach the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Either we expand the number of monomials.
   - work in polynomial ideals, XL F5 etc…

2. Or we expand the number of variables
   - MC-efficient coding
   - algebraic coding
Glossary

• **MC** = Multiplicative Complexity, informally counting the number of multiplications in algorithms
  – trying to do it with less

• **REMARK:** AES and Simon have INCREDIBLY low MC.
\( x \rightarrow x^{-1} \) \( n=4 \) [Boyar and Peralta 2008-9]

eprint.iacr.org/2009/191/

\[
\begin{align*}
t_1 &= x_1 + x_2 \\
t_4 &= t_1 \times t_3 \\
t_6 &= x_2 + t_2 \\
t_8 &= x_3 + y_2 \\
y_1 &= t_{10} + t_8 \\
y_3 &= t_{12} + t_1 \\
t_2 &= x_1 \times x_3 \\
y_4 &= x_2 + t_4 \\
t_7 &= t_6 \times t_5 \\
t_9 &= t_3 + y_2 \\
t_{11} &= t_3 + t_{10} \\
t_3 &= x_4 + t_2 \\
t_5 &= x_3 + x_4 \\
y_2 &= x_4 + t_7 \\
t_{10} &= x_4 \times t_9 \\
t_{12} &= y_4 \times t_{11}
\end{align*}
\]

**Fig. 1.** Inversion in \( GF(2^4) \).

5 AND 11 XOR
Bit-Slice Gate Complexity

PRESENT S-box

• Naïve implementation = 39 gates
• Logic Friday [Berkeley] = 25 gates
• Our result = 14 gates.

\[T_1 = x_2 \cdot x_1; \quad T_2 = x_1 \& T_1; \quad T_3 = x_0 \cdot T_2; \quad Y_3 = x_3 \cdot T_3; \quad T_2 = T_1 \& T_3; \quad T_1^ = Y_3; \quad T_2^ = x_1;\]
\[T_4 = x_3 \& T_2; \quad Y_2 = T_1 \^ T_4; \quad T_2^ = \neg x_3; \quad Y_0 = y_2 \cdot T_2; \quad T_2| = T_1; \quad Y_1 = T_3 \cdot T_2;\]

Fig. 1. Our implementation of the PRESENT S-box with only 14 gates.
Another S-box – CTC2

Our new design:
Best Paper!

Multiplicative Complexity and Solving Generalized Brent Equations With SAT Solvers
by
Nicolas Courtois, Daniel Hulme, Theodosis Mourouzis


IARIA Board
GOST Attacks with a SAT Solver

eprint.iacr.org/2011/626

<table>
<thead>
<tr>
<th>Rounds</th>
<th>4</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key size</td>
<td>128</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>2 KP</td>
<td>2 KP</td>
<td>3 KP</td>
<td>4 KP</td>
<td>6 KP</td>
<td>≈ 600 KP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cf.</td>
<td>page 13</td>
<td>Fact 15</td>
<td>page 40</td>
<td></td>
<td>page 26</td>
<td></td>
<td>page 26</td>
<td>page 207</td>
<td>page 35</td>
</tr>
<tr>
<td>cf. also</td>
<td>[97]</td>
<td></td>
<td>Fact 16</td>
<td></td>
<td></td>
<td></td>
<td>Fact 13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory bytes</th>
<th>small</th>
<th>$2^{43}$</th>
<th>$2^{46}$</th>
<th>$2^{68}$</th>
<th>small</th>
<th>$2^{69}$</th>
<th>small</th>
<th>small</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>$2^{24}$</td>
<td>$2^{128}$</td>
<td>$2^{127}$</td>
<td>$2^{107}$</td>
<td>$2^{110}$</td>
<td>$2^{94}$</td>
<td>$2^{94}$</td>
<td>$2^{83}$</td>
</tr>
</tbody>
</table>

**Table 1.** Principal attacks on 8 rounds of GOST with 2, 3, 4 and more KP
Two Philosophies

In both case we have two quantities:
\[ R = \text{number of equations} \]
\[ T = \text{number of monomials} \]

Main idea: \( R \) grows FASTER than \( T \).
Which is simultaneously real and “impossible”.

More precisely let
\[ F = \text{number linearly independent equations} \]
\( F \) cannot grow faster than \( T \), but \( R \) can.

The saturation when linear dependencies do appear
because they have to is frequently what we look for.
The principle of XL:

Multiply the initial equations by low-degree monomials:

\[ 1 = x_5 + x_0 x_1 + x_0 x_2 \]

becomes:

\[ x_1 \cdot 1 = x_1 \cdot (x_5 + x_0 x_1 + x_0 x_2) \]

(degree 3 now).
**How XL works:**

Initial system: \( m \) equations and \( n^2/2 \) terms.

Multiply each equation by a product of any \( D-2 \) variables:

- Equations: \( R = m \cdot \binom{n}{D-2} \)
- Terms: \( T = \binom{n}{D} \)

Idea: One term can be obtained in many different ways, \( T \) grows slower than \( R \).

Necessary condition: \( R/T > 1 \) gives \( m \cdot \binom{n}{D-2}/\binom{n}{D} > 1 \) and thus \( D \approx n/\sqrt{m} \).

If sufficient, the complexity of XL would be about \( T^\omega = \left( \binom{n}{n/\sqrt{m}} \right)^\omega \) Sub-exponential !?!
**XL works quite well**

<table>
<thead>
<tr>
<th>n</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>10</td>
<td>14</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>R</td>
<td>110</td>
<td>154</td>
<td>176</td>
<td>187</td>
<td>198</td>
</tr>
<tr>
<td>T</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
</tr>
<tr>
<td>Free</td>
<td>110</td>
<td>154</td>
<td>174</td>
<td>175</td>
<td>175</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20</th>
<th>20</th>
<th>20</th>
<th>20</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>420</td>
<td>840</td>
<td>1050</td>
<td>1260</td>
<td>1365</td>
</tr>
<tr>
<td>1351</td>
<td>1351</td>
<td>1351</td>
<td>1351</td>
<td>1351</td>
</tr>
<tr>
<td>420</td>
<td>840</td>
<td>1050</td>
<td>1260</td>
<td>1350</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33280</td>
<td>66560</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43745</td>
<td>43745</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33280</td>
<td>43744</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: XL simulations for $D = 3$.

- $n$ number of variables.
- $m$ number of equations.
- $D$ we generate equations of total degree $\leq D$ in the $x_i$.
- $R$ number of equations generated (independent or not).
- $T$ number of monomials of degree $\leq D$.
- Free number of linearly independent equations among the $R$ equations.

- XL will work when $Free \geq T - D$. 

N. Courtois 2001-16
The behaviour of XL

It is possible to predict the exact number of linearly independent equations in XL.

<table>
<thead>
<tr>
<th>$D$</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$\text{Min}(T, R)$</td>
</tr>
<tr>
<td>4</td>
<td>$\text{Min}(T, R - \binom{m}{2} - m)$</td>
</tr>
<tr>
<td>5</td>
<td>$\text{Min}(T, R - (n + 1)\binom{m}{2} - (n + 1)m)$</td>
</tr>
<tr>
<td>6</td>
<td>$\text{Min}(T, R \cdot \binom{n}{2} + \binom{n}{1} + \binom{n}{0} \cdot \binom{m}{2} + \binom{m}{1} + \binom{m}{3} + m^2)$</td>
</tr>
</tbody>
</table>
And “XSL”

“XSL is not an attack, it is a dream“

Vincent Rijmen, AES designer
The XL idea:

Multiplying the equations by one or several variables.
The XSL variant:

Multiplying the equations by one or several monomials (out of monomials present).
XL and XSL

Both work well, they operate a specific phase transition.

The curve reaches another curve and stays there.

<table>
<thead>
<tr>
<th>n</th>
<th>24</th>
<th>24</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>16</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>R</td>
<td>37200</td>
<td>62775</td>
<td>74400</td>
</tr>
<tr>
<td>T</td>
<td>55455</td>
<td>55455</td>
<td>55455</td>
</tr>
<tr>
<td>F</td>
<td>33800</td>
<td>53325</td>
<td>55454</td>
</tr>
</tbody>
</table>
Simulations on a “Toy Cipher”

Free/(T-T’) - XSL works for up to 16 rounds.
Three Stages

Algebraic attacks on block ciphers work in 3 stages:

1. **Write good equations** – overdefined, sparse or both.

2. **Expand** - to obtain a very overdefined system.

3. **Solve** at saturation / phase transition point.
Reinvented in 2006

Algebraic attacks on block ciphers today:
1. Write good equations – overdefined, sparse or both.

2. Expand - avoid / minimise impact of...

   • ElimLin alone and T’ method. Very powerful.
Reinvent it in 2016

Algebraic attacks on block ciphers today:

1. Write EVEN BETTER equations –
   => even more overdefined $R/T \approx 1$.
   => redundant “algebraic codes”
   => work on equations topology/density.

2. Expand - avoid / minimise impact of…

Part 1.

Find good equations: such that:

\[
\frac{R}{T} = \frac{1}{4} \text{ or so}.
\]
Can do Better?

Find better equations: such that:

\[ R \approx 1 \text{ already} \]

\[ T + \text{questions of equations density and topology} \]
The Redundancy+Oracle idea:

We can decrease the “regularity degree” by adding variables AND new facts coming from an oracle.
The Redundancy+Oracle idea:

Example 1: ElimLin.
Oracle=encryption oracle.

Example 2: EC point splitting.
Oracle=block box EC point addition.
A Thought Experiment

EC point splitting.

\[ P_1 + P_2 = Q \]

+ extra equations to code a “factor basis”.
More Overdefined

Same point splitting Pb.

\[
\begin{align*}
P1 + P2 &= Q \\
P1 + D &= P1' \\
P2 + E &= P2' \\
P1' + P2' &= (Q + D + E)
\end{align*}
\]

- added 2 constants D, E
- 2 new vars
  - linear ECC code expansion of vars
- 3 new eqs!

+ same x extra equations to code a “factor basis”.

• strict improvement: \(2+x \rightarrow 3+x\)
The Same Happens in ElimLin

By “magic” the regularity degree decreases with $K$

$K =$ data complexity ($K$ KP or $K$ CP).
Asymptotic Aspects

Something VERY disturbing happens in ElimLin.

How quickly $R$ or/and $F$ grow when $K$ increases?
Algebraic Attacks on Block Ciphers

8 Rounds of Simon 64/128

The Impossible Happens

Remark: In the long run it CANNOT be super-linear. \( \rightarrow \text{linear when } K \rightarrow \infty \)

However in the long run the cipher is broken for a fixed value of \( K \).
Redundancy Up and Downs

#Variables at the end of ElimLin when $K$ grows.
Back to EC Point Splitting Questions

Can we also produce a system of equations with fast growth due to redundant ECC coding?

\[ P1 + P2 = Q \]

+ extra equations to code a “factor basis”.

N. Courtois 2001-16
Elaborate Prototype [eprint 2016/704]

\[ P1 + P2 = Q \]

+ a very new unique method to code a “factor basis”
Are We Making Any Progress?

Possibly this approach is stupid and NOT as good as traditional highly-optimized Gröbner basis approach.

=> Everybody uses Semaev polynomials + "plug-and-pray" GB.

\[ S_3(x_1, x_2, x_3) = (x_1-x_2)^2x_3^2 - 2[(x_1+x_2)(x_1x_2+A) + 2B]x_3 + (x_1x_2-A)^2 - 4B(x_1+x_2) \]
Are We Making Any Progress?

Possibly this approach is stupid and NOT as good as traditional highly-optimized Gröbner basis approach.

=> Even if so, I believe this approach is BETTER because we avoid "plug-and-pray" and construct our degree falls and other equations more explicitly. More control/insights on what we do.
Merits of Redunancy

Linear ECC Code expansion

=>

NEW very regular families of I/O equations which we can construct explicitly
Example: New ECC I/O relations

D73 Theorem [Courtois 2016]

Theorem 4.2.1 (D73 Theorem). We consider the following set of variables on EC, a special form of ECC Code with 3 inputs and 7 outputs for any Weierstrass elliptic curve modulo a large $P$.

\[
\begin{align*}
P1 & \quad P2 & \quad P1 + P2 \\
(P1, P2, P3) & \mapsto P1 + P3 & P2 + P3 & P1 + P2 + P3 \\
& & P3
\end{align*}
\]

If all the 7 points are distinct from the ECC neutral element $\infty$ we have:

\[
sx1*sx2*(sx23-sx13) + sx1*sx3*(sx12-sx23) + sx2*sx3*(sx13-sx12) + sx123[sx1*(sx13-sx12)+sx2*(sx12-sx23)+sx3*(sx23-sx13)] = 0
\]