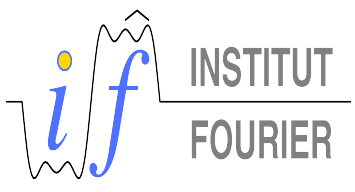
M2 Cybersecurity (Univ. Grenoble Alpes/Grenoble INP)

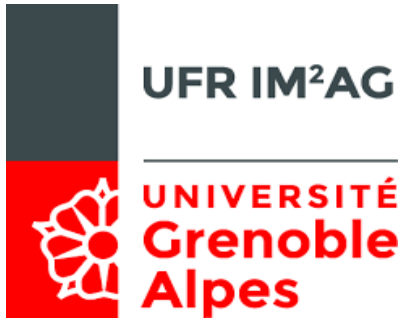
Cryptographic Mechanisms (Part 3)

The Post-Quantum Cryptography Era

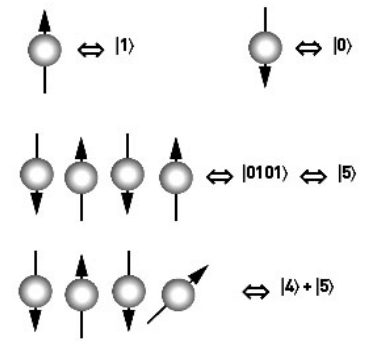
v1.1, December 2016

Philippe Elbaz-Vincent





RSA, Classical DLP, ECC and Quantum computers



In 1994, Peter Shor (`https://arxiv.org/abs/quant-ph/9508027`) gave algorithms for the factorization of integers and for solving the DLP running in quantum polynomial time and requiring a "big enough" quantum computer. Those algorithms have been improved during the past years in both space requirement and running time.

In 2003, it has been shown by John Proos and Christof Zalka (`https://arxiv.org/abs/quant-ph/0301141`) that we can solve ECDLP in Quantum polynomial time. Furthermore, they have shown that we can solve it with less qubits and faster than the (quantum) factorization of an equivalent RSA modulus.

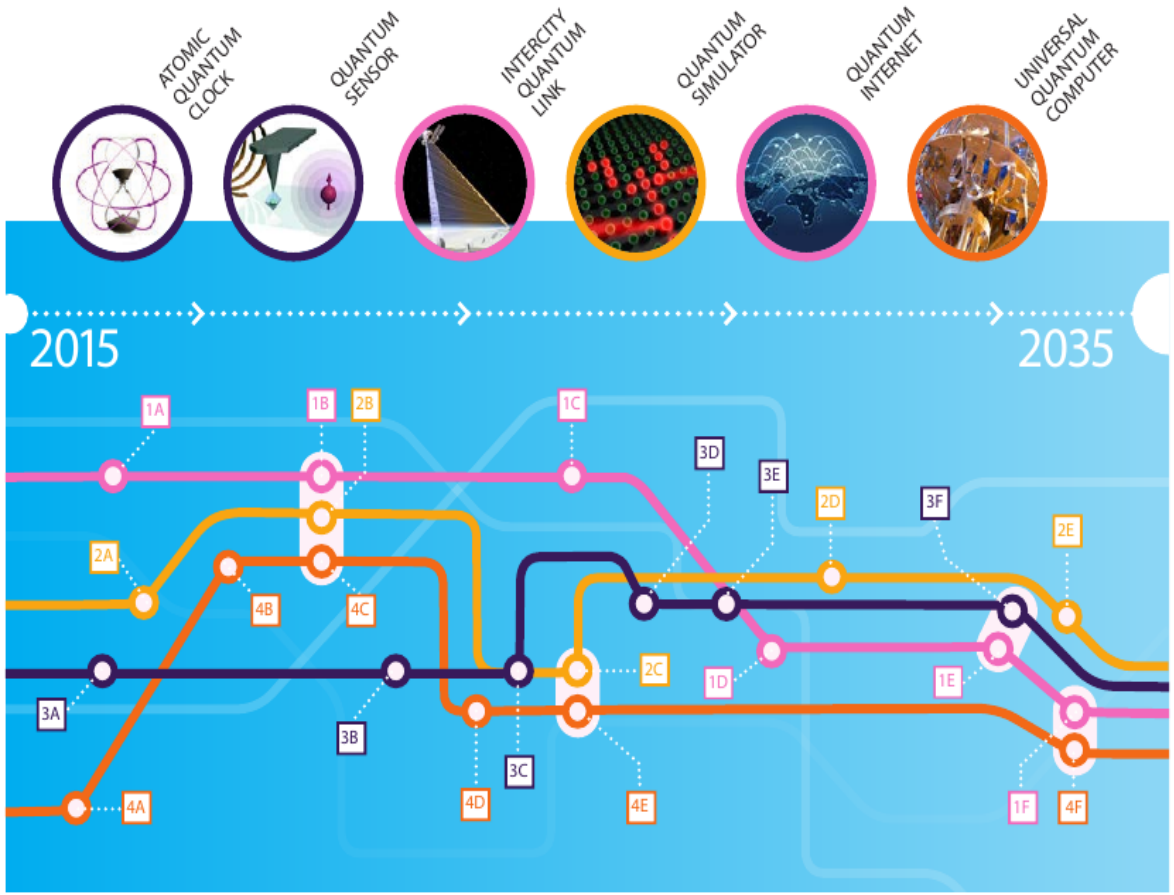
☞ Conclusion : A good public key cryptosystem for the postquantum era is (still ?) to be discovered...

A Quantum Computer Not universal...but Real !

(C) 2016 D-WAVE.

☞ It is not a "Quantum Turing machine", but it can perform certain type of combinatorial optimizations 100 millions of time faster than a conventinal supercomputer !
We still don't know if the D-WAVE One X^2 can speed up some cryptanalysis techniques...

The Post-Quantum Timeline (?)



(C) 2016 «Quantum Manifesto»

☞ **In 2017, there will be an international call (managed by the NIST) for Post-quantum cryptosystems...**

The (old) new era ?

- Lattice based cryptosystems : NTRU, Merkle/Hellman, Ajtai/Dwork, Goldreich/Goldwasser/Halevi, BLISS,...
- Multivariate cryptosystems,
- McEliece cryptosysems (based on coding theory).

NTRU : The history

NTRU (formerly *www.ntru.com*) is a (patented) public key cipher proposed in 1996 by Jeffrey Hoffstein, Jill Pipher and Joseph Silverman (a wellknown expert on ellipic curves !) while they were (and still are...) at Brown University (USA). Since then they founded the company NTRU.
The protocoles associated to NTRU are now mainly standardized in IEEE 1363.1 and X9.98 financial services standard (but most of the cryptographic mechanisms are patented ! with "free use" options).


J. Silverman


J. Pipher


J. Hoffstein

(cf. http ://www.brown.edu/Administration/George_Street_Journal/vol25/25GSJ01e.html)

NTRU has been bought by SecurityInnovation (`https://www.securityinnovation.com/`) in 2009.
It has been growing during the past years due to its stong relationships with homomorphic encryption.

NTRU : The motivations

The interest of NTRU comes from the following properties :

- "Short keys" and signatures,
- Very fast (in particular for embedded systems with low power comsumption),
- Easy to implement,
- The difficult problem behind the trapdoor function is strongly connected to a problem known to be NP-Hard for which only exponential algorithms are known (except for rare instances...)... even for a quantum computer.

Unfortunately, the cryptosystem had a difficult childhood

- evolution of the encryption scheme (due to bad choices of padding),
- many changes in the signature scheme (several versions have been broken easily),
- Theoretical existence of undecipherable messages (*this is not a problem in practice*)

Nevertheless, the mathematics behind the cryptosystem are solids.

NTRU : The framework

The NTRU cryptosystem needs the following parameters :

- a positive integer N, called *degree*,
- a positive integer q, called *large modulus*,
- a positive integer or a polynomial p, called *small modulus*,
- sets of integer polynomials $\mathcal{D}_f$, $\mathcal{D}_g$, called *spaces of private keys*,
- a set of integer polynomials $\mathcal{D}_m$, called *messages space*,
- a set of integer polynomials $\mathcal{D}_r$, called *Blinding Value Space*,
- a centering function (denoted center) allowing us to perform the reduction modulo q during encryption.

NTRU : The framework

We set

$$R = \mathbb{Z}[X]/(X^N - 1)\,,$$

and

$$R_q = \mathbb{Z}/q\mathbb{Z}[X]/(X^N - 1)\,,$$

if q is an integer and more generally $R_q = R/qR$ if q is a polynomial. Notice that, mathematically speaking, if q is an integer $R/qR \cong \mathbb{Z}/q\mathbb{Z}[X]/(X^N - 1)$.
We will denote $*$ the product of polynomials in R. A polynomial $a_0 + a_1X + \cdots + a_{N-1}X^{N-1}$ will be identified to the vector $[a_0, a_1, \ldots, a_{N-1}]$ (practical notation for storage when the polynomials are dense).

NTRU : Playing with truncated polynomials

Performing operations in R (or R_q) is just multiplying polynomials of degree $< N$ and reducing modulo $X^N - 1$.

☞ **But**, reducing mod $X^N - 1$ is like replacing X^N by 1 in R, then replacing X^{N+1} by X in R, and more generally X^{N+k} by X^k in R if $0 \leqslant k < N$. Then for $k = N$, we restart the substitution.

For instance, $2X^5 + 3X^4 + 1 \bmod X^3 - 1$ gives in R ; $2X^2 + 3X + 1$, because $X^5 = X^2 \mod X^3 - 1$ and $X^4 = X \mod X^3 - 1$.

Examples : Take $N = 5$ and let $a = 1 + 2X + 3X^4$ and $b = 2 + X^2 + 8X^3$. Then

$$\begin{aligned} a * b &= 24X^7 + 3X^6 + 22X^4 + 10X^3 + X^2 + 4X + 2\,, \\ &= 24X^2 + 3X + 22X^4 + 10X^3 + X^2 + 4X + 2\,, \\ &= 22X^4 + 10X^3 + 24X^2 + 7X + 2\,. \end{aligned}$$

Where, for the last equality, we used the usual addition for polynomials of degree $< N$ (here with $N = 5$).

NTRU : Playing with truncated polynomials

Exercises : Take $N = 7$ and compute $a + b$ and $a * b$ with $a = 3 + 2X^2 - 3X^4 + X^6$, $b = 1 - 3X + X^2 + 2X^5 - X^6$. Do the same for $N = 3$ and $a = 1 + X^2$ and $b = 1 - X^2$.

The following is the general formula for multiplying polynomials in R :

$$a * b = c_0 + c_1 X + c_2 X^2 + c_3 X^3 + \cdots + c_{N-2} X^{N-2} + c_{N-1} X^{N-1} ,$$

where the kth coefficient c_k is given by the formula

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_k b_0 + a_{k+1} b_{N-1} + a_{k+1} b_{N-2} + \cdots + a_{N-1} b_{k+1} .$$

In short mathematical notation, it is simply

$$c_k = \sum_{i+j=k \mod N, 0 \leqslant i < N} a_i b_j .$$

NTRU : Playing with truncated polynomials

In the case of R_q with q a positive integer (which will be often the case), we need to perform a modulo q reduction of the coefficients of the polynomials. For instance, if $q = 3$, $a = 3 + 2X^2 - 3X^4 + X^6$ will become $2X^2 + X^6$ in R_3 with $N = 7$.

An other operation that we will use is *inversion in* R_q : if a is a polynomial, the inverse of a in R_q, *if it exists*, is a polynomial A such that

$$a * A = 1 \mod q\,.$$

Example : Take $N = 7$, $q = 5$ and $a = 3 + 2X^2 - 3X^4 + X^6$. Then a is invertible in R_q and its inverse is $A = 4X^6 + X^5 + 3X^4 + X^3 + 2X^2 + 3X + 3$. Indeed,

$$\begin{aligned} a * A &= 4X^{12} + X^{11} - 9X^{10} - 2X^9 + X^8 + 2X^7 + 15X^6 - 4X^5 + 4X^4 + \\ &= 15 * X^6 + 5 * X^4 + 10 * X^2 + 10 * X + 11 \mod X^7 - 1\,, \end{aligned}$$

But a is not invertible in R_3 (i.e., $q = 3$).

NTRU : Invertibles elements of R_q

We will see that the invertible elements of R_q play an important role in the scheme. Here is a way to detect the non-invertible elements. The map $\rho : R_q \to \mathbb{Z}/q\mathbb{Z}$, $f \mapsto f(1)$ is well defined map which is in fact a ring homomorphism. This induces a well-defined group homomorphism $\rho : R_q^\times \to (\mathbb{Z}/q\mathbb{Z})^\times$. But the invertibles of $\mathbb{Z}/q\mathbb{Z}$ are wellknown. We deduce that if $f(1)$ is not invertible in $\mathbb{Z}/q\mathbb{Z}$ then f is non-invertible in R_q. Furthermore, we can show that if N is prime of the form $2M + 1$ with M prime, then the probability to pick a non-invertible random f is neglectible.
It is why, we often choose N prime of this form. Furthermore, we have better resistance to "combinatorial attacks" with such parameters.

NTRU : The centered norm

The *centered norm* of a polynomial (or a vector) a, denoted $||a||$, is defined by

$$||a|| = \sum_{i=0}^{N-1} a_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} a_i\right)^2 .$$

We define the width of a polynomial (or a vector), denoted $\mathrm{Width}(a)$, by

$$\mathrm{Width}(a) = \max(a_0, \ldots, a_{N-1}) - \min(a_0, \ldots, a_{N-1}) .$$

NTRU : Construction of public and private keys

1. We choose randomly some "small" polynomials f and g of $\mathcal{D}_f$, $\mathcal{D}_g$ respectively,
2. We set f_q (resp. f_p) the inverse of f in R_q (resp. R_p). We check that g is invertible in R_q.
3. The public key is given by $h = p * g * f_q \mod q$ and the private key is given by the pair (f, f_p).

NTRU : Encryption/Decryption

Encryption :

1. We choose randomly a "small" polynomial $r \in \mathcal{D}_r$.
2. If m is the message to encrypt, we set $c = r * h + m \mod q$.

Decryption :

1. We compute $a = \mathrm{center}(f * c)$, which reduce the coefficients in the range $[A, A + q - 1]$, where A is part of the centering data.
2. We compute $f_p * a \mod p$.

NTRU : Why the scheme works

The following identities explain why the scheme works

$$\begin{aligned} f * c &= f * p * r * h + f * m\,, \\ &= f * p * r * g * f_q + f * m\,, \\ &= p * r * g + f * m \mod q\,. \end{aligned}$$

The centering function will transform the last equality (modulo q) in an equality in R.

We can multiply by f_p and reduce modulo p.

$$\begin{aligned} f_p * a &= f_p * p * r * g + f_p * f * m\,, \\ &= m \mod p\,. \end{aligned}$$

The space $\mathcal{D}_m$ must ensure that we can identify m with $m \mod p$.
In the case where p is an integer, we can solve the centering problem by taking the coefficients of m, r, f and g between $-p/2$ and $+p/2$.
From a practical point of view, we choose integers d_f, d_g and d_r such that the respective polynomials have ; d 1s (ones) and possibly d -1s (minus ones) and 0 elsewhere (with $d = d_f$ or d_g or d_r).

NTRU : An example

Take $N = 11$, $q = 32$, $p = 3$. Consider $d_f = 4$, $d_g = d_r = 3$.

$$f = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$
$$g = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}\,.$$

We have

$$f_p = 1 + 2X + 2X^3 + 2X^4 + X5 + 2X^7 + X^8 + 2X^9$$
$$f_q = 5 + 9X + 6X^2 + 16X^3 + 4X^4 + 15X^5 + 16X^6 + 22X^7 + 20X^8 + 18X^9 + 30X^{10}\,.$$

We can compute h

$$h = pf_q * g = 8 + 25X + 22X^2 + 20X^3 + 12X^4 + 24X^5 + 15X^6 + 19X^7 + 12X^8 + 19X^9 + 16X^{10}\,.$$

Suppose now that we want to send the plaintext

$$m = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}\,,$$

to the recipient with the public key h (and private key (f, f_p)). We choose a random r with adequate parameters

$$r = -1 + X^2 + X^3 + X^4 - X^5 - X^7\,.$$

We compute

$$\begin{aligned} c &= r * h + m \\ &= 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10} (\mod 32). \end{aligned}$$

The recipient who get c performs the following operations for decryption. We use as center the range $[-q/2-1, +q/2]$ for the coefficients of polynomials. Notice that q being even, modulo q we can identify $q/2$ and $-q/2$. We choose $q/2$.
We get

$$a = 3-7X-10X^2-11X^3+10X^4+7X^5+6X^6+7X^7+5X^8-3X^9-7X^{10}\,.$$

Denote b the reduction of a modulo p. We have

$$b = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \mod 3\,,$$

and then

$$f_p * b = -1 + X^3 - X^4 - X^8 + X^9 + X^{10} \mod 3\,,$$

and we retrieve m.

NTRU : The trapdoor function

The trapdoor of NTRU is given by the function

$$F : \mathcal{D}_m \times \mathcal{D}_r \to R_q$$
$$(m, r) \mapsto m + r * h\,.$$

NTRU : Attacks and security parameters

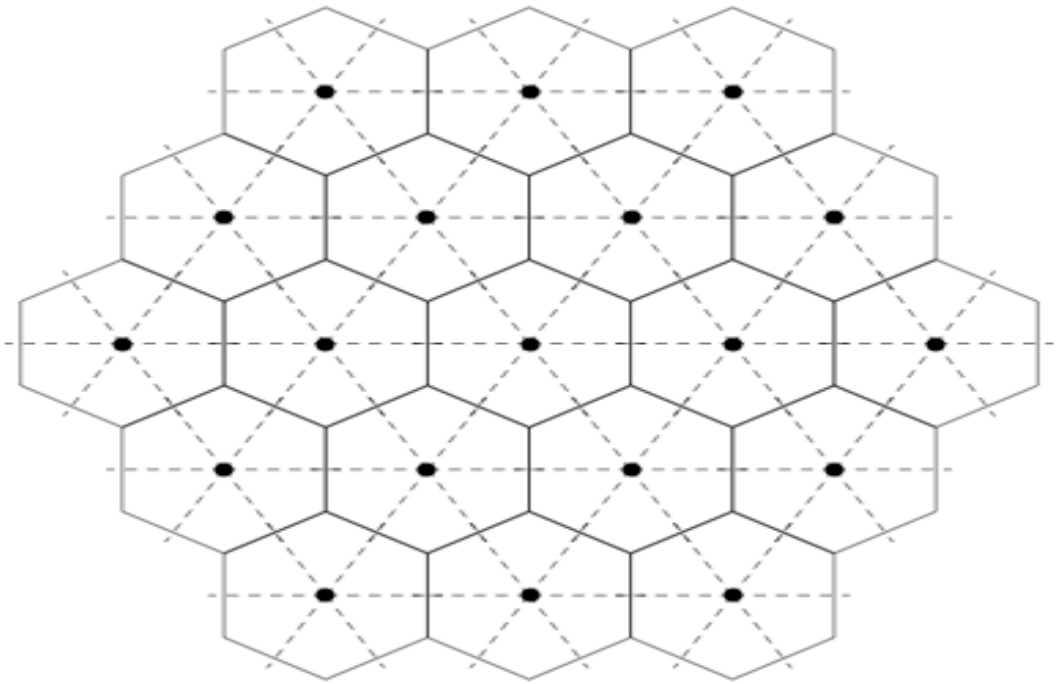
We have the following security constraints corresponding to the generic attacks on NTRU :

- Resistance to lattice reductions (and CVP, SVP) : we can associate to h (the public key) the following lattice

$$L_h = \{(u, v) \in R^2,\ v = h * u/p \mod q\}.$$

We have $\dim(L_h) = 2N$ and we notice that (f, g) is a point of the lattice. We can estimate its length (those data are public) and, for instance, when $f = 1 + pF$ we can settle of type SVP (Shortest Vector Problem) or CVP (Closest Vector Problem), by using LLL for lattice reduction. But, CVP is NP–Hard (Emde Boas, 1981) and SVP is also NP–Hard modulo "randomised reduction" (Ajtai, 1996). We do not known if breaking NTRU is equivalent to CVP or SVP. Notice that both problem are difficult for a quantum computer.

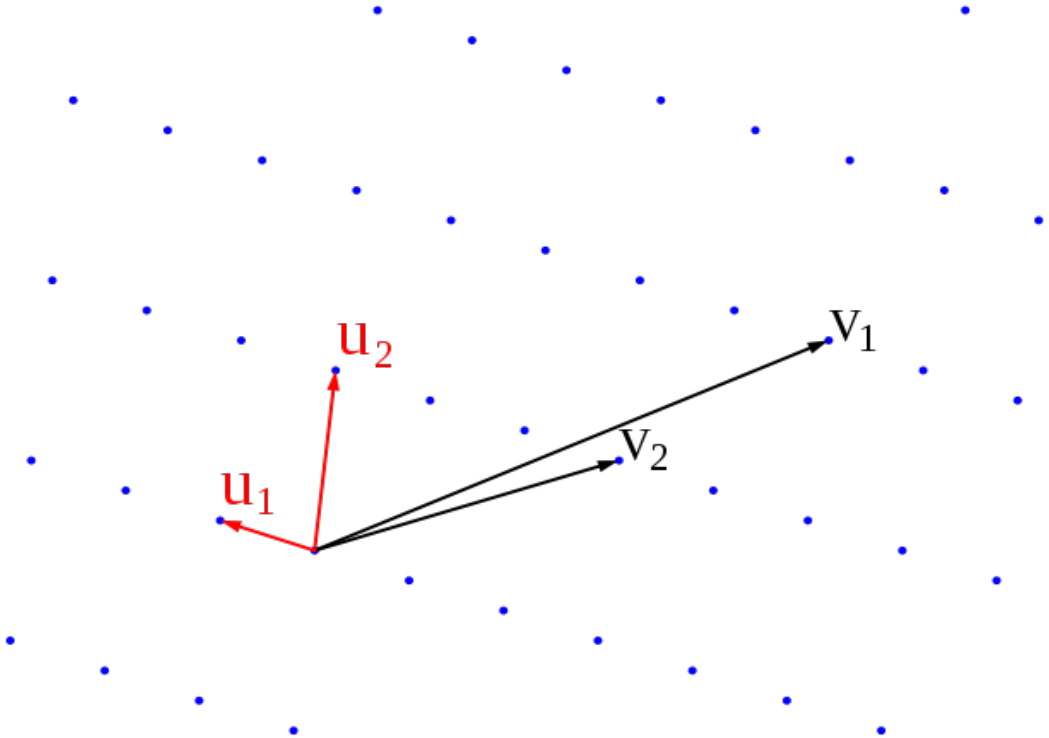
Simple example of lattice



the "honeycomb lattice"

☞ Lattices are "maximal discrete subgroups" of $\mathbb{R}^n$, meaning they are isomorphic to $\mathbb{Z}^n$. It is a regular grid of points in $\mathbb{R}^n$ with usually lots of symmetries.

Simple example of lattice reduction



example of two-dimensional lattice reduction

In order to solve SVP or CVP, we perform a lattice reduction in order to find a basis with shorter vectors. This is similar to the Gram-Schmidt method used for orthogonalisation.

Attacks on NTRUsign

- In 2006, Nguyen and Regev have presented an attack on a reduced version of NTRUSign-251 (without perturbation) using 90000 signatures in order to retrieve the secret key. Recently, they have shown that 400 signatures are enough. This result shows that *perturbation is necessary to the security of NTRUSign*. Notice that the previous scheme (NSS, 2002) was broken shortly after publication.

☞ lack of confidence from the community

Key sizes for NTRU

From generic attacks and other constraints, we have the following security parameters for NTRU.

	N	q	p
Moderate Security	167	128	3
Standard Security	251	128	3
High Security	347	128	3
Highest Security	503	256	3

We usually consider that RSA4096 is equivalent (in term of security) to NTRU503.
Exercise : Compare the length of public and private keys for RSA4096 and NTRU503.

BLISS

to be completed...

PARI/GP and `fpLLL`

For computing LLL we will use PARI/GP and we will also use the software `fpLLL`, available here

`http://perso.ens-lyon.fr/xavier.pujol/fplll/`

☞ You will have to familiarise yourself with fpLLL.

The Knapsack (Subset Sum) Problem

Let

$$a = (a_1, a_2, \ldots, a_n)$$

be a list of positive integers.
The Knapsack (Subset Sum) Problem is formulated as follows :
Given a target integer t, determine if there are values x_1, x_2,..., $x_n \in \{0, 1\}$ satisfying

$$x_1 a_1 + x_2 a_2 + \cdots + x_n a_n = t.$$

If this decision problem can be solved efficiently, then we can actually find x_1,..., x_n. For example, to find a value for x_1, it suffices to determine if either

$$x_2 a_2 + \cdots + x_n a_n = t$$

or

$$x_2 a_2 + \cdots + x_n a_n = t - a_1$$

has a solution

How Hard is the General Knapsack Problem ?

The general Knapsack Problem is an NP-complete problem, so it should be in general very hard (but again there could be easy instances).

The trivial solution method is to try all 2^n possible values for $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

A better method is to sort the following two sets and look for a collision :

$$\left\{ \sum_{j \leqslant \frac{n}{2}} x_j a_j \, , \, x_j = 0,1 \right\} .$$

$$\left\{ t - \sum_{j > \frac{n}{2}} x_j a_j \, , \, x_j = 0,1 \right\} .$$

This takes $O(n2^{\frac{n}{2}})$ operations.

There is still no algorithm known that solves all Knapsack Problems in fewer than $O(2^{\frac{n}{2}-\varepsilon})$ operations.

Building a Cryptosystem from a Knapsack Problem

There is a natural way to try to build a cryptosystem based on a hard knapsack problem.

Public key	$a = (a_1, a_2, \ldots, a_n)$
Plaintext	$x = (x_1, \ldots, x_n) \in \{0,1\}^n$
Ciphertext	$t = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n$

The problem with this approach is that in order to decipher the message, one needs to solve the knapsack problem (which is difficult). Furthermore, there could be several solutions to the equation.

☞ We need to introduce some sort of trapdoor in order to make the solving of the knapsack problem easier and also introduce some "rigidity" in order get unicity of the decryption.

Building a Cryptosystem from a Knapsack Problem (1/2)

Some knapsack problems are easy to solve.
Suppose the "weights" $a_1, \ldots, a_n$ are superincreasing, namely

$$a_j > a_1 + a_2 + \cdots + a_{j-1} \text{ for each } 1 < j \leqslant n.$$

Then we can easily find x_n, since

$$x_n = 1 \text{ if and only if } t > a_1 + a_2 + \cdots + a_{n-1}.$$

Having determined x_n, we are reduced to the lower dimensional knapsack problem

$$x_1 a_1 + \cdots + x_{n-1} a_{n-1} = t - x_n a_n,$$

so we can recover $x_{n-1}, \ldots, x_1$ recursively.

☞ Unfortunately, since $a_1, \ldots, a_n$ are public knowledge, an attacker can decipher the message as well.

Building a Cryptosystem from a Knapsack Problem (2/2)

The solution proposed by Merkle and Hellman in 1978 was to conceal the superincreasing set

$$a = (a_1, a_2, \ldots, a_n)$$

by some sort of invertible transformation (in fact a permutation). Then the private key will be the transformation applied to a.

To illustrate the general method, we will describe Merkle and Hellman's original single-transformation system and show how it can be viewed as a lattice problem and solved using lattice reduction.

Merkle and Hellman and others subsequently proposed more complicated knapsack-based cryptosystems, but so far, all practical systems have been broken using lattice reduction methods.

The Merkle/Hellman Knapsack Cryptosystem

Private Key : Superincreasing sequence $b_1, \ldots, b_n$ with $b_1 \approx 2^n$, $\ldots$, $b_n \approx 2^{2n}$, and $M, W \in \mathbb{Z}$ with $M > b_1 + \cdots + b_n$, $\gcd(M, W) = 1$, and a permutation σ of the integers $\{1, \ldots, n\}$.

Public Key : public key is $a_1, \ldots, a_n$ with $a_j = W b_{\sigma(j)} \mod M$.

Plaintext : $x = (x_1, \ldots, x_n) \in \{0, 1\}^n$.

Ciphertext : $t = x_1 a_1 + \cdots + x_n a_n$.

Decryption : We compute

$$c = W^{-1} t \mod M = \sum_{j=1}^{n} x_{\sigma^{-1}(j)} b_j \mod M .$$

The modulus M is large, so c exactly equals the sum. Also $b_1, \ldots, b_n$ is superincreasing, so we can easily solve this knapsack problem and recover the plaintext x.

Converting a Knapsack Problem to a Lattice Problem

Consider a knapsack problem associated to the n-tuple a to be solved :

$$t = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n \,. \qquad (*)$$

Define a lattice L_a (or $L_{a,t}$ if we want also to emphasise on the target) using the rows of the matrix $(n+1) \times (n+1)$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & a_1 \\ 0 & 1 & 0 & 0 & \cdots & 0 & a_2 \\ 0 & 0 & 1 & 0 & \cdots & 0 & a_3 \\ & \vdots & & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 1 & a_n \\ 0 & 0 & 0 & 0 & \cdots & 0 & -t \end{pmatrix}$$

If $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ solves $(*)$, then $v = (x_1, \ldots, x_n, 0) \in L_a$. By construction, v is a short vector of L_a (this can be seen using the Gaussian heuristic, see below), Thus we can hope to find it using LLL (or one of its variant).

Why Attempt To Use Lattices To Build Cryptosystems?

The reason that the Merkle/Hellman and other knapsack cryptosystems attracted attention is because they are much faster than RSA, often by a factor of 10 to 100 (see also how fast is NTRU). They are even faster than ECC (which is slightly faster than RSA due to its key sizes).
Recall that, if N and d are n bit numbers, it takes approximately

$$n^3 \text{ steps to compute } a^d \mod N .$$

But knapsack encrypt/decrypt take only about n^2 steps.

However, the speed advantages available from lattice operations combined with the fact that SVP and CVP are well-studied hard problems make it worth looking for other constructions whose security depends more directly on SVP and CVP.

☞ Moreover, SVP and CVP are also difficult problems for quantum computers...

The Ajtai/Dwork Lattice Cryptosystem

- Ajtai and Dwork (1995) described a lattice-based public key cryptosystem whose security relies on the difficulty of solving CVP in certain class of lattices $\mathcal{L}_{AD}$.
- They proved that breaking their system in the average case (i.e., for a randomly chosen lattice of dimension m in $\mathcal{L}_{AD}$) is as difficult as solving SVP for all lattices of dimension n (for a certain n that depends on m).
- This "average case"/"worst case" equivalence is a theoretical cryptographic milestone, but unfortunately the Ajtai/Dwork cryptosystem is impractical.
- Inspired by the work of Ajtai and Dwork, a more practical lattice-based cryptosystem was proposed in 1996 by Goldreich, Goldwasser, and Halevi (GGH).

The GGH Public Key Cryptosystem

Key Generation : Choose a lattice L and
Private Key $= \{v_1, \ldots, v_n\}$ a good (short) basis,
Public Key $= \{w_1, \ldots, w_n\}$ a bad (long) basis.

Encryption : The plaintext m is a binary vector. Also choose a small random "perturbation" vector r. The ciphertext is
$e = m_1 w_1 + m_2 w_2 + \cdots + m_n w_n + r$.

☞ *Note that the ciphertext vector e is not in the lattice L.*

Decryption : Find a vector u in L that is closest to e. If r is small enough, then $u = m_1 w_1 + \cdots + m_n w_n$, so solving CVP for e in L will recover m. The private good basis can be used to find u. First write $e = \mu_1 v_1 + \cdots + \mu_n v_n$ using real $\mu_1, \ldots, \mu_n \in \mathbb{R}$. Then round $\mu_1, \ldots, \mu_n$ to the nearest integer :

$$\lceil \mu_1 \rfloor \, v_1 + \cdots + \lceil \mu_n \rfloor \, v_n \text{ will equal } u.$$

$$\lceil \mu \rfloor = \left\lfloor \mu + \frac{1}{2} \right\rfloor .$$

GGH versus LLL

The security of GGH rests on the difficulty of solving CVP using a highly nonorthogonal basis. The LLL lattice reduction algorithm finds a moderately orthogonal basis in polynomial time. In practice, if $n = \dim(L) < 100$, then LLL easily finds a good enough basis to break GGH. Even for $n < 200$, variants of LLL give a practical way to break GGH.

The public key for GGH is a basis for L, so

$$\text{Size of GGH Public Key} = O(n^2) \text{ bits.}$$

GGH is currently secure for (say) $n = 500$, but with a public key of ≈ 250000 bits (more or less $30KB$) the cryptosystem is impractical. The NTRU Public Key Cryptosystem solves this problem by using a type of lattice whose bases can be described using only $\frac{1}{2} n \log_2(n)$ bits.

The NTRU lattice

We can associate to h (the public key) the following lattice

$$L_h = \{(u, v) \in R^2,\ v = h * u/p \mod q\}.$$

We have $\dim(L_h) = 2N$ and we notice that (f, g) is a point of the lattice. More explicitly, the lattice L_h is given by the rows of the matrix

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{pmatrix}$$

The NTRU lattice (continued)

We denote the previous matrix by $\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$. We have the following equality

$$\text{The lattice } L_h = \text{row span of } \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} = \text{row span of } \begin{pmatrix} f & g \\ * & * \end{pmatrix} .$$

$\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$ is the **long (bad) Public basis**

$\begin{pmatrix} f & g \\ * & * \end{pmatrix}$ is the **short (good) Private basis**.

Key sizes for NTRU

Recall that from generic attacks and other constraints, we have the following security parameters for NTRU.

	N	q	p
Moderate Security	**167**	**128**	**3**
Standard Security	**251**	**128**	**3**
High Security	**347**	**128**	**3**
Highest Security	**503**	**256**	**3**

We have the corresponding security : RSA4096 $\approx$ NTRU503 and RSA1024 $\approx$ NTRU251 (this illustrate the non-linearity in the dimensions of the lattice when compared to RSA). Notice also, that for NTRU251, its private key size is 1757 bits and 384 bits for the public key, while for NTRU503, the private key size is 4024 bits and 1000 bits for the public key. We can see that private key sizes are comparable, but NTRU is clearly faster in encryption/decryption time.

Gaussian Heuristic and SVP/CVP

If L is a "random" lattice of rank n, how long would we expect its shortest vector to be ?

And if $t \in \mathbb{R}^n$ is a "random" target point, how far would we expect the closest lattice point to be to t ?

The Gaussian Heuristic answers these questions.

The Gaussian Heuristic : The shortest nonzero vector in a "random" lattice L of rank n has length approximately

$$\lambda_1(L) = \min_{v \in L, v \neq 0} ||v|| \approx \sqrt{\frac{n}{2\pi e}} |L|^{\frac{1}{n}} .$$

Similarly, a "random" target vector $t \in \mathbb{R}^n$ satisfies

$$\min_{v \in L} ||v - t|| \approx \sqrt{\frac{n}{2\pi e}} |L|^{\frac{1}{n}} .$$

Gaussian Heuristic and NTRU Lattices

The NTRU lattice L_h has dimension $n = 2N$ and its basis is an upper diagonal matrix whose diagonal is half 1's and half q's. Hence $|L_h| = q^N$, so the Gaussian heuristic suggests that

$$\lambda_1(L_h) \approx \sqrt{\frac{2N}{2\pi e}}(q^N)^{\frac{1}{2N}} = \sqrt{\frac{qN}{\pi e}}\,.$$

However, by construction the NTRU lattice contains a short vector (f, g) of length $2d$. Typically $d \approx \frac{1}{3}N$ and $q \approx \frac{1}{2}N$, so in a typical NTRU lattice,

$$\frac{\text{Gaussian Heuristic}}{\text{Actual Shortest Vector}} \approx \frac{\sqrt{\frac{qN}{\pi e}}}{\sqrt{2d}} \approx \frac{1}{5}\sqrt{\dim(L_h)}\,.$$

Conclusion : The private key vectors in an NTRU lattice are $O(\sqrt{dim})$ shorter than the other vectors. In particular, solving SVP (or CVP) breaks NTRU.

The Gaussian Heuristic and Knapsack Lattices

The lattice L used to analyse knapsack cryptosystems has dimension $n+1$ and its basis is an upper triangular matrix with 1's on the diagonal except for one entry

$$t = x_1 a_1 + \cdots + x_n a_n \,.$$

The $x_i \in \{0,1\}$ are small, but the a_i satisfy $a_i \approx 2^{2n}$.
Thus $|L| = t \approx \frac{1}{2} n 2^{2n}$. But L contains the vector $v = (x_1, ..., x_n, 0)$ of length $||v|| \approx \sqrt{\frac{n}{2}}$. Hence for large n

$$\frac{\text{Gaussian Heuristic}}{\text{Actual Shortest Vector}} \approx \frac{4}{\sqrt{\pi e}} \approx 1.37 \,.$$

Thus the shortest vector in L is very likely to be the plaintext vector $(x_1, ..., x_n, 0)$, so solving SVP breaks the knapsack cryptosystem.

Attack of the Knapsack problem (after Lagarrias/Odlysko)

Denote by ρ_i the rows of the lattice $L_{a,t}$.

- They compute a reduced basis $c_1, \ldots c_n$ by using LLL.
- They look for at least one vector c_j of which the coordinates $c_{i,j}$ are either 0 either a fixed number λ.
- Take $x_i = \lambda^{-1} c_{i,j}$ and check whether $t = \sum a_i x_i$. If this is true, we succeed and stop !
- Otherwise we do again the previous steps replacing t by "the complement" $t' = \sum a_i - t$.

Comments on the Lagarrias/Odlysko method (1/2)

This algorithm performs therefore one or two LLL reductions ; for the second one the plaintext is replaced by the one with each x_i replaced by "its complement".

Let e_i be a solution of the equation $t = \sum a_i e_i$, with $e_i = 0$ or 1 and $e = (e_1, ..., e_n, 0)$. Remember that it is a vector in the lattice as

$$e = \sum e_i \rho_i + \rho_{n+1} .$$

Comments on the Lagarrias/Odlysko method (2/2)

Let $\Lambda(B)$ the set of the lattices $L_{a,t}$, with the bound $a_i < B$.

The number of such lattices is B^n. Lagarrias and Odlysko proved the following result :

Theorem (Lagarrias/Odlysko) : Let e be a vector with coordinates 0 or 1 with $\sum e_i \leqslant \frac{n}{2}$.

Suppose that $B = 2^{\beta n}$, with $\beta > 1,54725$.

Then then number of lattices of the previous kind with e non zero vector of minimal length and the $a_i < B$, is
$B^n - O(B^{n-c_\beta} \log(B)^2)$, with $c_\beta = 1 - 1,54725/\beta > 0$.

Therefore almost all the lattices we consider contains e as small vector. The density of the lattices is β^{-1}. The theorem works as soon as the density is $< (1,54725)^{-1} = 0,645$.

Motivation for working with polynomial systems of equations

Multivariate polynomials are just polynomials in several indeterminates (or variables if you prefer).
We are interested in polynomials over finite fields. In fact, we easily get polynomials...
Indeed, using for instance Lagrange interpolation, every function over $\mathbb{F}_{q^n}$ can be written as a polynomial

$$f(x) = \sum_{0 \leqslant i < q^n} a_i x^i \, .$$

But $\mathbb{F}_{q^n}$ can be seen as an extension of degree n over $\mathbb{F}_q$ (or more exactly here, as n-dimensional vector space over $\mathbb{F}_q$). Thus, f can be in fact written as a set $(f_1, \ldots, f_n)$ of multivariate polynomials over $\mathbb{F}_q$ where

$$f_i(x_1, \ldots, x_n) = \sum_{\alpha \in \mathbb{N}^n} a_\alpha \prod_{1 \leqslant i \leqslant n} x_i^{\alpha_i} \, .$$

Cryptosystems based on multivariate polynomials...

Quadratic Multivariate Polynomials and MQ

☞ A generic multivariate polynomial of degree d in n unknowns defined over $\mathbb{F}_q$ is handled with complexity $O\left(\binom{n+d}{d}\right)$.

Thus, we should use quadratic polynomials.

The following system of quadratic equations, for $k = 1, \ldots, m$ is called *Multivariate Quadratic Systems* (MQ)

$$\sum_{1 \leqslant i \leqslant j \leqslant n} \alpha_{i,j}^{(k)} x_i x_j + \sum_{1 \leqslant i \leqslant n} \beta_i^{(k)} x_i + \delta^{(k)} = y_k \, .$$

☞ Solving the MQ system is a problem NP-complete (can be proved by reduction to 3-SAT). The MQ problem was considered hard on the average. Nevertheless, easy for $m = 1$ and $m = O(n^2)$.

Idea : use MQ type problem for building asymmetric cryptosystems.

The cryptosystem HFE

HFE stands for Hidden Field Equation, it is an asymmetric cryptosystem designed by Jacques Patarin in 1996. We will describe a basic version of HFE.

Let

$$f(x) = \sum_{i,j} \beta_{i,j} x^{2^{\theta_{i,j}} + 2^{\varphi_{i,j}}} + \sum_{k} \alpha_k x^{2^{\varepsilon_k}} + \mu$$

be a polynomial in x over $\mathbb{F}_{2^n}$ of degree d, for positive integers $\theta_{i,j}, \varphi_{i,j}, \varepsilon_k$.

As seen before, we can represent f as a family of n polynomials of n variables $x_1, \ldots, x_n$ over $\mathbb{F}_2$:

$$f = (g_1, \ldots, g_n),$$

with $g_i \in \mathbb{F}_2[x_1, \ldots, x_n]$.

The cryptosystem HFE (continued)

Due to the choice of f and using the fact that the Frobenius morphism $x \mapsto x^2$ is a $\mathbb{F}_2$-linear endomorphism of $\mathbb{F}_{2^n}$, the g_i are quadratic polynomials (i.e., the total degree of g_i is 2).
Let S and T be two $n \times n$ non-singular matrices over $\mathbb{F}_2$ then we can compose S, f and T :

$$S(f(Tx)) = (h_1(x_1, \ldots, x_n), \ldots, h_n(x_1, \ldots, x_n)),$$

with $x = (x_1, \ldots, x_n)$ and again $\mathbb{F}_{2^n}$ is regarded as an n-dimensional vector space over $\mathbb{F}_2$. Due to linearity, the h_i are again quadratic polynomials.

Description of HFE

Private key : the function f, two affine or linear bijections S and T as above.

Public key : A representation of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ (could be part of the domain parameters), polynomials h_i for $i = 1, \ldots, n$ as above, computed using the private key f, S and T.

Encryption : to encrypt the n-tuple $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ (representing the message), we compute the ciphertext

$$y = (h_1(x_1, \ldots, x_n), \ldots, h_n(x_1, \ldots, x_n)) .$$

Decryption : to decrypt the ciphertext y, first we find all the solutions z of the (univariate) equation $f(z) = T^{-1}y$ and then we compute $S^{-1}z$ (we need some extra data in order to recover the unique solution).

☞ Notice that in our settings, we will only (or mainly) consider *random quadratic* polynomial, meaning that $\beta_{i,j}, \alpha_k, \mu$ are random in $\mathbb{F}_{2^n}$.

Finding the roots of a polynomial in $\mathbb{F}_{2^n}$

Finding the roots of a polynomial of degree d with coefficients in $\mathbb{F}_{2^n}$ can be done in $O(\mathsf{M}(d)\log(d))$ operations in $\mathbb{F}_{2^n}$ where $\mathsf{M}(d)$ is the cost of polynomial multiplications.

In practice we can use software like PARI/GP (`polrootsff`), Singular or NTL (to name a few). Both are reasonably efficient (NTL is very fast for this). To get an idea, with NTL on a modern computer we can find one root of a polynomial of degree 129 over $\mathbb{F}_{2^{80}}$ in few milliseconds.

HFE : a toy example

Let consider $n = 3$ with $\mathbb{F}_8 = F[t]/(t^3 + t + 1)$. We define f as

$$f(x) = (t^2+t+1)x^6 + (t^2+t+1)x^5 + tx^3 + x^2 + (t^2+t+1)x + t\,.$$

For simplicity, we will choose S and T to be the identity.

HFE : a toy example

The public key can be described by the following three polynomials over $\mathbb{F}_2$

$$
\begin{aligned}
h_1(x_1, x_2, x_3) &= x_1x_2 + x_1x_3 + x_3 \\
h_2(x_1, x_2, x_3) &= x_1x_2 + x_2x_3 + x_2 + x_3 + 1 \\
h_3(x_1, x_2, x_3) &= x_1 + x_3 \, .
\end{aligned}
$$

HFE : a toy example

Suppose we want to encrypt the plaintext $p = (0, 1, 1)$ which correspond as an element of $\mathbb{F}_8$ to $t^2 + t$. We obtain as ciphertext $c = (c_1, c_2, c_3)$, given by the following evaluations :

$$\begin{aligned} c_1 &= h_1(p) = 1, \\ c_2 &= h_2(p) = 0, \\ c_3 &= h_3(p) = 1\,. \end{aligned}$$

The ciphertext correspond to the polynomial $t^2 + 1$

HFE : a toy example

Factorising $f(x) - (t^2 + 1)$ gives the following list of factors :

$$
\begin{aligned}
g_1 &= x^2 + (t^2 + 1)x + t^2 + t\,, \\
g_2 &= x^2 + (t^2 + t)x + t^2 + t\,, \\
g_3 &= x + t^2\,, \\
g_4 &= x + t^2 + t\,.
\end{aligned}
$$

As g_1 and g_2 are irreducibles, they cannot be solutions. So the plaintext is either t^2 or $t^2 + t$. It is where we need some extra data (hash values, redundancy such as parity bits,...), in order to distinguish the correct value. Notice that here the computations were simplified due to the fact we took S and T to be the identity.

Solving the system over $\mathbb{F}_2$

In order to break HFE, we have to solve the system

$$\begin{aligned} h_1 - c_1 &= 0\,, \\ &\vdots \\ h_n - c_n &= 0\,, \end{aligned}$$

and for this we can use Gröbner basis computations.

Solving the system over $\mathbb{F}_2$

The Gröbner basis of a family $(f_1, \ldots, f_m, x_1^2 - x_1, \ldots, x_n^2 - x_n)$ in $\mathbb{F}_2[x_1, \ldots, x_n]$ describe all the solutions (in $\mathbb{F}_2$) of the system $f_1 = 0; f_2 = 0; \cdots; f_m = 0$. In particular, there are no solutions if and only if the Gröbner basis is equal to 1, and there is exaclty one solution if and only if the Gröbner basis is equal to $(x_1 - a_1, \ldots, x_n - a_n)$ where $a_i \in \mathbb{F}_2$. Then $(a_1, \ldots, a_n)$ is the solution.

We add the field equations $x_i^2 = x_i$ (notice that here $+1 = -1$) in order to get the correct solutions.
It has be shown (Faugère), that the complexity of the computation is bounded by a polynomial in 2^n if the maximal degree of the polynomials occuring in the computations is less than n.

Solving the HFE challenge 1 ($d = 96$, $n = 80$)

The HFE challenge was solved by Faugére in 1999 using his F5 algorithm for Gröbner basis computations. It took more or less two days computations on the best computers of the period. Using today Magma version on high end computers, one can solve the HFE challenge in less than one hour, see

`http://magma.maths.usyd.edu.au/users/allan/gb/#challenge`

XL : Outline

The XL Algorithm is an algorithm for solving systems of multivariate polynomial equations (XL stands for eXtended Linearisations, or for multiplication and linearisation). As we will see, each independent equation obtained by relinearisation exists (in a different form) in XL, and thus XL can be seen as a simplified and improved version of relinearisation.

- Start with a system of multivariate quadratic equations

$$l_k = \sum a_{ijk} x_i x_j - b_k = 0$$

with variables $x_i \in \mathbb{F}$ ($\mathbb{F}$ a given finite field).

XL : Outline

- For I a sequence of elements $i_1, ..., i_k$ in $\{1, ..., n\}$, we take

$$x_I = \prod_{j=1}^{k} x_{i_j}$$

  and denote by $|I|$ its degree. (nb : $|I| = k$).
- One fixes an integer D large enough
- Let $\mathcal{I}_\mathcal{D}$ be the vectorial space generated by all $x_I l_k$ with degree $\leqslant D$.
- $\mathcal{I}=$ ideal generated by the l_k : it is $\cap_D \mathcal{I}_\mathcal{D}$
- the algorithm looks for equations in $\mathcal{I}_\mathcal{D}$ easier to solve.

XL : Outline

The XL Algorithm Execute the following steps :

1. Multiply : Generate all the products $x_I l_k$ with $|I| \leqslant D - 2$.
2. Linearise : Consider each monomial in x_I, $|I| \leqslant D$ as a new variable and perform Gaussian elimination on the equations obtained in 1.
   The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
3. Solve : Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm).
4. Repeat : Simplify the equations and repeat the process to find the values ofthe other variables.

XL : Outline

The XL algorithm is very simple, but it is not clear for which values of n and m it ends successfully, what is its asymptotic complexity, and what is its relationship to relinearisation and Gröbner basis techniques. Despite its simplicity XL may be succesfull for randomly generated over defined systems of multivariate equations.

HFE Challenge 1 (solved in Magma)

```
Magma V2.18-2     Fri Jan 27 2012 02:15:20 on palo-alto [Seed = 4254287995]
Type ? for help.  Type <Ctrl>-D to quit.
Time: 5299.120
Ideal of Polynomial ring of rank 80 over GF(2)
Order: Graded Reverse Lexicographical
Variables: x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9], x[10], x[11],
x[12], x[13], x[14], x[15], x[16], x[17], x[18], x[19], x[20], x[21], x[22],
x[23], x[24], x[25], x[26], x[27], x[28], x[29], x[30], x[31], x[32], x[33],
x[34], x[35], x[36], x[37], x[38], x[39], x[40], x[41], x[42], x[43], x[44],
x[45], x[46], x[47], x[48], x[49], x[50], x[51], x[52], x[53], x[54], x[55],
x[56], x[57], x[58], x[59], x[60], x[61], x[62], x[63], x[64], x[65], x[66],
x[67], x[68], x[69], x[70], x[71], x[72], x[73], x[74], x[75], x[76], x[77],
x[78], x[79], x[80]
Inhomogeneous, Dimension 0
```

```
Groebner basis:
[
   x[74]^2 + x[74],
    x[74]*x[77] + x[74],
    x[77]^2 + x[77],
    x[74]*x[79] + x[74],
    x[77]*x[79] + x[77],
    x[79]^2 + x[79],
    x[1],
    x[2] + 1,
    x[3] + 1,
    x[4] + x[74] + x[77] + 1,
    x[5] + x[79] + 1,
    x[6] + x[79],
    x[7] + x[74] + x[77] + x[79] + 1,
    x[8] + x[77] + 1,
    x[9] + x[74] + 1,
    x[10] + x[79] + 1,
    x[11] + x[74] + x[79] + 1,
    x[12] + x[74] + x[79] + 1,
    x[13] + 1,
    x[14] + x[74] + x[77],
    x[15] + x[77] + x[79] + 1,
    x[16] + x[74] + x[77],
    x[17] + x[77] + x[79] + 1,
    x[18] + x[74] + x[79],
    x[19] + x[77] + x[79],
    x[20] + x[77] + x[79] + 1,
    x[21] + x[77] + x[79],
    x[22] + x[77] + x[79] + 1,
    x[23] + x[74] + x[77] + x[79] + 1,
    x[24] + x[77] + x[79] + 1,
    x[25] + x[74] + x[77] + x[79],
    x[26] + x[74] + 1,
    x[27] + x[74] + x[77] + x[79] + 1,
    x[28] + x[79] + 1,
    x[29] + x[74] + x[79] + 1,
    x[30] + x[74],
    x[31] + x[74],
    x[32] + 1,
    x[33] + x[79] + 1,
    x[34] + x[77],
    x[35] + x[74] + x[77] + x[79],
    x[36] + x[77],
    x[37] + x[74] + x[77] + x[79] + 1,
    x[38] + x[74] + x[77] + x[79],
    x[39] + x[74] + 1,
    x[40] + x[74] + x[77] + x[79] + 1,
    x[41] + x[77] + 1,
    x[42] + x[77],
    x[43] + x[74] + 1,
    x[44],
    x[45] + x[74] + 1,
    x[46] + x[77] + x[79] + 1,
    x[47] + x[74],
    x[48] + x[77] + 1,
    x[49] + x[74] + x[77] + x[79],
    x[50] + 1,
    x[51] + x[74] + 1,
    x[52] + 1,
    x[53] + x[74] + x[77] + x[79],
    x[54] + x[74],
    x[55] + x[77],
    x[56] + x[79] + 1,
    x[57] + x[74],
    x[58],
    x[59] + x[79] + 1,
    x[60] + x[74] + x[77] + x[79] + 1,
    x[61] + x[74] + x[79],
    x[62] + x[79],
    x[63] + x[79],
    x[64] + x[77] + 1,
    x[65] + x[74] + x[77],
    x[66] + x[79],
    x[67] + x[74] + x[77],
    x[68] + x[79],
    x[69] + x[74] + x[77] + 1,
    x[70] + x[79] + 1,
    x[71] + x[77] + 1,
    x[72] + x[79],
    x[73] + x[74] + x[77] + x[79],
    x[75] + x[77] + x[79],
    x[75] + x[77] + x[79],
    x[76] + x[77] + x[79] + 1,
    x[78],
    x[80] + 1
]
```

```
[
    [ 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
    1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
    1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
    1, 1, 0, 1, 1 ],
    [ 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
    0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
    0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
    1, 1, 0, 1, 1 ],
    [ 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
    1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1,
    1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1,
    0, 0, 0, 1, 1 ],
    [ 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
    1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
    1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
    1, 0, 0, 0, 1 ]
]
Time: 0.000
Total time: 5310.840 seconds, Total memory usage: 14467.78MB
```