Secure data processing

To many the field of cryptography deals with sealing data...
- Plus a little bit of authentication and signature.

... But it can do much much more fancy things.

Folklore: a (micro-)processor processes only clear data.
- SW-solutions: e.g., virtualization (but it’s just software!).
  - Subject to software vulnerabilities exploitation.
- HW-assisted solutions: HW-enforced system partitioning in different fashions.
  - Subject to physical or cyberphysical attacks.

What if a processor could work directly on encrypted data without having to be disclosed any secret?
- This is where the magic of homomorphic encryption kicks in...
Outline of the talk

- Homomorphic encryption?
- Additive systems.
- Fully homomorphic encryption.
- Towards practical « cryptocomputing ».

A crypto primer in one slide

Informally a cryptosystem is defined as a set of two functions:

- encrypt(« cleartext », « some key material ») = « some unintelligible more or less provably random looking possibly much larger piece of garbage » aka the « ciphertext ».
- decrypt(« ciphertext », « some (possibly different) key material ») = the « cleartext ».

Public-key (aka asymmetric) crypto:

- The encryption (public) key is different from the (private) decryption key.
- Realm of relatively slow algorithms with, theoretically quite well understood security.

Symmetric crypto:

- A single (hence secret) key encrypts and decrypts.
- Realm of fast to superfast, carefully crafted algorithms with lesser well understood security.
Homomorphic encryption (informally)

- An homomorphic encryption system is a cryptosystem which, on top of allowing to encrypt and decrypt data, allows to perform some calculations (to some extent) in the encrypted domain.
  - Usually asymmetric, probabilistic schemes.
    - Deterministic homomorphic encryption schemes being intrinsically insecure.

- In essence, the « cryptocomputer »:
  - Keeps its algorithm private.
  - Can insert any (cleartext domain) data into the calculation.
    - E.g., using the public key.
  - Has access to neither intermediate nor final calculations results.
    - The « cryptocomputer » looses track of any data it injects in the calculation as soon as it interacts with encrypted-domain data.
    - The « cryptocomputer » cannot evaluate any condition depending on encrypted-domain data (more on that later on).

Trust model

- In the most basic settings, two parties are involved:
  - The user: owner of some private data.
  - The server: owner of an algorithm and possibly some data which it is willing to inject in the calculation.

- However, the server has complete control over the algorithm.
  - So the user must trust that the server will perform consistently with a functional specification – although it has no access to the algorithm details.
Let’s dream a little bit…

- Homomorphic encryption paves the way for a wide spectrum of new service settings where users can benefit from cloud services taking into account privacy-critical data, still without effectively giving them away.
  - Undisclosed cross-valorization of data (and algorithms).
  - Intrinsic data protection on vulnerable platforms.
  - Privacy-preserving outsourcing.
  - Etc.

- And, as an engineer, I lack imagination...
  - More real apps later on in the talk.

Homomorphic encryption (more formally)

- All known schemes are probabilistic (by necessity) and (generally) operate at the bit level.

- « API » specification:
  - \( enc_{pk} : \mathbb{Z}_2 \rightarrow \Omega \).
  - \( dec_{sk} : \Omega \rightarrow \mathbb{Z}_2 \).
  - \( add_{pk} : \Omega \times \Omega \rightarrow \Omega \).
  - \( mul_{pk} : \Omega \times \Omega \rightarrow \Omega \).

  where \( \Omega \) is a large cardinality set e.g. \( \mathbb{Z}_{q^n} \).

- Basic properties: for all \( m_1 \in \mathbb{Z}_2 \) and all \( m_2 \in \mathbb{Z}_2 \),
  - \( dec_{sk} (add_{pk} (enc_{pk}(m_1), enc_{pk}(m_2))) = m_1 \oplus m_2 \) (exclusive-or).
  - \( dec_{sk} (mul_{pk} (enc_{pk}(m_1), enc_{pk}(m_2))) = m_1 \odot m_2 \) (logical and).

- However, this is not enough as being able to decrypt after one operation does not imply being able to decrypt after an arbitrary number of operations.
  - All known FHE schemes are subject to a noise amplification phenomenon (managing that noise is why designing FHE schemes is difficult).
Formally

- What is really needed:
  - For all polynomial $p_{⊕,⊗} : Z_2^n \rightarrow Z_2$ and all $m_1 \in Z_2^n, \ldots, m_n \in Z_2^n$ we need,
    - $p_{⊕,⊗}(m_1, \ldots, m_n) = \text{dec}_a(p_{⊕,⊗}(m_1), \ldots, \text{enc}_{p_{⊕,⊗}}(m_n))$.

- As a consequence, any boolean circuit can be evaluated in the encrypted domain.
  - Or, equivalently, any program with a static control structure.
    - That is no if-then-else, no loop with crypto-domain variable-dependent termination, etc.
      - Although it might seem highly restrictive, it is not that bad (more on that later).

- So the question is whether or not there exists both secure and efficient fully homomorphic cryptosystems.
  - Security: by reduction to a reference problem and a concrete parameterization, $\theta(\lambda)$, relatively to the best known attacks on that problem.
  - Efficiency: polynomial or polylog overhead, $p(\lambda)$, per operation.
    - There is sometimes a huge difference between « theoretical » and « practical » efficiency.

Semantic security?

- Under a hardness assumption of a reference problem, a passive adversary can learn nothing about the plaintext value from the ciphertext in (expected) polynomial time.
  - Analogous to Shannon’s perfect secrecy for computationally constrained adversaries.

- Semantically secure schemes are necessarily probabilistic.
  - In particular, similar cleartexts (e.g., 0 and 1) must have many different ciphertexts (which often lead to non negligible expansion factors).

- Many such schemes are known, e.g.:
  - Paillier (1999).
    - Pick a random $r \in Z_n, c=g^r \mod n^2$.
    - $g$ and $n$ being the public key.
  - Etc.

- Homomorphic properties do not prevent semantic security.

"I can’t find an efficient algorithm, but neither can all these famous people."

"I can’t find a solution, but neither can all these famous people."
Any idea which of these ciphertexts are encryptions of 0 and which are encryptions of 1?

<table>
<thead>
<tr>
<th>Encryptions of 1:</th>
<th>Encryptions of 0:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2924608136398721289</td>
<td>2630732326863393797</td>
</tr>
<tr>
<td>1474366196710302793</td>
<td>1740934252228146561</td>
</tr>
<tr>
<td>2567207967840829840</td>
<td>325345389878009424</td>
</tr>
<tr>
<td>2151667377681512486</td>
<td>2174818941702764676</td>
</tr>
<tr>
<td>2593610843691957567</td>
<td>1139566886042106641</td>
</tr>
<tr>
<td>1987943116330592230</td>
<td>821239897411894783</td>
</tr>
<tr>
<td>2664828730912251679</td>
<td>1367467991441746522</td>
</tr>
<tr>
<td>2948529452920457725</td>
<td>2956811213964447041</td>
</tr>
<tr>
<td>1618408717616192430</td>
<td>846307085896778735</td>
</tr>
<tr>
<td>1034410661573368898</td>
<td>3104530909158659299</td>
</tr>
</tbody>
</table>

Any idea which of these ciphertexts are encryptions of 0 and which are encryptions of 1?
Any idea which of these ciphertexts are encryptions of 0 and which are encryptions of 1?

- 2653519422972370784
- 413448423265763204
- 1291313543354005947
- 2305675383275869842
- 2669623261641867485
- 69870838835387379
- 1155688282103802917
- 1937042751156905495
- 1473855045263783637
- 1196473292824158820
- 881049879282423562
- 1333527331613497707
- 1229345928423327835
- 688962940663792303
- 616039272910300078
- 129449037792119560
- 2173164225139652217
- 2828738679241450444
- 1990034337788215012
- 1144129954771463695

Answer:

- 2653519422972370784
- 413448423265763204
- 1291313543354005947
- 2305675383275869842
- 2669623261641867485
- 69870838835387379
- 1155688282103802917
- 1937042751156905495
- 1473855045263783637
- 1196473292824158820
- 881049879282423562
- 1333527331613497707
- 1229345928423327835
- 688962940663792303
- 616039272910300078
- 129449037792119560
- 2173164225139652217
- 2828738679241450444
- 1990034337788215012
- 1144129954771463695
The $\lambda$ parameter

- Reduction of the cryptanalysis of a system to the hardness assumption of a reference problem (e.g., factoring) means only that there is no other hidden weakness.
  - But it says nothing about the concrete dimensioning of the scheme parameters (e.g., the size of the modulus).

- Based on the panel of (hopefully exponential time) best known attacks on the reference problem, the cryptanalysts job is then to derive parameters such that any of these attacks requires an order of magnitude of at least $2^\lambda$ non trivial operations.

Concrete security levels:
- Less than 40 bits (trivial).
- 64 bits (low) –~ DES.
- 80 bits (quite strong).
- 128 bits (stronger) – AES-128.
- 256 bits (long term) – AES-256.

Back in the 20th century: Paillier’s system

- A public-key, semantically secure, cryptosystem with the following homomorphic properties:
  - $\text{dec}_{sk}(\text{enc}_{pk}(m_1) \cdot \text{enc}_{pk}(m_2) \mod n^2) = m_1 + m_2 \mod n$.
  - $\text{dec}_{sk}(\text{enc}_{pk}(m_1)^k \mod n^2) = k \cdot m_1 \mod n$.

- So we have the ability to add two ciphertexts and to multiply a ciphertext by any clear value (i.e., any value known by the cryptocomputer).
  - Of course, any additive cryptosystem can perform multiplications by a clear value in pseudopolynomial time.
    - $3x = x+x+x$ innit? 😊

- So what?
Underestimate the power of additive-only homomorphic cryptosystems you will not!

And, BTW, of multiplicative-only ones you will neither!

Applications of additive systems

- Basically, any linear operator i.e.,
  - Get the results of $Ax$ where $A$ is a cleartext m-by-n matrix and $x$ is an n-dimensional vector of cryptovariables.

- Not everything is linear but quite a few (interesting) things are:
  - FIR filtering.
  - Fourier, wavelets and various other transforms.
  - Linear system resolution (with a cleartext matrix).
  - Multilinear regression.
  - Counting (e.g., for data aggregation).
  - Etc.

- As a consequence, the field of signal processing over encrypted data did not wait for fully homomorphic systems to develop (e.g. Fontaine & Galand 2007).
**The magic of PIR**

- **PIR = Private Information Retrieval.**
  - I.e., retrieve a database record without revealing which one to the database owner.

- Doable with an additive-only system:
  - The querier sends a vector with an encryption of 1 in the $i^{th}$ position and $n-1$ encryptions of 0 in the other positions.
  - The DB owner homomorphically evaluates the following dot product:
    
    $$ r = \text{enc}(1) \times t[i] + \sum_{j \neq i} \text{enc}(0) \times t[j]. $$

  - Then the querier recovers $t[i]$ as $\text{dec}(r) = t[i]$.

- Combines good performances and wide app spectrum:
  - Private DB queries (obviously ☺).
  - Keywords search, packet inspection.
  - Private shortest path computation, knapsack resolution.

---

**A simple example of PIR magic**

- Suppose I want to retrieve a number in the list \{7,3,42\} without revealing which one.
- In fact I want the last one.
- So I send encryptions of 0, 0 and 1:
  1373399826202456717, 5985821055574142, 3125316372858344503.
- The server computes:
  - $3 \odot 1373399826202456717 = 755791940058174710.$
  - $7 \odot 59858210555741427653 = 1856930702731239845.$
  - $42 \odot 3125316372858344503 = 2753337671359622347.$
- Then sum the three values to get 146642186406800568, result which it sends to me.
- Then I decrypt(146642186406800568) and get 42!
The 2009 breakthrough


Analysis of the 2009 breakthrough

- In essence, Gentry’s first results is a first instance of a semantically secure fully homomorphic cryptosystem which computational overhead is polynomial in $\lambda$.

- Although « theoretical efficiency » (i.e., polynomial overhead) was achieved this first construction was mostly of theoretical interest.
  - A « heroic » implementation work (Gentry & Halevi, 2010) lead to a public key size of 2.3 GB, 2.2 hours of key generation and 30 minutes for performing one « recrypt » operation (which should be performed fairly often after each multiplication, although not systematically).

- Still, the pandora box was opened and progress has been extremely fast-paced since 2009.
A simple cryptosystem

The symmetric version of van Dijk et al. (2010) cryptosystem:

- **Key:**
  - An odd integer randomly chosen in \( p \in [2^{\eta-1}, 2^\eta] \).
- **Encryption of** \( m \in \{0, 1\} \):
  - Randomly choose a large \( q \) and \( r \) (\( 2r < q/2 \)) and let \( c = qp + 2r + m \).
- **Decryption:**
  - \( m := (c \mod p) \mod 2 \).

And that’s it!

- Although the public key version is slightly more complex.

Security of the scheme:

- Reference problem (approximate GCD):
  - Given a set \( x_1, \ldots, x_n \) of integers randomly chosen close to multiples of an integer \( p \), find \( p \).
  - Fact: under the hardness assumption of the above problem, the above scheme is semantically secure.

The noise problem

Homomorphic properties:

- \( m_1 + m_2 = D(E(m_1) + E(m_2)) \)
  - \( (((q_1 + q_2)p + 2(r_1 + r_2) + m_1 + m_2) \mod p) \mod 2 = m_1 \oplus m_2 \)
- \( m_1 m_2 = D(E(m_1) E(m_2)) \)
  - \( (((q_1 + 2r_1 + m_1)(q_2 + 2r_2 + m_2) \mod p) \mod 2 = ((q_1'p + 2r_1' + m_1m_2) \mod p) \mod 2 = m_1 \odot m_2 \).

*But...* These properties unfortunately do not hold forever.

- Indeed, if \( r_1 \) and \( r_2 \) respectively require \( N_1 \) and \( N_2 \) bits then \( r_1 + r_2 \) (respectively \( r' \)) requires \( \max(N_1, N_2) + 1 \) (respectively \( N_1 + N_2 \)) bits.
- As long as the noise, \( r \), following an operation is less than \( p \), decryption is possible otherwise we are doomed!
  - This is where the difficulties crop up: some noise reduction techniques are required to keep the noise below the decryption threshold and, so far, noise reduction techniques are algorithmically complex and computationally costly.
  - Although things are getting better...
Two main blueprints for FHE schemes

- **Bootstraping.**
  - Intuitively: use a cryptosystem homomorphic enough to (homomorphically) evaluate
    \[ c = c_1 \times (或 x) \times c_2 \text{ followed by } c' = \text{enc}_{pk}(\text{dec}_{sk}(c)) \]
    and hence denoise soon enough to remain below the decryption threshold.
  - Main issues: complex algorithmic techniques and high computational cost of the bootstraping operation.
    - Plus the circularity assumption: encrypting the secret key with the public key must be secure.
  - At present, bootstraping is still too computationally costly an operation.
    - Present state of the art still requires several minutes per bootstraping.

- **Leveled cryptosystems.**
  - Do not require any bootstraping.
  - Calculations are performed over a « ladder » of cryptosystems.
    - These systems are much more computationally efficient when it comes to noise management.
    - But they still have issues... (e.g., think IIR filtering).

Reminder: boolean circuits

- Definition: a directed graph \( G=(V,A) \) where vertices are either input, outputs or operator (XOR, AND) and which arcs corresponds to data transfers.

- Main characteristics:
  - Depth: number of arcs on a longest path from an input vertex to an output vertex.
  - Multiplicative depth: largest number of AND operators required to compute an output.
  - Topological order: \( f : V \rightarrow \{1,...,|V|\} \) such that for all arc \( (v,w) \), \( f(v) < f(w) \). Defines an order in which the operators can be evaluated.
  - Topological class: the set of operator at the same depth. These operators can be executed in parallel.

One of the first candidate with non prohibitive overhead (O(λL^3) per operation).

Principle of a leveled homomorphic cryptosystem.
- The calculation is spread over a ladder of cryptosystems (from large to low magnitude modulus).
  - Additions (XORs) are performed within the same level.
  - Multiplication (ANDs) require a level change.
- Hence, the critical parameter, is the multiplicative depth.

Computing with a leveled cryptosystems

Addition :
- Let c_1 and c_2, two « cryptobits » having respective levels n_1 and n_2.
- Step 1 : level those cryptobits to max(n_1, n_2).
  - Using key switching operators.
- Step 2 : addition operator.
  - A vector sum in BGV-12.

Multiplication :
- Let c_1 and c_2, two « cryptobits » having respective levels n_1 and n_2.
- Step 1 : level those cryptobits to l=max(n_1, n_2).
- Step 2 : multiplication operator.
  - A tensorial product in BGV-12 followed by a projection operator giving a ciphertext which can be decrypted only at level l+1.
  - Informally, the tensorial product « spreads » the noise and the projection operator (towards a smaller modulus) « decreases » its magnitude.
The best known candidates (so far)

  - And subsequent variants.

  - And subsequent variants (e.g. Fan-Vercauteren).

- **YASHE.**

Computing (for real) over encrypted data

- Given an homomorphic system, it is conceptually not difficult to define an integer type provided with operators doable hermetically in the encrypted domain:
  - Classical Optimized x-bits adders and multipliers.
  - Multiplication by a cleartext integer.
  - Negation and substraction (e.g., via 2-complementation).
  - Left shifting (by injection of encryptions of 0) and right shifting (by replication of the sign « cbit »).
  - Comparisions <, >, etc.

- And data dependent control?
  - Data dependent control needs to be regularized using e.g. a conditional assignment operator :
    - \( x = c ? a : b \) being equivalent to \( x = c \times a \oplus \neg c \times b \).
    - Note that the cryptocomputer can know (from time to time) whether or not a variable is boolean (e.g. after evaluating a boolean operator) but must not have access to its value.
Example: bubble sorting encrypted data

- Regularization of the inner if-then-else using the select operator.

- Static control structure hence systematic worst case complexity.
  - A price to pay for not leaking any information.

- Still, this is generic C++ code.

```cpp
template<typename integer>
void bsort(integer * const arr, const int n)
{
    assert(n>0);
    for(int i=0; i<n-1; i++)
    {
        for(int j=1; j<n-i; j++)
        {
            integer swap = arr[j-1] > arr[j];
            integer t = select(swap, arr[j-1], arr[j]);
            arr[j-1] = select(swap, arr[j], arr[j-1]);
            arr[j] = t;
        }
    }
}
```

Where `select(c, a, b) ≡ c ? a : b`.

Array dereferencing and assignment

- With cleartext indices:
  - Straightforward.

- With encrypted indices:
  - Dereferencing:
    \[
    t[i] = \sum_{j=1}^{n} \chi(i, j) \times t[j]
    \]
    with \( \chi(i, j) = 1 \) if \( i=j \), 0 otherwise.
  - I.e., it’s just an == operator.
  - Assignment \( t[i] := v \):
    \[
    t[j] := \chi(i, j) \times v \oplus (1 - \chi(i, j)) \times t[j], \forall j
    \]
    for \( j=1 \) to \( n \).
  - Hence array assignment and dereferencing are in \( O(n) \) (sic!).
  - But it’s the intuitive price for index privacy.
Proabilistic bit-level semantically secure encryption intrinsically induces non negligible expansion factors.

- Which can result in prohibitve communication or storage overhead.

Still, it is possible to work around that difficulty by, e.g.:
- Feed some AES-encrypted data in the homomorphic encryption box using the public key.
- Provide the cryptocomputer with the AES private key encrypted with the homomorphic cryptosystem.
- Homomorphically strip the AES encryption layer!

Still this is difficult to do this at reasonable computational cost using present state of the art systems.
- At least in principle, it means that any device (e.g. an embedded system with a HW AES accelerator) can easily be interfaced with an homomorphic cryptocomputing server.
- So we start to design FHE-friendly symmetric algorithms (in collaboration with specialized crypto teams).

Let’s see if I was clear enough...

If $\text{AES}^{-1}([x]_{\text{key}}, \text{key}) = x$

then

$\text{AES}^{-1}([x]_{\text{FHE}, \text{key}})_{\text{FHE}} = ?$
Let's see if I was clear enough...

If $\text{AES}^{-1}([x]_{\text{key}},\text{key}) = x$

then

$\text{AES}^{-1}([[x]_{\text{key}}]_{\text{FHE}},[\text{key}]_{\text{FHE}}) = [x]_{\text{FHE}}$

Full deployment scenario

Cloud private data

Cloud side

$\text{FHE}_{pk}$

$[\text{SYM}_{sk}]_{\text{FHE}}$

User side

$\text{FHE}_{sk}$

$\text{FHE}_{pk}$

$\text{SYM}_{sk}$

Cryptocomputing box

Crunching

Number

Homom. Decrypt with $[\text{SYM}_{sk}]_{\text{FHE}}$

Homom. Decrypt with $\text{FHE}_{sk}$

Reencrypt with $\text{FHE}_{sk}$

Decrypt (normally) with $\text{SYM}_{sk}$

(result)...

(Almost) only standard crypto here
Algorithms analysis

- Good news: for static control structure programs, static and dynamic analysis coincides.
  - Hence: running a program on a bit-level execution support on any input data is sufficient to characterize it.
  - This is needed for leveled systems as the multiplicative depth is required to determine the number of level.

- Hence, an instrumented cleartext runs becomes a compiler front-end for free.
  - Boolean circuit generation.
  - Then further post-optimizations.
  - Then (parallel) code generation.

CEA LIST’s technology approach

- Prototype of a compiler infrastructure for high-level cryptocomputing-ready programming, taking C++ code as input.

- Parallel code generation and « cryptoexecution » runtime environment.
  - OpenMP-based so far.

- Optimized prototypes of the most efficient FHE systems known so far.
A simple example

- A dummy Wikipedia-inspired medical diagnosis:
  - Blood composition threshold checking.

- Steps:
  - Algorithm implementation, testing and compilation is the clear.
  - Key generation and inputs encryption.
  - Program execution in the encrypted domain.
    - 128 bits of security with 812.
    - < 1 seconds RTD with server.
  - Outputs decryption.

- A simple yet realistic algorithm practically executed in the encrypted domain!

<table>
<thead>
<tr>
<th>Cardiology diagnostic tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Name</td>
</tr>
<tr>
<td>Total Cholesterol</td>
</tr>
<tr>
<td>LDL-C</td>
</tr>
<tr>
<td>HDL-C</td>
</tr>
<tr>
<td>Triglycerides</td>
</tr>
<tr>
<td>Blood Pressure</td>
</tr>
<tr>
<td>C-reactive protein</td>
</tr>
<tr>
<td>Hemoglobin</td>
</tr>
<tr>
<td>Cholesterol</td>
</tr>
<tr>
<td>Creatine</td>
</tr>
<tr>
<td>Creatine kinase</td>
</tr>
<tr>
<td>Uric acid</td>
</tr>
<tr>
<td>Calcium</td>
</tr>
<tr>
<td>Phosphorus</td>
</tr>
<tr>
<td>Magnesium</td>
</tr>
<tr>
<td>Potassium</td>
</tr>
<tr>
<td>Sodium</td>
</tr>
<tr>
<td>Chloride</td>
</tr>
<tr>
<td>Bicarbonate</td>
</tr>
<tr>
<td>Albumin</td>
</tr>
<tr>
<td>Total protein</td>
</tr>
<tr>
<td>Glucose</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
<tr>
<td>Lipid panel</td>
</tr>
<tr>
<td>Lipid ratio</td>
</tr>
<tr>
<td>Lipid profile</td>
</tr>
</tbody>
</table>
LES DÉBUTS DU CRYPTOLOGIE HOMOMORPHE...

QUEL BAROUF !
TU FAIS DUBI AVEC TON CRAY ?

UN TRÉ A BULLES !
The following people have contributed some of the work, thoughts or results presented in this talk (in alphabetical order):

- Carlos Aguilar (XLIM & LAAS).
- Sergiu Carpov (CEA LIST).
- Paul Dubrulle (CEA LIST).
- Simon Fau (CEA LIST).
- Caroline Fontaine (LABSTICC & ENSTB).
- Guy Gogniat (LABSTICC).
- Oana Stan (CEA LIST).

Further reading:

  - Along with the many references therein.