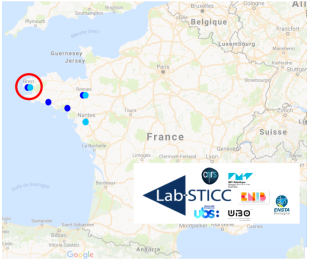
About Homomorphic Encryption Implementation Progresses and Challenges

Caroline Fontaine
Lab-STICC, CNRS & IMT-Atlantique
caroline.fontaine@imt-atlantique.fr



Outline

Outline

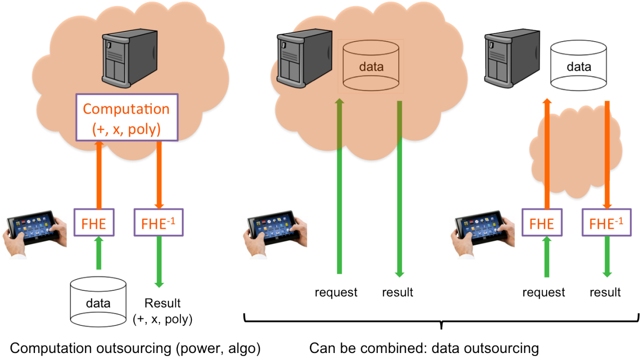
1. Context and Introduction
2. Applications and Practical Issues
   - Security
   - How to express high-level algorithms ?
   - Huge expansion of ciphertexts
   - Complexity
3. Conclusion

Fully Homomorphic Encryption

Usual encryption : SSL (Internet), Credit Cards, . . .

Fully Homomorphic Encryption [FHE] : Since 2009, we know how to evaluate polynomials (= Boolean circuits = programs) on encrypted data (since 1978 we only knew how to add OR to multiply, not both).

Homomorphic Encryption : we are dreaming of . . .



A revolution : data and/or services outsourcing without losing confidentiality !

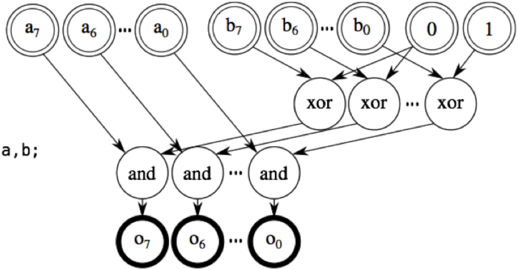
Impact : citizens, administrations, companies, military, . . .

Domains : health care, power plants, multimedia content delivery, . . .

Computations : comparing, sorting/filtering, clustering, compressing, . . .

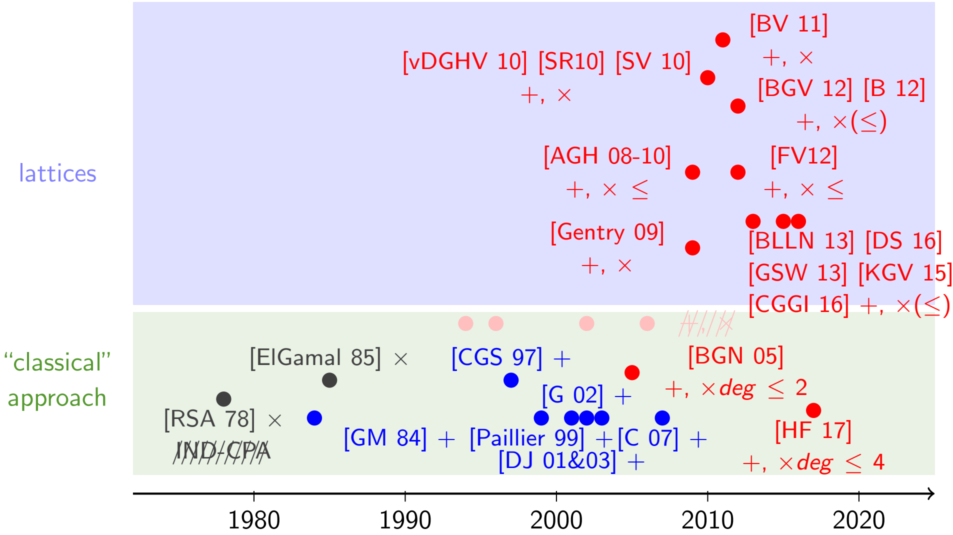
Program's output = Circuit Eval = Polynomial Eval

```
#include <iostream>
#include <stdint.h>
#include "integer.h"
void f
  (std::istream &i,
   std::ostream &o)
{
  SlicedInteger<int8_t> a,b;
  i >> a >> b;
  b = b ^ 0x01;
  a &= b;
  o << a;
}
```

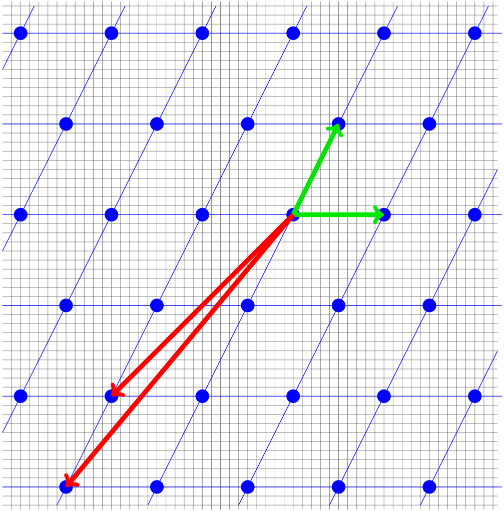


$$F_i(x) = x_i x_{i+8} \qquad i = 1. \ldots, 7$$
$$F_0(x) = x_8(x_{16} + 1)$$

It has been a long quest to handle polynomials



Lattice based S/FHE in a nutshell . . .



Lattice based S/FHE in a nutshell . . .

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0,1\}$: α, β random $\qquad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\qquad m = (c \bmod s) \bmod 2$

Lattice based S/FHE in a nutshell . . .

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0, 1\}$: α, β random $\qquad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\qquad m = (c \bmod s) \bmod 2$
- Homomorphic addition : $c + c' = m + m' + 2(\alpha + \alpha') + (\beta + \beta')s$

Lattice based S/FHE in a nutshell ...

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0,1\}$: α, β random $\qquad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\qquad m = (c \bmod s) \bmod 2$
- Homomorphic addition : $c + c' = m + m' + 2(\alpha + \alpha') + (\beta + \beta')s$

Condition :

To ensure a coherent decryption, we need : $m + m' + 2(\alpha + \alpha') < s$

Lattice based S/FHE in a nutshell . . .

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0,1\}$: α, β random $\quad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\quad m = (c \bmod s) \bmod 2$
- Homomorphic addition : $c + c' = m + m' + 2(\alpha + \alpha') + (\beta + \beta')s$

Condition :

To ensure a coherent decryption, we need : $m + m' + 2(\alpha + \alpha') < s$

If $2\alpha < s/2$, $2\alpha' < s/2$, and if c and c' are fresh ciphertexts, then it is ok.

Lattice based S/FHE in a nutshell . . .

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0,1\}$: α, β random $\qquad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\qquad m = (c \bmod s) \bmod 2$
- Homomorphic addition : $c + c' = m + m' + 2(\alpha + \alpha') + (\beta + \beta')s$

Condition :

To ensure a coherent decryption, we need : $m + m' + 2(\alpha + \alpha') < s$

If $2\alpha < s/2$, $2\alpha' < s/2$, and if c and c' are fresh ciphertexts, then it is ok.

If c_i is not a fresh ciphertext, we might not be able to decrypt it properly (too much noise) !

Lattice based S/FHE in a nutshell ...

Ex : FHE over the integers [vDGHV 10]

- Secret key (symmetric version here) : s
- Encryption of $m \in \{0,1\}$: α, β random $\quad c = m + 2\alpha + \beta s$
- Decryption : $c \bmod s = m + 2\alpha$ $\quad m = (c \bmod s) \bmod 2$
- Homomorphic addition : $c + c' = m + m' + 2(\alpha + \alpha') + (\beta + \beta')s$

Condition :

To ensure a coherent decryption, we need : $m + m' + 2(\alpha + \alpha') < s$

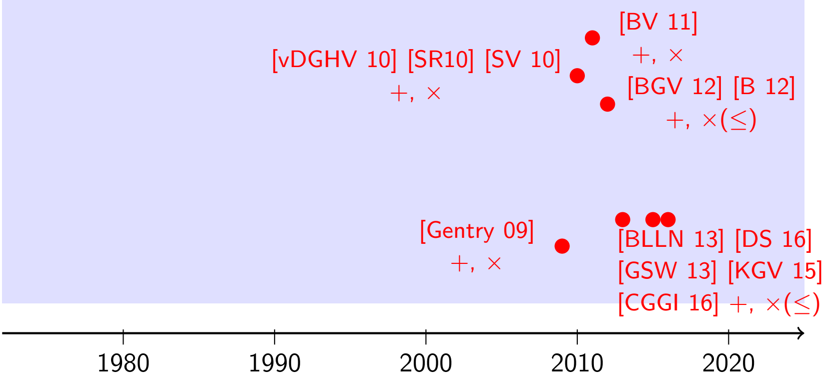
If $2\alpha < s/2$, $2\alpha' < s/2$, and if c and c' are fresh ciphertexts, then it is ok.

If c_i is not a fresh ciphertext, we might not be able to decrypt it properly (too much noise) !

And it is even worse in the case of homomorphic multiplication !

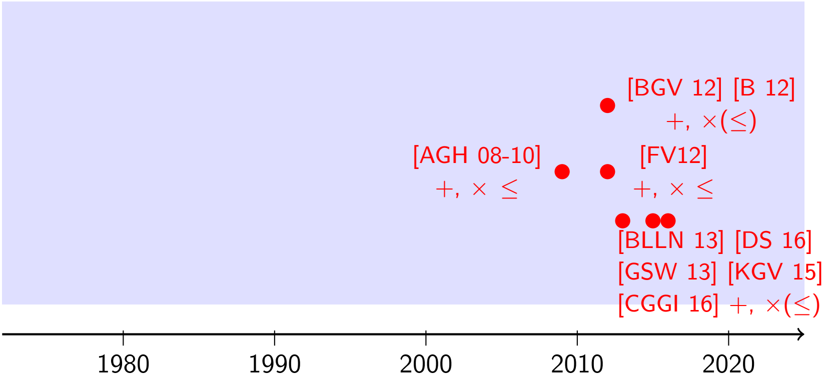
The challenge is to keep control of this noise during computation.

How to handle this noise ? (1/2)



- FHE : × unbounded → using bootstrapping
  - once the setting is fixed, "any" circuit can be evaluated
  - 2009-2014 : too complex to be used in practice
  - BUT recent improvements, *e.g.* [PV15] to optimize bootstrapping use, [CGGI16] to accelerate it

How to handle this noise ? (2/2)



- SHE schemes : × bounded → without bootstrapping
  - a limited (but often sufficient) number of multiplications
  - maximum mult. depth is related to the setting (cannot be modified afterwards)
  - a lower complexity

Outline

1. Context and Introduction

2. Applications and Practical Issues
   - Security
   - How to express high-level algorithms ?
   - Huge expansion of ciphertexts
   - Complexity

3. Conclusion

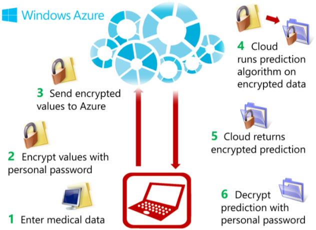
PoC : Outsourcing of (medical) diagnosis

Cardiology diagnostic tests

Test Name	Lower/normal risk	High risk	Cost $US (approx)
Total Cholesterol	<200 mg/dL	>240 mg/dL	
LDL-C	<100 mg/dL	>160 mg/dL	$150*
HDL-C	>60 mg/dL	<40 mg/dL	
Triglyceride	<150 mg/dL	>200 mg/dL	
Blood Pressure	<120/80 mmHg	>140/90 mmHg	
C-reactive protein	<1 mg/L	>3 mg/L	$20
Fibrinogen	<300 mg/dL	>460 mg/dL	$100
Homocysteine	<10 µmol/L	>14 µmol/L	$200
Fasting Insulin	<15 µIU/mL	>25 µIU/mL	$75
Ferritin	male 12–300 ng/mL female 12–150 ng/mL		$85
Lipoprotein(a) - Lp(a)	<14 mg/dL	>19 mg/dL	$75
Calcium Heart Scan	<100	>300	$250–600

(*) due to the high cost, LDL is usually calculated instead of being measured directly
source: Beyond Cholesterol, Julius Torelli MD, 2005 ISBN 0-312-34863-0

Simple threshold tests in cardiology diagnosis executed in the encrypted domain



Outsourced medical diagnosis

Some recent experimental results

Example of demonstration

- A dummy-yet-realistic « Wikipedia-inspired » medical diagnosys.
- Setup:
  - Algorithm implementation, compilation and deployment on a server.
  - Homomorphic precalculation of Kreyvium keystream on the server.
  - The Android tablet sends the Kreyvium-encrypted private user health data.
  - The server receives and homomorphically « transcrypts » to FHE.
  - The server homomorphically executes the diagnostic algorithm and sends back the encrypted answer to the tablet.
  - As the FHE secret key owner, the tablet is the only party able to decrypt and thus interpret the server reply.
- Characteristics:
  - Fan-Vercauteren sFHE.
  - Full-blown end-to-end 128 bits security.
  - 3.3 secs for program execution on the server.
  - < 4 secs RTD towards servers.
- Claim: practicality achieved for not-too-big-data realistic algorithms!

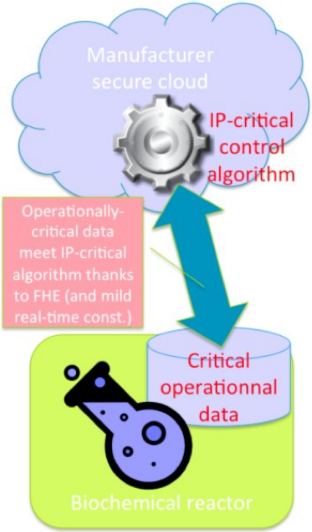
- Facteur de risque cardiovasculaire :

+1 si homme d'âge > 50 ans.
+1 si femme d'âge > 60 ans.
+1 si antécédents familiaux.
+1 si fumeur.
+1 si diabètes.
+1 si hypertension.
+1 si taux HDL < 40.
+1 si poids > taille-90.
+1 si activité physique/jour < 30.
+1 si homme consommant plus de 3 verres/jour.
+1 si femme consommant plus du 2 verres/jour.

"Intelligent" and "Evolving" algorithms

IP concerns and software update for the service provider :

- Targeted advertising
- Access Control with respect to user profile
- Biometric authentication
- Medical Diagnosis
- Cloud-based biochemical reactor control
- Machine Learning (deep learning)



Want to play ? (1/2)

2011 : open-source implementation of [SV10] by [PBS11] http://www.hcrypt.com

2012 : private implem. of [BGV12] dedicated to AES homo. eval. [GHS12]

2013-16 : private platform at CEA [AFFGS13,FSFAG13] , home-made implem. of [BGV12] (vect and poly) and [FV12] , + HElib and more recent open-source libraries

2013 : open-source implementation of [vDGHV10] with the improvements from [CNT12] : https://github.com/coron/fhe

2013 : private implementation in [CLT 13] dedicated to AES homomorphic evaluation using an improved version of [vDGHV10]

2013 : private implementation of [BLLN 13] , with good performances with 2 or 3 multiplicative depth

Want to play ? (2/2)

2013 : open-source implem. of [SV10] and [BGV12] called HElib by Halevi et al. http://shaih.github.io/HElib/

2014 : open-source implem. of [FV12] and [BLLN13] YASHE, compared in [LN14] https://github.com/tlepoint/homomorphic-simon

2015 : open-source library called SEAL1.0, based on YASHE' http://sealcrypto.codeplex.com/

2016 : open-source library to efficiently handle polynomials, called NFLlib https://github.com/quarkslab/NFLlib

2016 : open-source implementation of [FV12] based on NFLlib https://github.com/CryptoExperts/FV-NFLlib

2016 open-source multi-precision moduli library, called HElib-MP https://github.com/tricosset/HElib-MP, based on HElib

2016 : SEAL1.0 is replaced by SEAL2.1, based on another implementation of [FV12] http://sealcrypto.codeplex.com/

"soon" : FV with RNS from [BEHZ16] ; NFLlib based implementation of SHIELD [KGV15] ; library related with [CGGI16]

alse see common API http://bristolcrypto.blogspot.jp/2017/02/homomorphic-encryption-api-software.html (work in progress)

Outline

Which kind of security ?

Semantic Security

Semantic security is necessary !
(and as S/FHE schemes are malleable, IND-CCA2 can never be achievable).

$\Rightarrow$ probabilistic encryption

Which kind of security ?

Semantic Security

Semantic security is necessary !
(and as S/FHE schemes are malleable, IND-CCA2 can never be achievable).

⇒ probabilistic encryption
⇒ expansion (ciphertexts are longer than plaintexts)
and parameters setting has a huge impact on expansion !

Which kind of security ?

Semantic Security

Semantic security is necessary !
(and as S/FHE schemes are malleable, IND-CCA2 can never be achievable).

$\Rightarrow$ probabilistic encryption
$\Rightarrow$ expansion (ciphertexts are longer than plaintexts)
and parameters setting has a huge impact on expansion !

e.g. expansion is equal to (without batching) :

- equal to 2 with Paillier cryptosystem (only $+$)
- around $5,000$ with elliptic curve based solution [HF17] ($+, \times deg \leq 4$)
- between $500,000$ and $1,000,000$ for lattice-based S/FHE

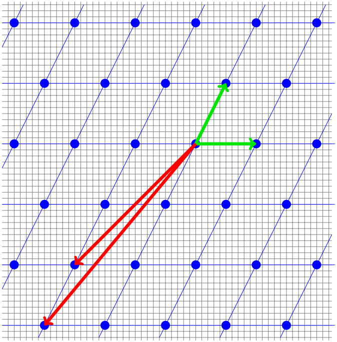
Which security level ?

Security Analysis of elliptic curve based schemes

Computational Security (w.r.t. DLP). Well understood and studied.

Security Analysis of lattice based schemes

Computational Security (w.r.t. hard problems as LWE, R-LWE,. . .)
Theoretical studies essentially focus on asymptotic and generic estimations (may be not so close to real S/FHE situations).
Some experiments (based on LLL, BKZ,. . .) provide estimations (but may remain too optimistic today).



See *e.g.* [Alb15,ABD16][Peik16][BF16][Alb17][AN17] .

Which security level for lattice based S/FHE ?

See the (online) estimator provided by Martin Albrecht (always evolving) :

SageMathCell

Type some Sage code below and press Evaluate.

```
load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
n, alpha, q = 256, 0.000976562500000000, 65537
set_verbose(1)
_ = estimate_lwe(n, alpha, q)
```

Evaluate   Language: Sage

Share

```
  mitm   bop: ≈2^917.3,  oracle:       370,    mem:  ≈2^904.3,    rop:  ≈2^913.3
   bkw   bop: ≈2^148.7,  oracle:  ≈2^131.5,      m:  ≈2^124.7,    mem:  ≈2^132.5,    rop:  ≈2^144.7,      b:         8,    t1:           8,     t2:        15,      l:
   sis sieve: ≈2^142.9,  oracle:   ≈2^28.6, delta_0: 1.0045951,   bkz2:  ≈2^224.3,   beta:       325,     lp:  ≈2^212.2,  quantum_sieve:  ≈2^134.1,      |v|: 1360.4505
   dec   rop: ≈2^122.8,  oracle:   ≈2^18.8,  sieve:  ≈2^121.7, delta_0: 1.0050017,   bkz2:  ≈2^183.4,   beta:       286,     lp:  ≈2^180.4,  quantum_sieve:  ≈2^113.9
kannan sieve: ≈2^114.4,  oracle:   ≈2^15.0, delta_0: 1.0051468,   bkz2:  ≈2^170.4,   beta:       274,     lp:  ≈2^169.6,  quantum_sieve:  ≈2^107.0,        m:       744
```

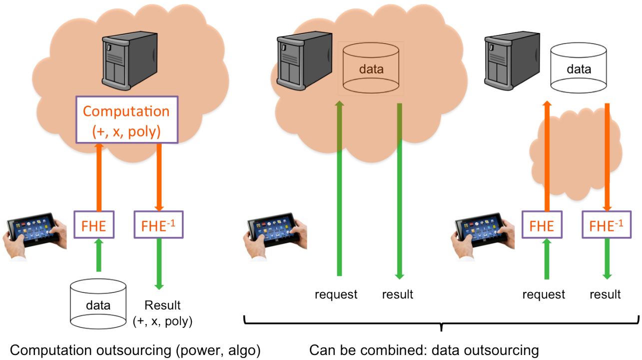
Help | Powered by SageMath

$\Rightarrow$ it is really hard today to know how to choose the right parameters to ensure a given security level (*e.g.* 128) and we really need more targeted attacks and studies to derive precise guidelines for the choice of parameters (see [MBF16] for a first attempt, based on the current state-of-the-art).

Outline

1. Context and Introduction
2. Applications and Practical Issues
   - Security
   - How to express high-level algorithms ?
   - Huge expansion of ciphertexts
   - Complexity
3. Conclusion

Applications : we are dreaming of . . .



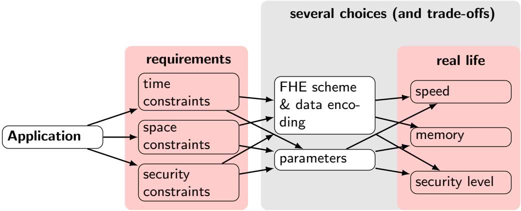
A revolution : data and/or services outsourcing without losing confidentiality !
Impact : citizens, administrations, companies, military, . . .
Domains : health care, power plants, multimedia content delivery, . . .
Computations : comparing, sorting/filtering, clustering, compressing, . . .

How to help programers ?

Our goal

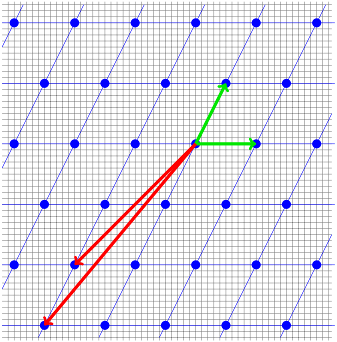
To help programers (not crypto specialists !) to use S/FHE in the development of their software/hardware stuff [AFF+13][FAR+13][CS14]

1. Cryptographers are necessary to help choosing the most appropriate S/FHE scheme & data encoding & parameters :



2. This being done, programers must be able to go further alone, without interacting with cryptographers !

With cryptographers : choosing data encoding (1/2)

Your (sliced) data $\longleftrightarrow$ 

Each piece of (sliced) data has to be related with one plaintext (a point of the lattice, *i.e.* integers or polynomials)

With cryptographers : choosing data encoding (2/2)

Your data : managing bits or integers ? (slicing)

Processing integers may seem more interesting at a first glance, BUT in some cases using integers will reduce the set of algorithms one can execute in the encrypted domain, *e.g.* `if-then-else` implies a management at the bit-level.

In case we choose an encoding at the bit-level, we need to redefine integers encoding to get operators on integers (based on those on bits, with 2's complement, sign bit, . . .), for :

`addition multiplication substraction` $\ll$ $\gg$

Batching (packing several plaintext into one)

To process several bits (resp. integers) at the same time, *e.g.* using Chinese Remaining Theorem.

Programers are not obliged to implem. S/FHE

From Armadillo platform [AFF+13][FAR+13][CS14] :

Definition of C++ classes `ClearBit` and `CryptoBit` written with the help of cryptographers (link with data encoding and S/FHE scheme) :

```
class C++ template<typename bit, int size>
```

Any programer can then use them :

Example

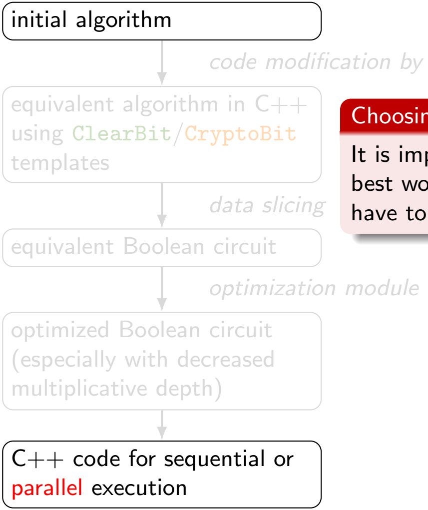
Applying a bubble sort on data in clear :

```
bsort<Integer<ClearBit,8> >(arr,n);
```

Applying the **same** bubble sort on encrypted data :

```
bsort<Integer<CryptoBit,8> >(arr,n);
```

Software Compilation Process and Optimization



Choosing the right algorithm

It is important to choose the algorithm with the best worst-case complexity (not usual!) if tests have to been performed over the encrypted data.

Software Compilation Process and Optimization

initial algorithm

↓ *code modification by the programer*

equivalent algorithm in C++ using ClearBit/CryptoBit templates

↓ *data slicing*

equivalent Boolean circuit

↓ *optimization module*

optimized Boolean circuit (especially with decreased multiplicative depth)

↓

C++ code for sequential or parallel execution

C++ classes ClearBit and CryptoBit

Applying a bubble sort on data in clear :

```
bsort<Integer<ClearBit,8> >(arr,n);
```

Applying the **same** bubble sort on encr. data :

```
bsort<Integer<CryptoBit,8> >(arr,n);
```

Software Compilation Process and Optimization

initial algorithm

↓ *code modification by the programer*

equivalent algorithm in C++ using `ClearBit`/`CryptoBit` templates

↓ *data slicing*

equivalent Boolean circuit

↓ *optimization*

optimized Boolean circuit (especially with decreased multiplicative depth)

↓

C++ code for sequential or parallel execution

Going down at the Boolean level

Data slicing & conversion Pgm → Boolean circuit.

Use XOR and AND for ClearBit
Use HE-ADD and HE-MULT for CryptoBit

Program → Boolean circuit

Comparisons of Encrypted Data

How to perform tests and express `if-then-else` ?

Boolean bitwise operators :
$$\begin{cases} \texttt{a < b} : \text{MSB of } \texttt{a+(-b)} \\ \texttt{a > b} : \text{MSB of } \texttt{b+(-a)} \\ \texttt{a = b} : \texttt{(a < b) NOR (a > b)} \end{cases}$$

"`if c then x = a else x = b`" can be achieved through the following operator : $x = \texttt{select}(c,a,b) = \begin{cases} a & \text{if } c=1 \\ b & \text{otherwise} \end{cases}$

$x = $ `select(c,a,b) = (c AND a) XOR ((NOT c) AND b)`

- no data leakage ;-)
- BUT bit-level encoding + worst-case complexity as we have to evaluate the whole circuit (all the branches of the circuit)

Bubble sort : a meaningful example

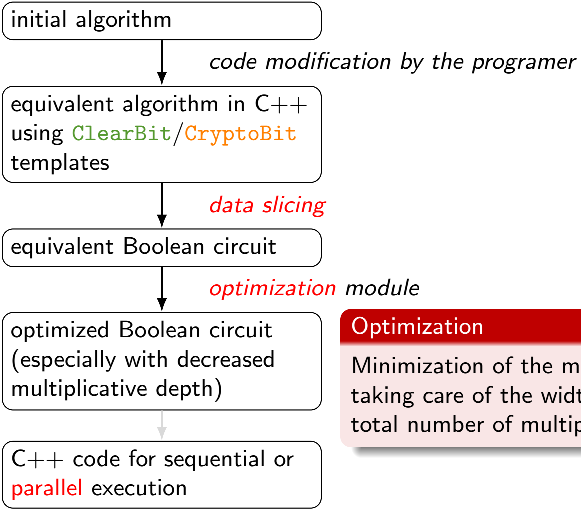
Classical bubble sort :

```
void bsort(int *arr,int n)
{
  for(int i=0;i<n-1;i++)
  {
    for(int j=1;j<n-i;j++)
      if(arr[j-1]>arr[j])
      {
        int t=arr[j-1];
        arr[j-1]=arr[j];
        arr[j]=t;
      }
  }
}
```

Rewritten bubble sort :

```
void bsort(int *arr,int n)
{
  for(int i=0;i<n-1;i++)
  {
    for(int j=1;j<n-i;j++)
      {
        int gt=arr[j-1]>arr[j];
        int t=gt*arr[j-1]^(!gt*arr[j]);
        arr[j-1]=gt*arr[j]^(!gt*arr[j-1]);
        arr[j]=t;
      }
  }
}
```

Software Compilation Process and Optimization



Optimization

Minimization of the multiplicative length (also taking care of the width of the circuit and the total number of multiplications and additions).

Optimizing the Boolean circuit

Characterization of # add, # mul, × depth

Estimation and optimization possible with the help of `ClearBit`.

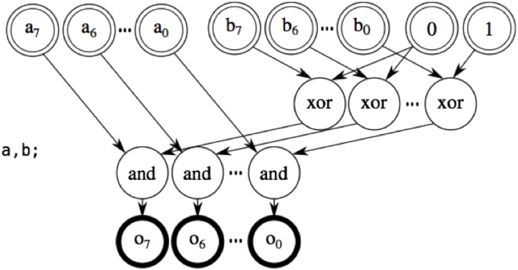
Some values for classical algorithms (before optimization) :

	$\sum_{i=1}^{10} t[i]$ (4 bits)	threshold (4 bits)	$b^2 - 4ac$ (4 bits)	bubble sort (10x4 bits)	FFT (256x32 bits)
# add	99	390	126	2372	7291592
# mul	27	60	32	238	5296128
× depth	4	5	7	69	166
	(16 bits)		(16 bits)	(10x8 bits)	
# add	423		1188	3240	
# mul	279		1126	2790	
× depth	16		32	136	

⇒ `ClearBit` class helps to debug the implementation and to optimize it !

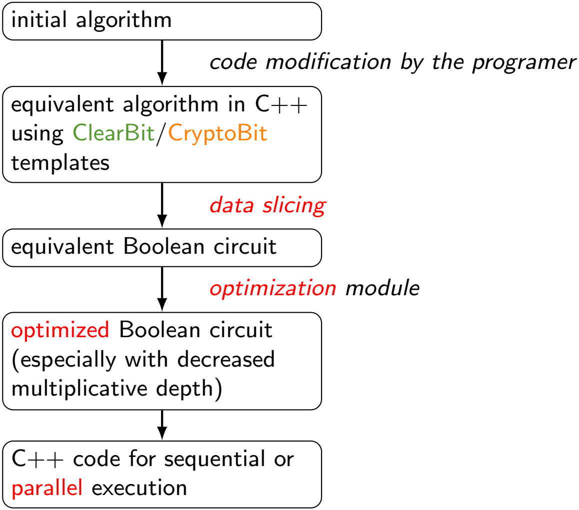
The trick : Program = Circuit = Polynomial

```
#include <iostream>
#include <stdint.h>
#include "integer.h"
void f
  (std::istream &i,
   std::ostream &o)
{
  SlicedInteger<int8_t> a,b;
  i >> a >> b;
  b = b ^ 0x01;
  a &= b;
  o << a;
}
```



$$F_i(x) = x_i x_{i+8} \qquad i = 1\ldots,7$$
$$F_0(x) = x_8(x_{16}+1)$$

Software Compilation Process and Optimization



Outline

1. Context and Introduction
2. Applications and Practical Issues
   - Security
   - How to express high-level algorithms ?
   - Huge expansion of ciphertexts
   - Complexity
3. Conclusion

An awful expansion factor !

Expansion (without batching)

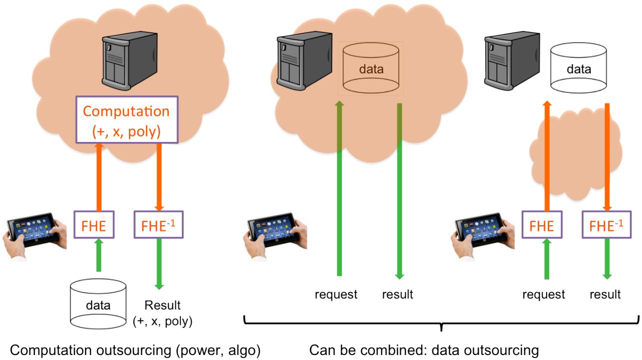
Current estimations of security parameters lead to an expansion factor

- equal to 2 with Paillier cryptosystem (only $+$)
- around $5,000$ with elliptic curve based solution [HF17] $(+, \times deg \leq 4)$
- between $500,000$ and $1,000,000$ for lattice-based S/FHE

$\Rightarrow$ pb to store and process, and to transmit data encrypted with S/FHE !

1. it would be very nice to design new schemes with a lower expansion,
2. we can help by choosing a good data representation and pack several plaintexts together (batching : CRT, SIMD, RNS),
3. we also have to do our best to manage huge ciphertexts, *e.g.* properly combining classical symmetric encryption with S/FHE.

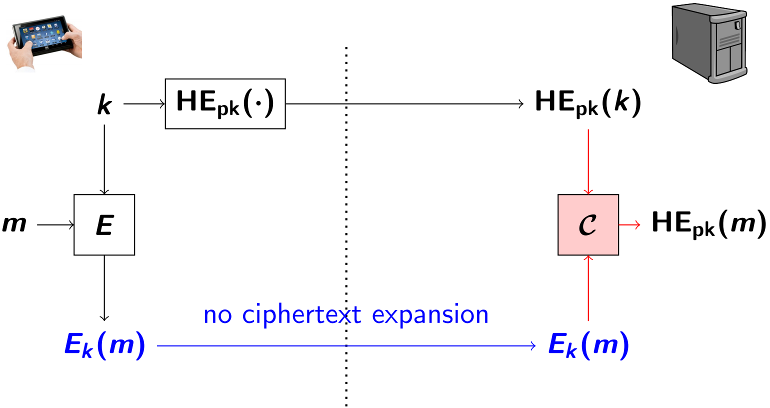
Applications : we are dreaming of . . .



A revolution : data and/or services outsourcing without losing confidentiality !
Impact : citizens, administrations, companies, military, . . .
Domains : health care, power plants, multimedia content delivery, . . .
Computations : comparing, sorting/filtering, clustering, compressing, . . .

How to efficiently upload S/FHE ciphertext ?

[NLV11]



$$\mathcal{C} = \mathbf{HE.Eval}_{E^{-1}}$$

What kind of symmetric encryption is the most appropriate ?

HE-friendly ciphers ? (1/2)

Main goal

To minimize the multiplicative depth of the decryption function.

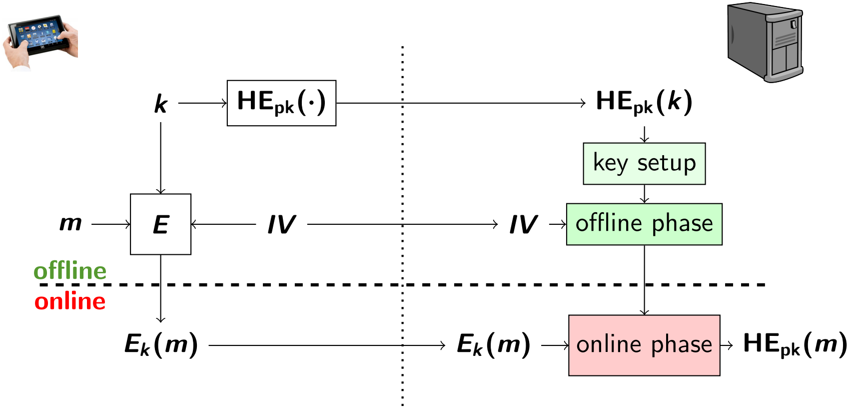
First concrete proposals have been block ciphers

- Already existing block ciphers :
  - Optimized implementations of AES [GHS12][CCKL+13][DHS14]
    $\rightarrow$ but AES's $\times$ depth remains too large ($\rightarrow$ too slow)
  - Lightweight block ciphers : SIMON [LN14] , PRINCE [DSES14]
    $\rightarrow$ SIMON behaves better than AES
    $\rightarrow$ PRINCE behaves better than SIMON, but remains too slow
- Dedicated block cipher : Low-MC-80 and Low-MC-128 [ARSTZ15]
  $\rightarrow$ but subject to some interpolation attacks (sparse ANF)
  $\Rightarrow$ a tweaked version has been presented at FSE 2016's rump session (more rounds), but security remains not clear (≤ 118)

Ciphertext decompression with IV-based encryption

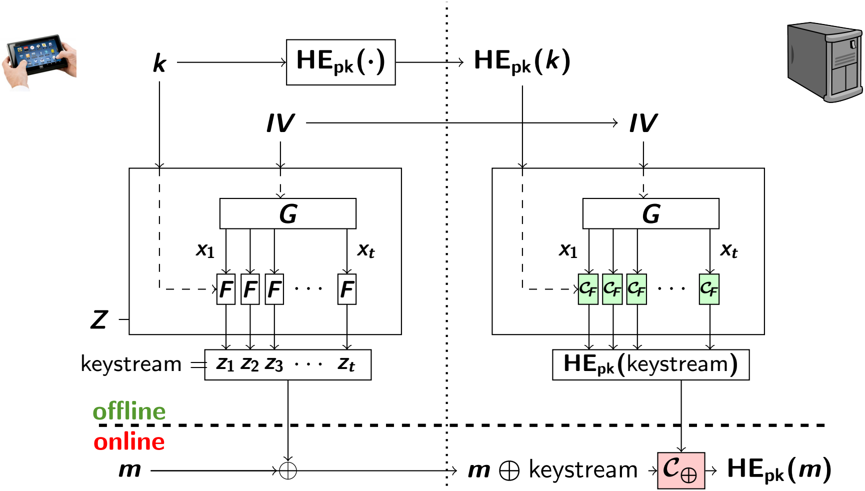
A new approach [CCF+16]

to reduce the online phase to a minimum . . .



Ciphertext decompression with IV-based encryption

. . . with an additive stream cipher ;-)



HE-friendly ciphers ? (2/2)

Using a stream cipher reduces on-line phase to the minimum.
Current candidates for function *F* are :

[CCCF+16] :

- Trivium : coming from eSTREAM (2008), firmly established security, 80 bits security
- Kreyvium : based on Trivium, same security confidence, 128 bits security

[MJSC 16] :

- Flip : lower complexity, but security should be more deeply analyzed [DLR 16]

According to today's state-of-the-art, Kreyvium seems to be the best available solution (but may be replaced by Flip if new security analysis is good).

Outline

Complexity issues

Complexity

High computation complexity related to the noise management.

Cryptographic issues :
⇒ it should be nice to have less complex S/FHE schemes, even if a huge effort has still been done and complexity already decreased a lot, and to optimize the use of bootstrapping, modulus switching, re-linearization, etc (*e.g.* see [PV15] for bootstrapping opt.).

Application related issues :
⇒ for a given target, we need to carefully choose the right algorithm (with the best worst-case complexity !)

⇒ we need to optimize the implementation (circuit optimization, bits/integers & batching, software/hardware implementation).

Some recent experimental results

FROM RESEARCH TO INDUSTRY
cea tech

Example of demonstration

- A dummy-yet-realistic « Wikipedia-inspired » medical diagnosys.
- Setup:
  - Algorithm implementation, compilation and deployment on a server.
  - Homomorphic precalculation of Kreyvium keystream on the server.
  - The Android tablet sends the Kreyvium-encrypted private user health data.
  - The server receives and homomorphically « transcrypts » to FHE.
  - The server homomorphically executes the diagnostic algorithm and sends back the encrypted answer to the tablet.
  - As the FHE secret key owner, the tablet is the only party able to decrypt and thus interpret the server reply.
- Characteristics:
  - Fan-Vercauteren sFHE.
  - Full-blown end-to-end 128 bits security.
  - 3.3 secs for program execution on the server.
  - <4 secs RTD towards servers.
- Claim: practicality achieved for not-too-big-data realistic algorithms!

- Facteur de risque cardiovasculaire :

+1 si homme d'âge > 50 ans.
+1 si femme d'âge > 60 ans.
+1 si antécédents familiaux.
+1 si fumeur.
+1 si diabètes.
+1 si hypertension.
+1 si taux HDL < 40.
+1 si poids > taille-90.
+1 si activité physique/jour < 30.
+1 si homme consommant plus de 3 verres/jour.
+1 si femme consommant plus du 2 verres/jour.

Outline

Conclusion

Nice

Very nice applications + post-quantum encryption :-)
A lot of efforts and progresses (everything is moving really fast).
Quite a lot of implementations available now.

Making small applications affordable ! We are on the right way :-)

BUT still a lot of (theoretical and practical) work to be done :

- security (to be better understood)
- expansion (to be better decreased and managed)
- complexity (worst-case complexity, bootstrapping optimization, etc)
- implementation optimization (Boolean circuit, software & hardware)
- help programers to choose the right scheme with an adapted setting (and do not forget "classical" crypto solutions)

How to choose the right solution and implementation ?

Several implementations have been publicly released,
BUT they are often tested separately :-(

There are very few attemps of comparisons based on public implementations : [LN14] + more recent experiments to be published in the next weeks/months (couldn't finish before this talk :-()

How to choose the right solution and implementation ?

Several implementations have been publicly released,
BUT they are often tested separately :-(

There are very few attemps of comparisons based on public implementations : [LN14] + more recent experiments to be published in the next weeks/months (couldn't finish before this talk :-()

Why is it difficult to FAIRLY compare schemes ?

- **Security :** it is difficult today to precisely link parameters setting with a given security level (hence difficult to be sure to compare the same security level for several schemes).
- **Expansion :** batching has been proposed for some schemes, not all.
- **Complexity :** we should compare implementations with the same optimization level.
- **Data encoding :** some schemes work on bits/integers/polynomials.

How to choose the right solution and implementation ?

Hence, choosing among lattice based schemes like BGV, FV, SHIELD, or even more classical schemes like BGN or BGN2 based on elliptic curves is not easy.

How to choose the right solution and implementation ?

Hence, choosing among lattice based schemes like BGV, FV, SHIELD, or even more classical schemes like BGN or BGN2 based on elliptic curves is not easy.

And even for a given scheme, implementations may use

- different lattice structures,
- different noise generation strategy,
- different optimization level,
- different batching techniques.

Ex : FV from SEAL 2.1 and from FV-NFLlib are very different !

$\Rightarrow$ Hence we should provide very precise benchmarks to be fair.

Questions ?

Thanks to all co-authors and collaborators (academic & industry)

French activities :

- design (S/FHE + friendly symmetric)
- security analysis
- batching
- compilation : software, hardware
- benchmarking and parameters setting

Announcement : IEEE WIFS in Rennes, Dec 4-7 2017