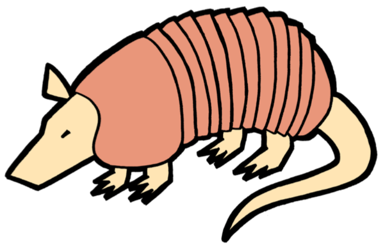
Cryptocomp

list ceatech

CINGULATA

An open-source toolchain for compiling and running programs over FHE

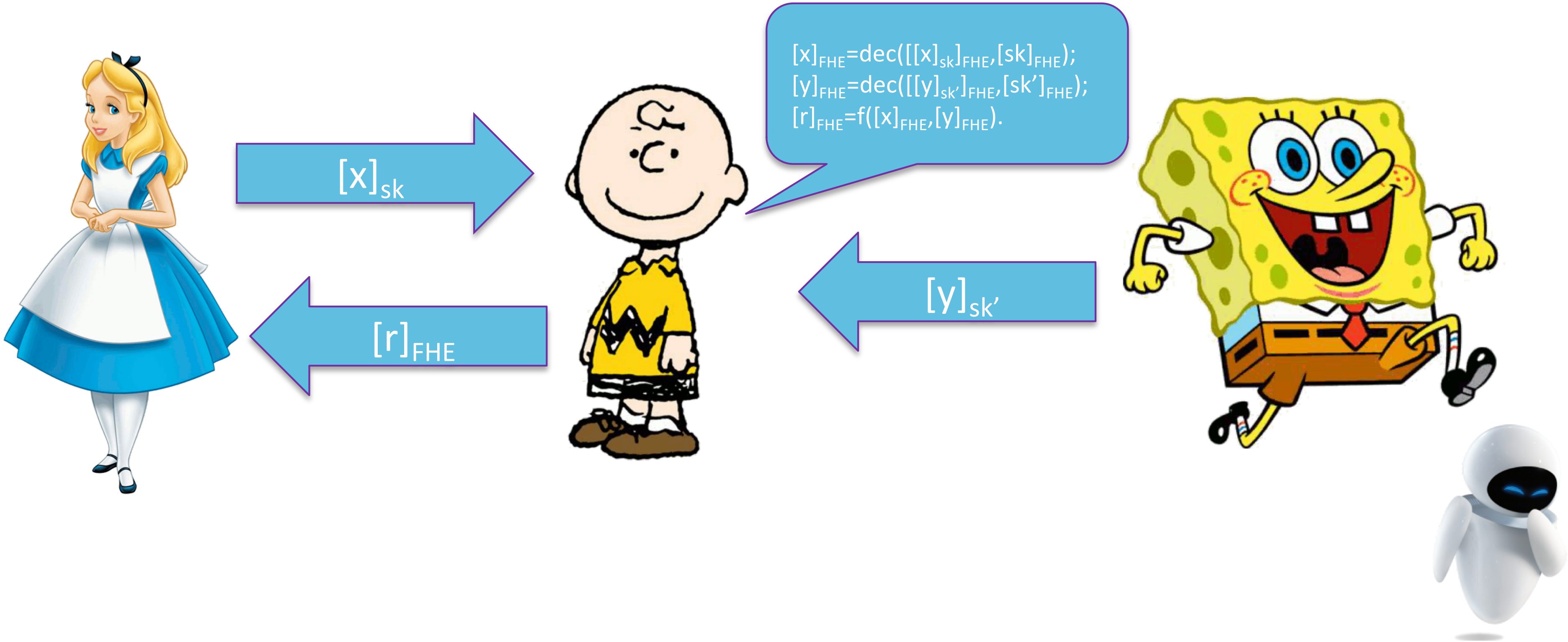


|Sergiu Carpov>/√2+|Renaud Sirdey>/√2
CEA LIST
(based on work with additional people)

May 2018

The FHE dream

- Can Charlie do something useful for Alice using both Alice and Bob data but without revealing them (the data) to him (Charlie) ?



FHE in a nutshell

- On top of allowing to encrypt and decrypt data, an FHE scheme allows to perform (any) calculations in the encrypted domain.
  - Without access to either intermediate or final calculations results by the computer.

- Although the first generation of systems were too costly, practicality has now been achieved for a first round of (lightweight-enough) applications.

Cryptosystem API:
- $enc_{pk} : Z_2 \rightarrow \Omega$.
- $dec_{sk} : \Omega \rightarrow Z_2$.
- $add_{pk} : \Omega \times \Omega \rightarrow \Omega$.
- $mul_{pk} : \Omega \times \Omega \rightarrow \Omega$.

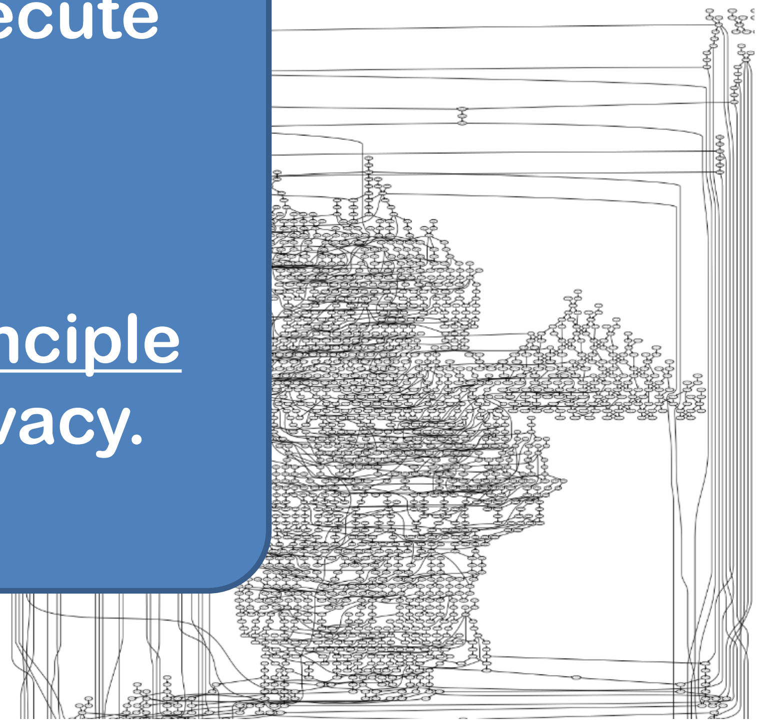
where Ω is a large cardinality set e.g. Z_qn.
Key properties: for all $m_1 \in Z_2$ and all $m_2 \in Z_2$
- $dec_{sk}(add_{pk}(enc_{pk}(m_1),enc_{pk}(m_2))=m_1 \oplus m_2)$.
- $dec_{sk}(mul_{pk}(enc_{pk}(m_1),enc_{pk}(m_2))=m_1 \otimes m_2$.

(and these properties hold long enough…)

The quest for universality

- So we can … circuits.
  - I.e. dire… which v… inputs, … (XOR, A…
- Boolean … control st… oblivious
- Oblivious … are Turi…
- Hence … everything computable!

- And, b. t. w., it is also possible to homomorphically execute an encrypted Turing machine.
- Hence, we can <u>in principle</u> ensure algorithm privacy.



FHE performances?

- Somewhat FHE:
  - Multiplications have an increasing cost with the mutliplicative depth.
  - Additions are « free ».
  - Large (depth-dep.) overheads.

depth	kb/bits	ms/AND
0	3	-
1	10	4
2	21	8
5	83	28
7	148	66
10	282	150
15	601	376
20	1039	680

- Bootstrapped FHE (TFHE):
  - Both multiplications and additions have the same cost, independently of the depth.
  - Smaller (depth-independent) overheads (8 ko/bit).

- HomNOT($\mathbf{c}$): instantaneous
- HomAND($\mathbf{c_1}, \mathbf{c_2}$): 13 ms
- HomXOR($\mathbf{c_1}, \mathbf{c_2}$): 13 ms
- HomMUX($\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$): 26 ms

The « strange » FHE computer

- No ifs (unless regularized by conditionnal assignment).
- No data dependant loop termination (need upper bounds and fixed-points).
- Array dereferencing/assignment in O(n) (vs O(1)).
- Algorithms always realize (at least) their worst-case complexity!
  - Complexity of dichotomic search?
- Can handle only a priori (multiplicative) bounded-depth programs (w/o bootstrapping).

Example: bubble sorting

- Regularization of the inner if-then-else using a cond. assignment operator.
- Static control structure hence systematic worst case complexity.
  - A price to pay for not leaking any information.

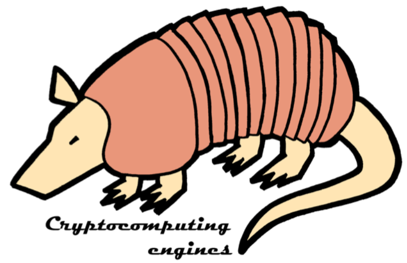
```
template<typename integer>
void bsort(integer * const arr,
  const int n)
{
 assert(n>0);

 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.
```

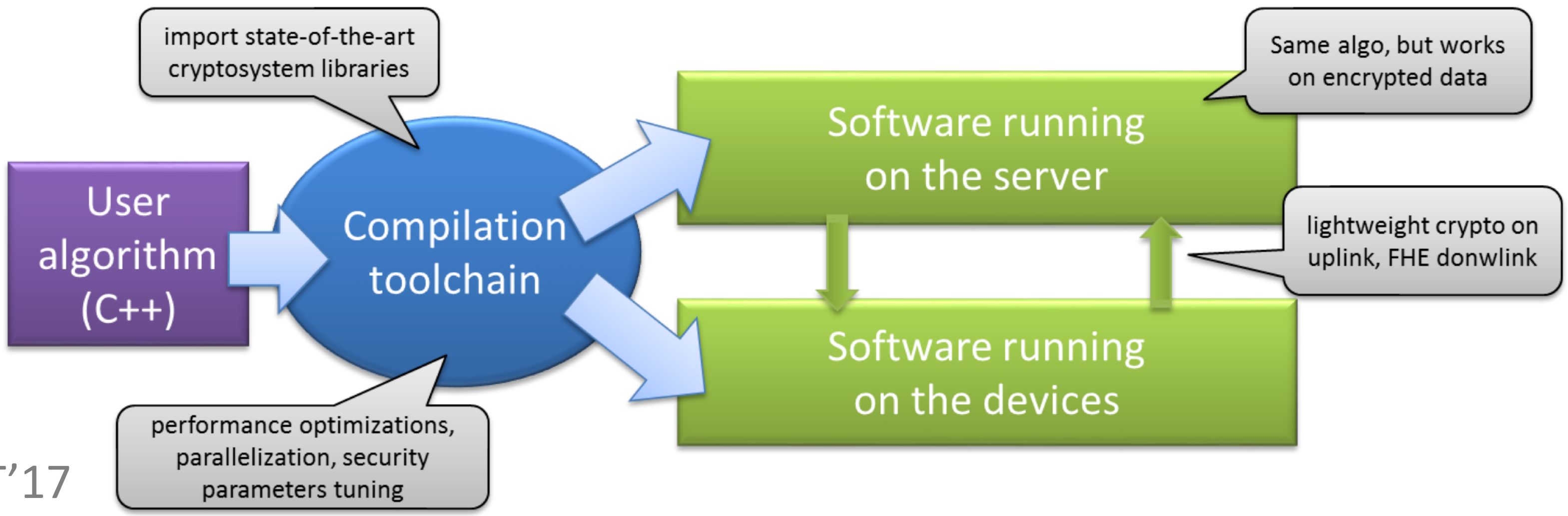
Array dereferencing and assignment

- With cleartext indices:
  - Straightforward.
- With encrypted indices:
  - Dereferencing: $$t[i] \equiv \sum_{j=1}^{n} \chi(i,j) \times t[j]$$
    with χ(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.

The CINGULATA compiler & RTE…

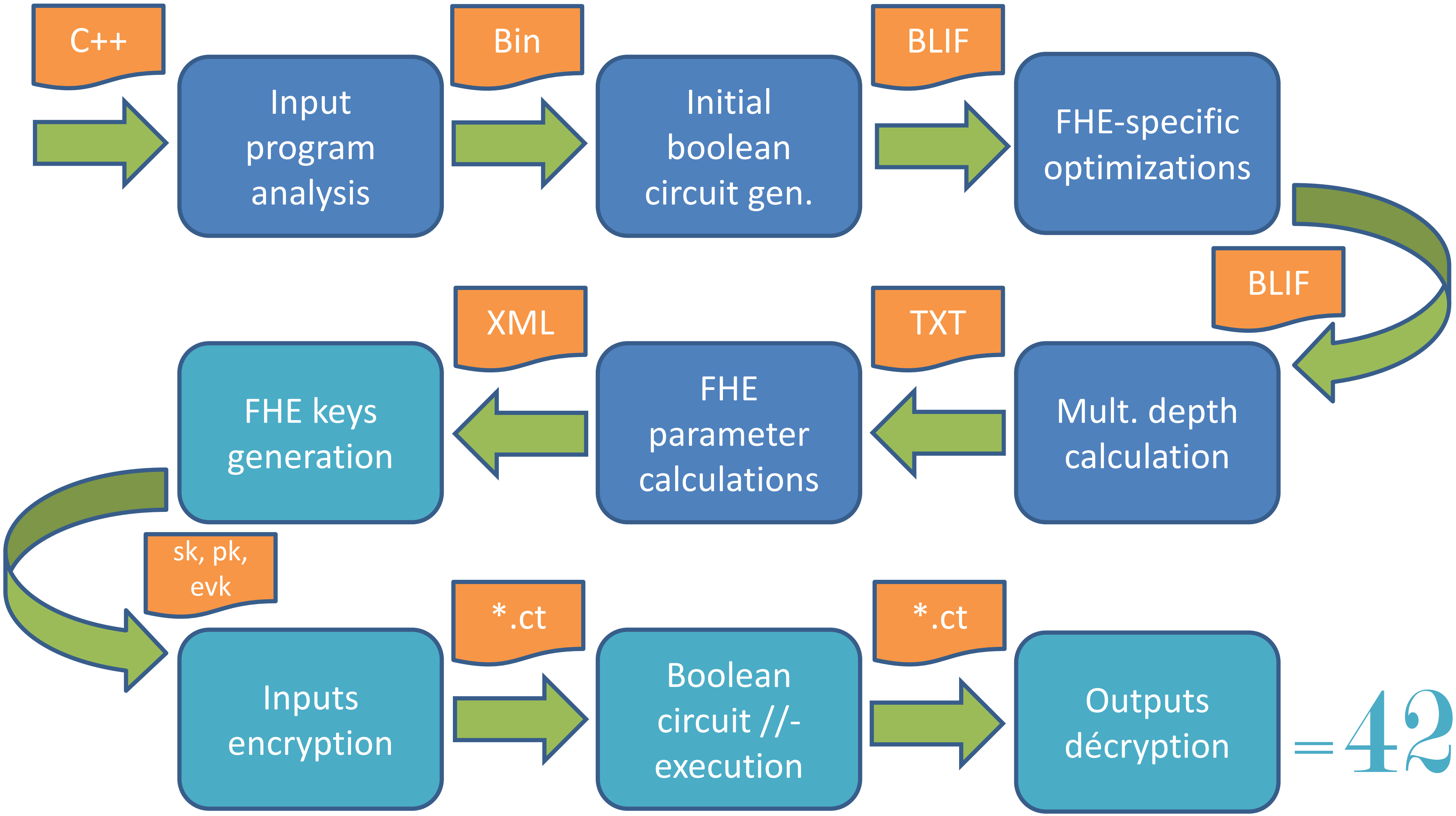
- A compiler infrastructure for high-level cryptocomputing-ready programming, taking C++ code as input.
- Boolean circuit optimization (ABC-based), parallel code generation and « cryptoexecution » (OpenMP-based) runtime environment.
- Optimized prototypes of the most efficient FHE systems known so far.
  - Also, with support of open source libs such as HELIB and (coming soon) TFHE.





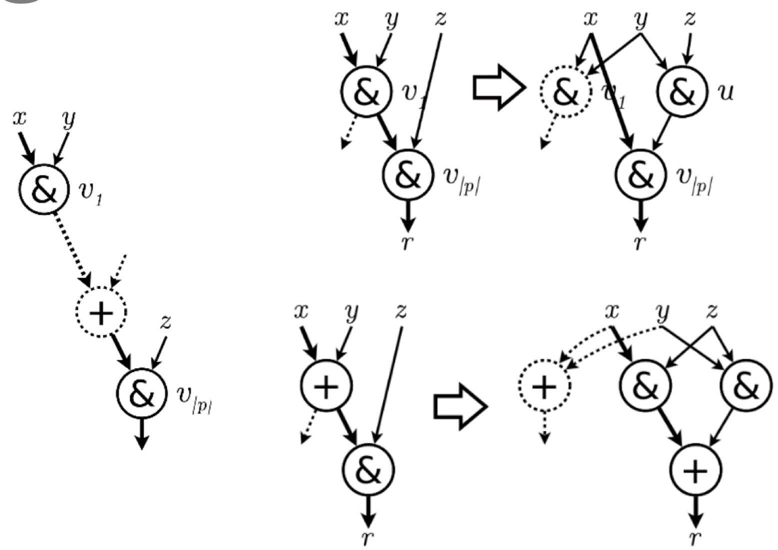
ASIACCS'15, PST'17

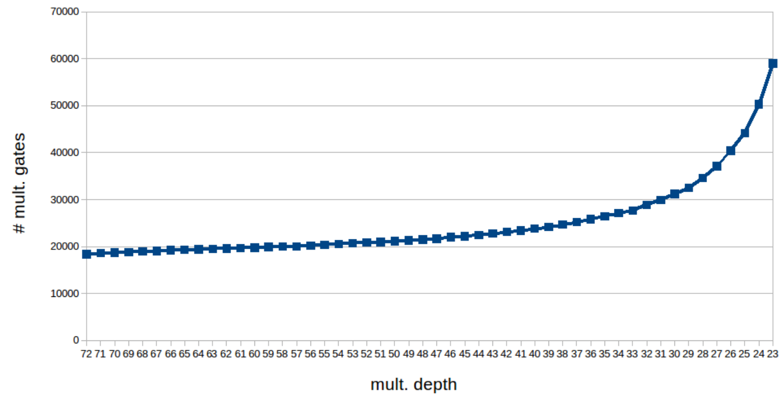
The CINGULATA compilation process



FHE-specific optimization modules

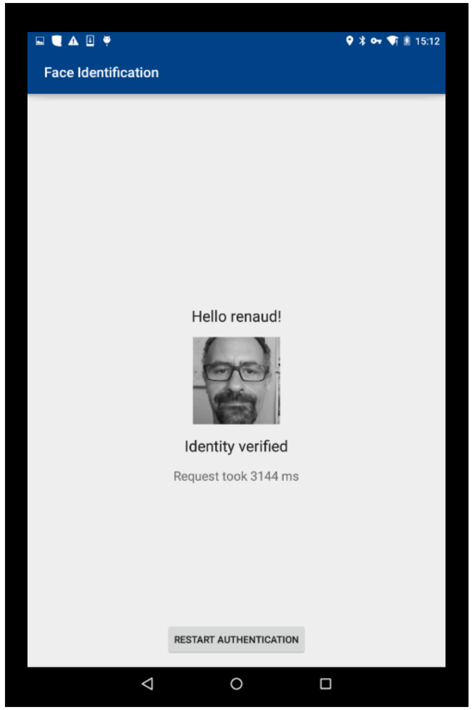
- Specifically targeted towards sFHE.
  - « multiplicative depth busting ».
  - As a secondary objective, multiplication count decreasing.
- Based on the iterative application of local circuit rewriting operators.
- Interesting results on some real-world algorithms.
  - E.g. multiplicative depth automatically downed from ~70 (prohibitively high) to ~20 (practically achievable) on RLE.

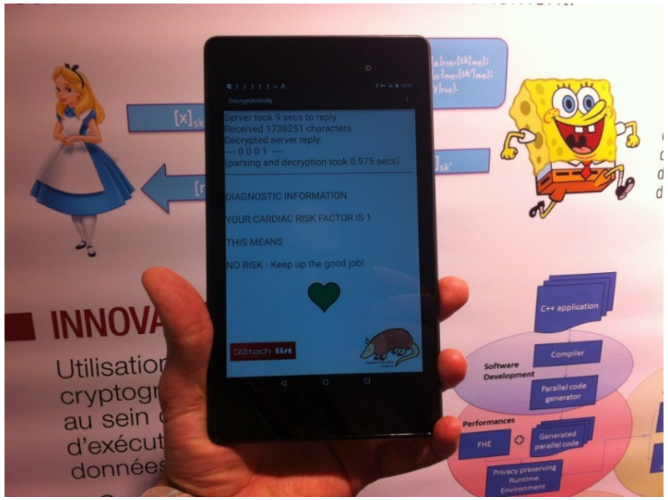




Practical achivements

- Medical diagnostic (various flavors, various complexities) – between 3 secs and 2 mins.
- Face authentication <4 secs RTD.
- Genome-based diagnostic < 10 mins.
- Energy-consumption profile classification < 1 secs.
- And a few others…





IEEE Cloud'16, IEEE CloudCom'16, ICISSP'17, SMARTGREENS'18.

IDASH 2017

- Secure genome analysis competition.
- Learn a logistic regression model over HE encrypted data.
- Batched gradient descent algorithm:
  - Straightforward implementation in Cingulata.
  - TFHE cryptosystem.
- 2nd place by AUC score.

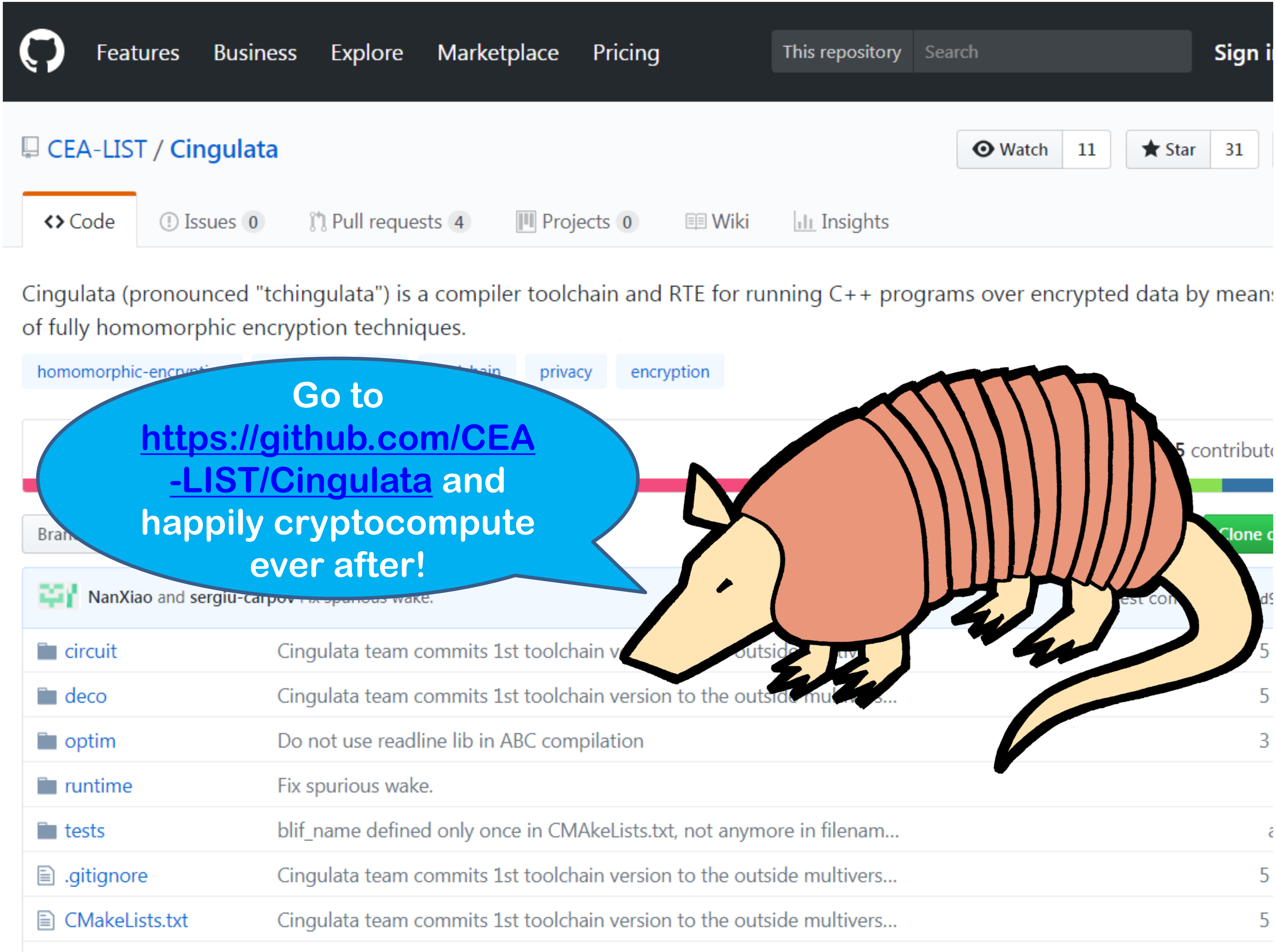
TRACK 3: BEST-PERFORMING TEAMS

Evaluated on (three datasets of 1422 records for training/ 158 records for testing + 18 features)

Teams	AUC	Encryption		Secure learning		Decryption		Overall time (mins)
		Size (MB)	Time (mins)	Time (mins)	Memory (MB)	Size (MB)	Time (mins)	
SNU	0.6934	537.667	**0.060**	**10.250**	2775.333	64.875	**0.050**	10.360
CEA LIST	0.6930	**53.000**	1.303	2206.057	**238.255**	**0.350**	**0.003**	2207.363
KU Leuven	0.6722	4904.000	4.304	155.695	7266.727	10.790	0.913	160.912
EPFL	0.6584	1011.750	1.633	15.089	1498.513	7.125	0.017	16.739
MSR	0.6574	1945.600	11.335	385.021	26299.344	76.000	0.033	396.390
Waseda*	0.7154	20.390	1.178	2.077	7635.600	20.390	2.077	5.332
Saarland	N/A	65536.000	1.633	48.356	29752.527	65536	7.355	57.344

History and next steps.

- First open-source release in Dec. 2017.
  - Developped mostly during CRYPTOCOMP.
- Next steps (non exhaustive):
  - More backends:
    - TFHE (gate-bootstraping flavor).
    - SEAL.
    - TFHE (LUT flavor).
  - More optimization modules.
  - Longer term : integration of VC techniques in the compilation process.


Features
Business
Explore
Marketplace
Pricing
This repository
Search
CEA-LIST / Cingulata
Watch 11
Star 31
Code
Issues 0
Pull requests 4
Projects 0
Wiki
Insights
Cingulata (pronounced "tchingulata") is a compiler toolchain and RTE for running C++ programs over encrypted data by means of fully homomorphic encryption techniques.
privacy
encryption
Go to https://github.com/CEA-LIST/Cingulata and happily cryptocompute ever after!
NanXiao and sergiu-carpov
circuit
Cingulata team commits 1st toolchain v
deco
Cingulata team commits 1st toolchain version to the outside mu
optim
Do not use readline lib in ABC compilation
runtime
Fix spurious wake.
tests
blif_name defined only once in CMAkeLists.txt, not anymore in filenam...
.gitignore
Cingulata team commits 1st toolchain version to the outside multivers...
CMakeLists.txt
Cingulata team commits 1st toolchain version to the outside multivers...

BACKUPS

The CINGULATA compilation process

1. Input program analysis.
2. Initial boolean circuit generation.
3. Boolean circuit FHE-specific optimisation.
4. Mult-depth calculation.
5. FHE parameters generation.
6. FHE keys generation.
7. Input encryption.
8. Boolean circuit execution.
9. Output decryption.