RSAConference2020
San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: **CRYP-R01**

Faster homomorphic encryption is not enough: improved heuristic for multiplicative depth minimization of Boolean circuits

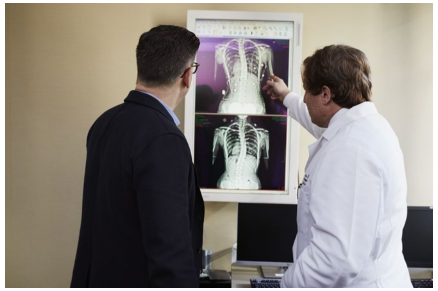
Pascal AUBRY

Research Engineer
CEA, LIST

Privacy protection

Services

Private data



Healthcare

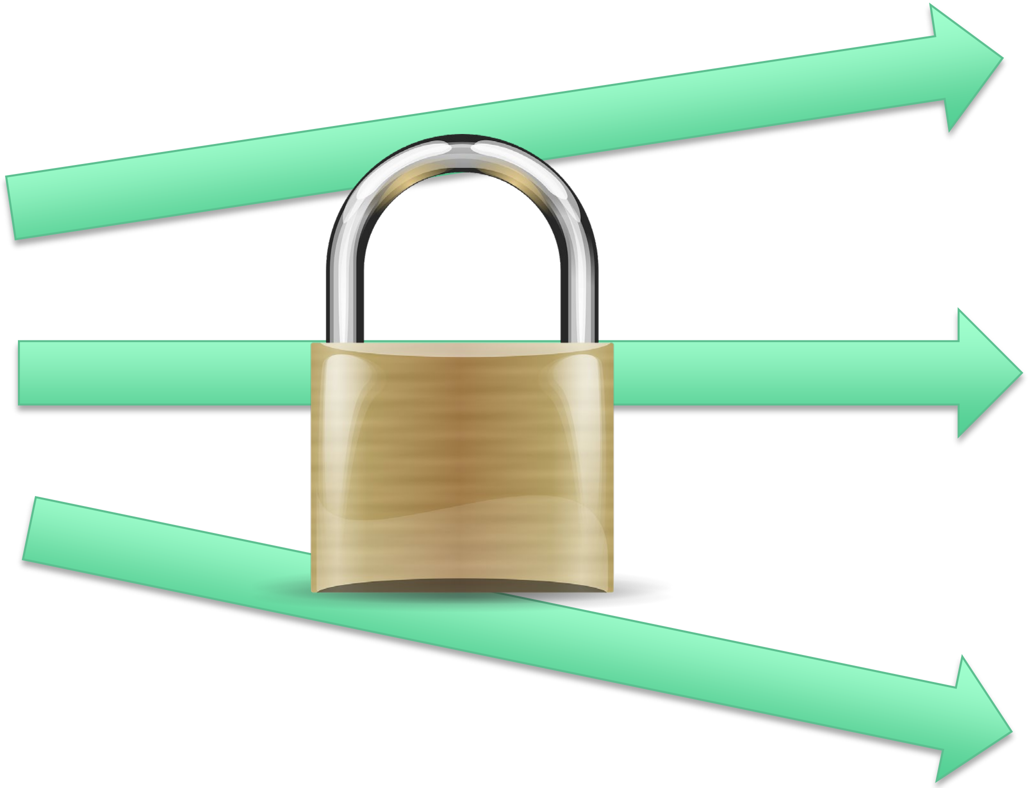
Smart manufacturing

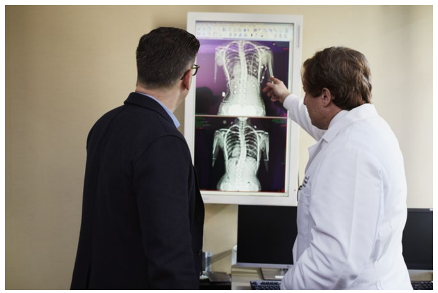
Artificial Intelligence

Privacy protection

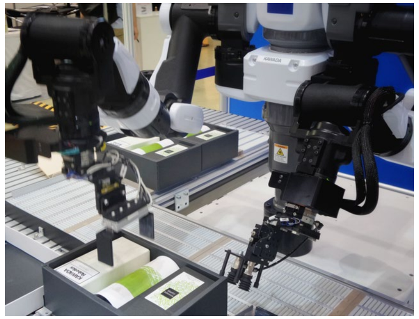
Services

Private data





Healthcare



Smart manufacturing

Artificial Intelligence

We want to ensure privacy of our data when using these services!

GRPD

Sanction:
Up to 4% of the annual worldwide turnover of the preceding year

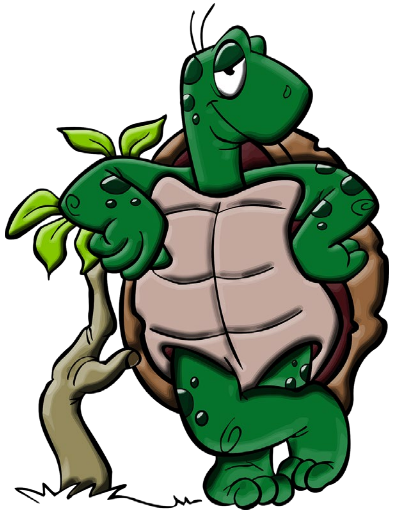
Homomorphic encryption

- Theoretically perfectly suited to address these challenges

Homomorphic encryption

- Theoretically perfectly suited to address these challenges

- … but there are performance issues



Performance issues

- Many ways to explore :
  - Using techniques such as fast bootstrapping when available
  - Using techniques such as batching when available
  - Using cryptosystem with cleartext domain larger than $\mathbb{Z}_2$(when algorithm allows it)
  - Faster homomorphically encryption schemes
  - Specially optimized FHE
  - Truly general purpose FHE works over $\mathbb{Z}_2$:
    - Optimizing directly the Boolean circuit of the homomorphic program
      - Minimizing the ciphertext size
      - Minimizing the execution time

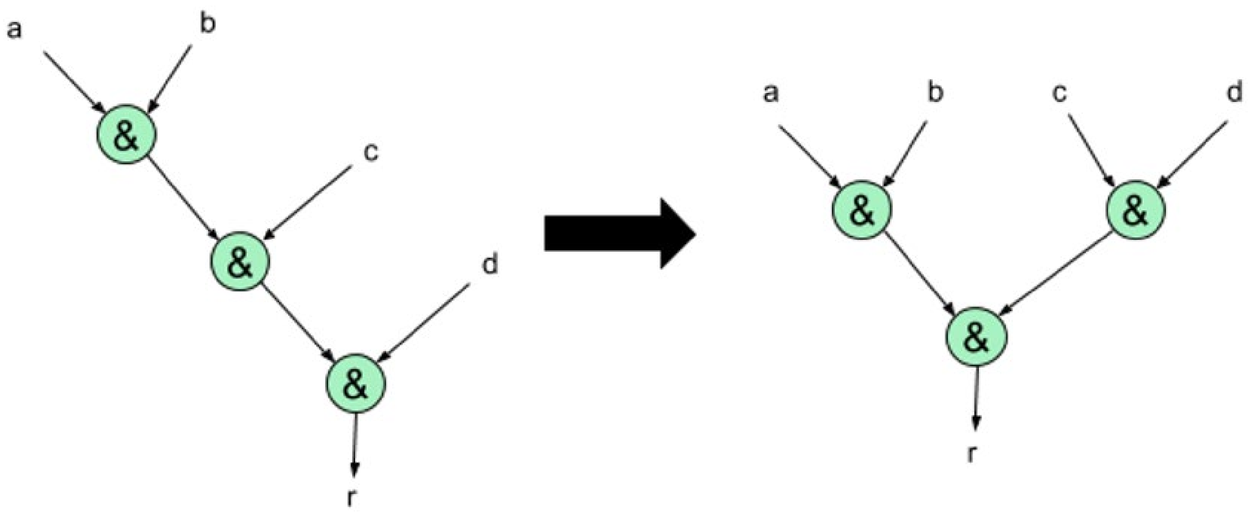
BFV Schemes

- FHE encryption is necessarily probabilistic
  - Ciphertexts have noise which grows after each operation:
    - Noise growth: multiplication >> addition
    - Decryption is impossible above a certain noise level
  - Multiplicative Depth
    - Maximal number of sequential multiplications which are supported by a FHE instantiation
    - Boolean circuit : max number of ANDs between input and output nodes

Multiplicative depth

- Transformation from $\mathbb{Z}_2{}^n$ to $\mathbb{Z}_2$
- Circuit can be developed as a EXOR sum of product
  - Upper bound to minimal multiplicative depth : $[log_2(n)]$.
  - … but the number of gates increase exponentially

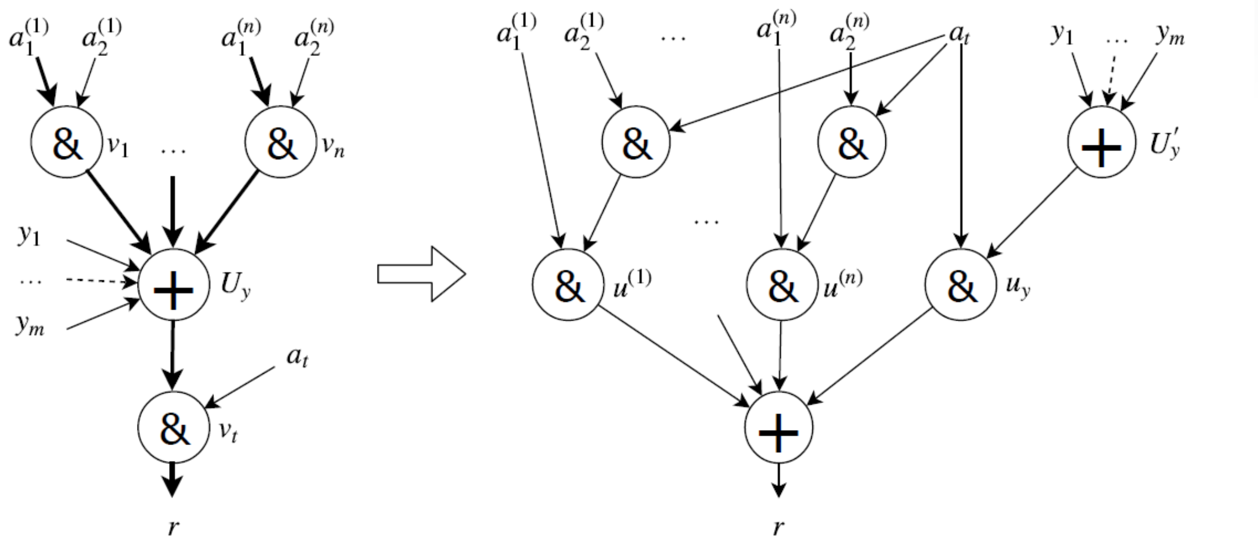
Rewriting operators (Basics)



Associativity of AND: r=((a.b).c).d = (a.b).(c.d)

Multiplicative depth is different for circuits with the same functionality

Rewriting operators (expert)



The equivalent Boolean equation is :

$$\left(\bigoplus_{i=1}^{n}\left(a_1^{(i)}\cdot a_2^{(i)}\right)\oplus\bigoplus_{i=1}^{m}y_i\right)\cdot a_t=\left(\bigoplus_{i=1}^{n}\left(a_t\cdot a_2^{(i)}\right)\cdot a_1^{(i)}\right)\oplus\left(a_t\cdot\bigoplus_{i=1}^{m}y_i\right).$$

Condition to reduce multiplicative depth

- Multiplicative depth of the output r, l(r), is reduced by one with this rewrite operator if :

$$\min_{u \in \mathrm{pred}(v_k)} l(u) < l(v_t) - 2, \forall v_k, k \in \{1, .., n, t\}$$

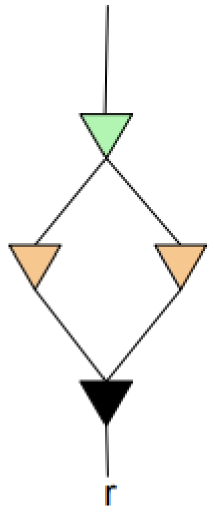
Heuristic

Input: C – input boolean circuit
Output: C_{out} – multiplicative depth optimized

$C_{out} \leftarrow C$
while termination conditions are not verified **do**
    $\Delta^{min} \leftarrow$ compute minimial reducible cones set
    if Δ^{min} is not empty **then**
        Rewrite cones from Δ^{min}
        Update multiplicative depth of C
    end if
    if $l_{\max}(C_{out}) > l_{\max}(C)$ **then**
        $C_{out} \leftarrow C$
    end if
end while

Cone selection method

- Combinatorial optimization problem
  - Known as DAG vertex deletion problem
    - DVD problem is UG-hard
- Network flow based algorithm used to find Δ_{min}



Benchmark suite (EPFL Combinational Benchmark Suite)

Circuit name	#input	#output	x depth	#AND
adder	256	129	255	509
div	128	128	4253	25219
max	512	130	204	2832
multiplier	128	128	254	14389
square	64	128	247	9147
arbiter	256	129	87	11839
i2c	147	142	15	1161
mem_ctrl	1204	1231	110	44795
priority	128	8	203	676
router	60	30	21	167

Experimental results

Circuits name	inital		this work				previous work		
	x depth	# AND	x depth	# AND	ratio	time (s)	X depth	# AND	Time (s)
adder	255	509	**9**	16378	28.3	125	11	1125	**40.0**
div	4253	25219	**532**	190855	8	**3731**	1436	31645	72000
max	204	2832	**26**	7666	7.8	**14**	27	4660	1712
multiplier	254	14389	**57**	23059	4.5	**31**	59	17942	14810
square	247	9147	**26**	11306	9.3	**13**	28	10478	9840
arbiter	87	11839	**10**	5183	8.7	**43**	42	8582	72000
i2c	15	1161	**7**	1213	2.1	**0.1**	8	1185	7.3
mem_ctrl	110	44795	**40**	54816	2.4	**85**	45	49175	66222
priority	203	676	102	876	2.0	**0.5**	102	1106	22.2
router	21	167	11	198	1.9	0.0	11	204	0.5

Run-time complexity

Circuit name	estimated FHE execution acceleration factor		
	this work		previous work
	lowest depth	best	
adder	44.91	408.29	**419.52**
div	10.98	**40.26**	7.66
max	32.04	**61.03**	48.53
multiplier	15.68	17.46	**18.70**
square	105.81	**109.34**	97.10
arbiter	**257.93**	**257.93**	6.69
i2c	**5.16**	**5.16**	3.93
mem_ctrl	**7.43**	**7.43**	6.32
priority	**3.40**	**3.40**	2.69
router	**3.50**	**3.50**	3.40

FHE speed-up obtained at lowest depth found

Best speed-up at the **non necessarly** lowest depth

Tf-idf ranking

- Tf-idf score : $\mathrm{score}(q,d) = \mathrm{coord}(q,d) \,.\, \sum_{t \in q} \mathrm{tf}(t,d)\,.\mathrm{idf}^2(t)$
  - With $\mathrm{idf}(t) = \log\left(\frac{num\_docs}{1+n_t}\right)$
- Developed with Cingulata toolchain
  - Implementation takes account of multiplicative depth

[#terms, #docs]	mult. depth	opt. md	speed-up
[10, 10]	16	15	1.13
[20, 20]	17	15	1.17
[30, 30]	17	15	1.21
[50, 50]	18	16	1.15

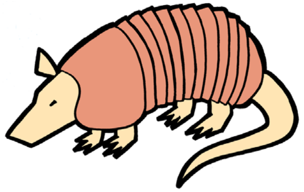
Summary

- Homomorphic encryption is a solution for privacy protection
- There are performance issues
- One solution is working on Boolean circuit optimization
- We proposed an efficient method to do that
- We obtain speed-up for our applications
- There still are many perspectives
  - Heuristic improvements
  - Generalization to arithmetic circuits

Try it by yourself!

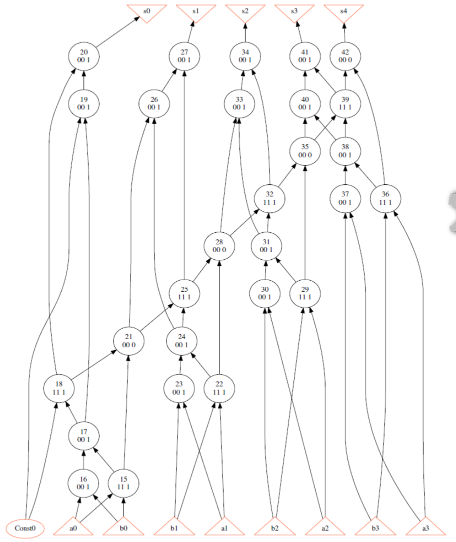
- Homomorphic encryption : Cingulata
  - Cingulata (pronounced "tchingulata") is a compiler toolchain and RTE for running C++ programs over encrypted data by means of fully homomorphic encryption techniques.
  - Open source.
  - Support BFV and TFHE
  - Available on https://github.com/CEA-LIST/Cingulata

Cingulata



transform

and optimize



execute

homomorphically

Cingulata API

Boolean Circuits

Data stay private

RSA Conference2020

RSAConference2020

#RSAC

RSA®Conference2020