

Estudio de viabilidad del uso de librerías de criptografía homomórfica en procesos reales

Autor: Javier Junquera Sánchez

Tutor: Alfonso Muñoz Muñoz



Universidad Europea de Madrid

CURSO 2018/19

- 1 Introducción
- 2 Base teórica
- 3 Librerías
- 4 Implementación
- 5 Resultados
- 6 Conclusiones
- 7 Trabajos futuros

¿Qué es HE?

- **Criptografía**

“[...] actualmente se encarga del estudio de los algoritmos [...] para dotar de seguridad a las comunicaciones [...]” (contributors., [2019a](#))

Cifrado / Descifrado

$$e(m, k_1) = c$$

$$d(c, k_2) = m$$

Simétrico / Asimétrico

$$k_1 = k_2$$

$$k_1 \neq k_2$$

- **Criptografía homomórfica**

“[...] un sistema de cifrado es homomórfico si es capaz de realizar una operación algebraica concreta sobre un texto original, equivalente a otra operación algebraica sobre el resultado cifrado de ese texto original. [...]” (contributors., [2019b](#))

Cifrado maleable

$$e(m_1) \oplus e(m_2) = e(m_1 \oplus m_2)$$

¿Qué es HE?

Un ejemplo “de juguete”

RSA

$$c_1 = e(m_1) = m_1^e \pmod{n}$$

$$c_2 = e(m_2) = m_2^e \pmod{n}$$

RSA en el producto

$$\begin{aligned}c_1 * c_2 &= e(m_1) * e(m_2) \\&= m_1^e * m_2^e \pmod{n} \\&= (m_1 * m_2)^e \pmod{n} \\&= e(m_1 * m_2)\end{aligned}$$

Homomorphic Encryption Standardization

An Open Industry / Government / Academic Consortium to Advance Secure Computation

[Home](#) [Introduction](#) [Standard](#) [Participants](#) [Standards Meetings](#) [Affiliated Workshops](#)
[Mailing Lists](#) [Contact](#)

Homomorphic Encryption

Homomorphic Encryption provides the ability to compute on data while the data is encrypted. This ground-breaking technology has enabled industry and government to provide never-before enabled capabilities for outsourced computation securely.

<https://homomorphicencryption.org/>

Homomorphic Encryption Standarization

Estándar definido principalmente por tres elementos:

- API
- Seguridad de los esquemas
- Aplicaciones de la HE

Además de mantener encuentros de trabajo:

- Identifican librerías
- Generaciones de HE

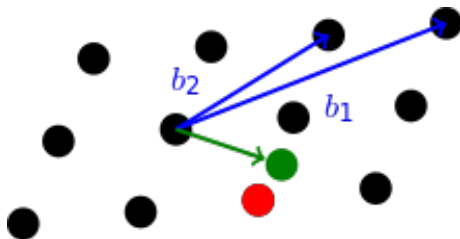
Determinar viabilidad de implementaciones HE para sistemas en producción, en función de:

- Usabilidad
¿Es fácil de usar?
- Capacidad
¿Qué puede hacer?
- Eficiencia
¿Qué coste (computacional, económico...) tiene?

Lattice based cryptography

La seguridad de los sistemas criptográficos se basa en problemas matemáticos:

- Factorización
- Logaritmo discreto
- **Problemas con vectores** \implies *Lattice based crypto*



Regev (Regev, 2010) propone un esquema criptográfico para el problema del aprendizaje con errores (**LWE**)

No solo HE

Los esquemas criptográficos basados en *lattices* se postulan como sistemas *post-quantum*

$$\boxed{\vec{b}} := \boxed{\vec{s}} \times \boxed{A} + \boxed{\vec{\eta}}$$

Cifrado

- 1 Generar valores a_i y $b_i = (a_i \times s + e_i) \pmod{q}$
- 2 Cifrar x : $(c_1, c_2) = (\sum_{j=1}^m a_{ij}, \sum_{j=1}^m b_{ij} + x * (q/2))$

Descifrado

- 1 Descifrar x : $x \approx c_2 - (c_1 * s)$
 - 1 $b + x * (q/2) - a * s$
 - 2 $a * s + e + x * (q/2) - a * s$
 - 3 $e + x * (q/2) \approx x * (q/2)$
- 2 Determinar el resultado
 - 1 $x/(q/2) \approx 0 \implies x = 0$
 - 2 $x/(q/2) \approx 1 \implies x = 1$

```
$ python3 test_lwe.py  
x > 0  
c1: [52840, 310572, 667947, 5318  
c2: [490109, 128329, 317715, 104  
Result: [3, 3, 6]  
$ python3 test_lwe.py
```

<https://asciinema.org/a/qMaeVWh1ktPG9TULjmve7QzNT>

Podemos diferenciar tres tipos de HE:

- Partially Homomorphic Encryption
- Somewhat Homomorphic Encryption (SHE)
- Fully Homomorphic Encryption (FHE)

Y tres generaciones (y pico) de HE:

- Pre-HE: RSA, Boneh–Goh–Nissim
- Primera generación: Técnica de *bootstrapping* (Gentry, [2009](#))
- Segunda generación: BGV, BFV, CKKS...
- Tercera generación: GSW, TFHE...

- Segunda generación
 - HELib
 - **Microsoft SEAL**
 - PALISADE
 - HeaAn
 - LoL
 - NFLlib
- Tercera generación
 - **TFHE**
 - FHEW

Otras dos importantes (para futuro)

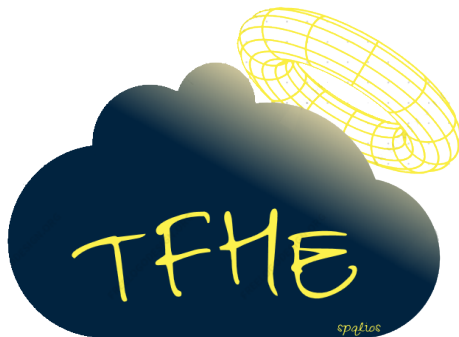
- cuHe (<https://github.com/vernamlab/cuHE>)
- Cingulata (<https://github.com/CEA-LIST/Cingulata>)

Microsoft SEAL



<https://github.com/Microsoft/SEAL>
(*Microsoft SEAL (release 3.3)*, 2019)

- Pocas operaciones
Sumas, restas y productos.
- Capacidad limitada de cómputo
Esquemas de segunda generación SHE (BFV y CKKS)
- Muy buena documentación



<https://github.com/tfhe/tfhe>

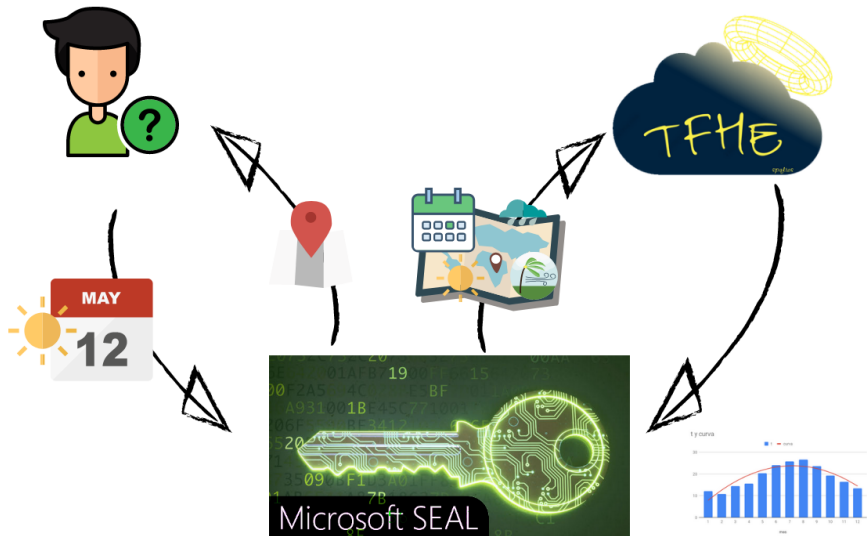
(Chillotti, Gama, Georgieva, y Izabachène, 2016)

- Un esquema: TFHE
- Un parámetro: λ
- Sólo puertas lógicas
 - OR, AND, XOR...
 - NOT
 - ¡MUX!
- Cada puerta tarda $\sim 20\text{ms}$

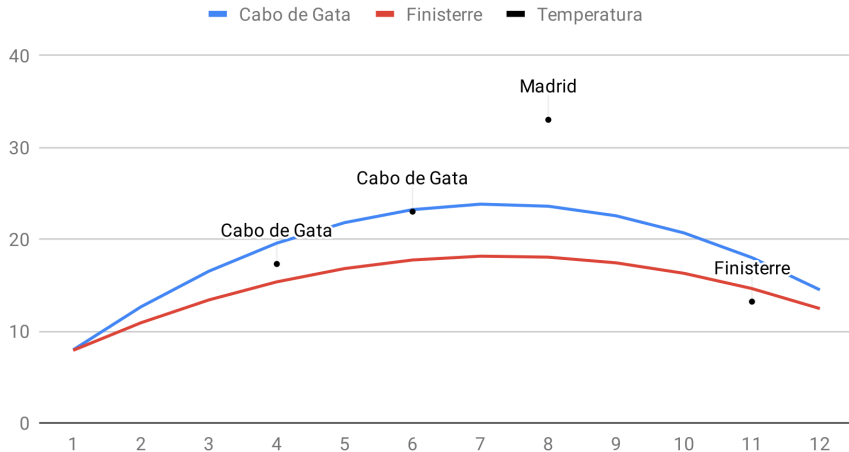
“Si Spiderman puede balancearse sobre su cuerda el tiempo suficiente para lanzar una nueva cuerda, ¡puede volar!”

— El equipo de TFHE

Geoposicionamiento

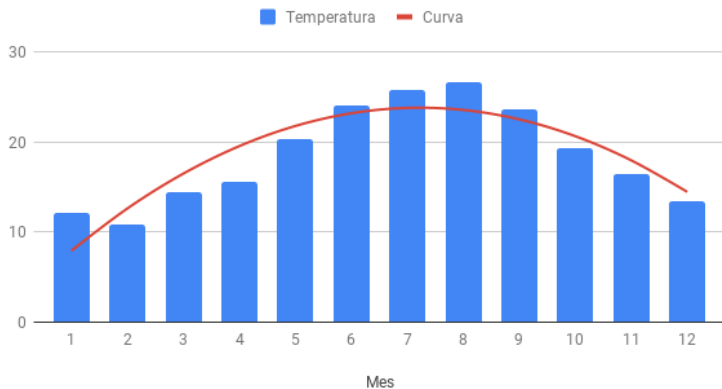


Temperatura / Regresión Cuadrática



- Generar curva de regresión

Regresión cuadrática (Cabo de Gata)



Curva de regresión

Fórmula de curva

$$y = ax^2 + bx + c$$

Parametrización de curva

$$\begin{cases} \sum_{i=0}^n y_i &= a * \sum_{i=0}^n x_i^2 + b * \sum_{i=0}^n x_i + c * n \\ \sum_{i=0}^n x_i * y_i &= a * \sum_{i=0}^n x_i^3 + b * \sum_{i=0}^n x_i^2 + c * \sum_{i=0}^n x_i \\ \sum_{i=0}^n x_i^2 * y_i &= a * \sum_{i=0}^n x_i^4 + b * \sum_{i=0}^n x_i^3 + c * \sum_{i=0}^n x_i^2 \end{cases}$$

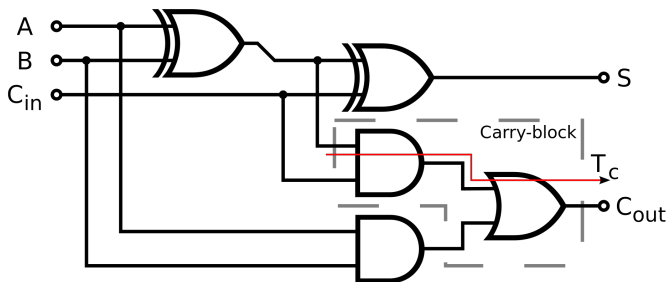
¿Qué necesitamos?

Sólo necesitamos saber a , b y c para reconstruir la curva

$+f(-)3$

<https://gitlab.com/junquera/tfhe-math>
(Junquera, 2019)

- Operaciones aritméticas con puertas lógicas ($+$, $-$, $*$, \div)



- Gestión de signo, decimales y tamaño de palabra


```
junquera@opa:~/Documentos/project
echo "Ok!"
Ok!
junquera@opa:~/Documentos/project
Cliente TFHE
1) Generar datos de Cabo de Gata
2) Generar datos de Finisterre (c
3) Analizar resultados del servic
4) Descifrar archivo
Elija opción > 1
¿Dónde? > ciudad_A
junquera@opa:~/Documentos/project
cloud.key secret.key
junquera@opa:~/Documentos/project
¿Dónde está la clave pública? > 
```


<https://asciinema.org/a/4S69lCdKI25Q6ALYzcqNTTIEa>

- Distancia de punto a curvas (CKKS + *Batching*)

$$\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} = y - (x^2 * \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} + x * \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix})$$

Demo SEAL

```
Inserte temperatura > 22.3
Inserte mes > 6
junquera@opa:~/Documentos/projec
params.data x.data y.data
junquera@opa:~/Documentos/projec
public.key relin.key secret.ke
junquera@opa:~/Documentos/projec
curves
rel
texts
junquera@opa:~/Documentos/projec
public.key relin.key secret.ke
junquera@opa:~/Documentos/projec
curve_names.data params.data r
junquera@opa:~/Documentos/projec
```

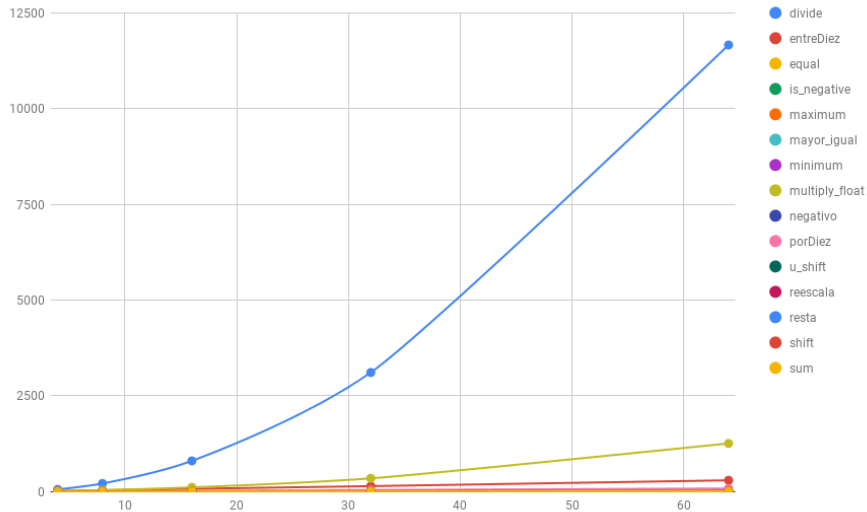


<https://asciinema.org/a/5OTphARYxFoRAkfahycjjTsSA>

Resultados TFHE

operación	puertas lógicas	tiempo estimado	tiempo real
compare_bit	2	0,04	0
equal	128	2,56	3
is_negative	1	0,02	0
minimum/maximum	388	7,76	14
add_bit	5	0,1	0
sum	320	6,4	8
negativo	192	3,84	8
resta	512	10,24	16
multiply	46826	936,52	1257
mayor_igual	128	2,56	4
shiftr/shiftr	771	15,42	19
u_shiftr/u_shiftr	129	2,58	0
divide	85776	1715,52	11662

Crecimiento TFHE



Tiempo cálculo de curva

Cálculo	Tiempo estimado (s)	Tiempo real (s)	Proporción
initVectores	76092	68940	0.90
calcCuadrados	5028	4555	0.90
calcDuplas	11313	10431	0.92
calcComplejos	12570	11522	0.91
CalcC	46776	39818	0.85
CalcB	14208	12222	0.86
CalcA	14192	12200	0.86
Total	180179	159688	0.89
		¡1.84d!	

Muy rápida pero...

- Máximo 19 multiplicaciones realizables teóricamente
- Trabajar con números pequeños (en la práctica, menos de 9 productos, o rompe el esquema)

Estudio teórico	2 meses, 2 horas semana \implies 18 horas
Estudiar libs.	1,5 meses, 2 horas semana \implies 12 horas
Implementación	1 mes, 6 horas día \implies 90 horas

Coste total

Con un sueldo mínimo de 30000€/año^a:

$$120\text{horas} * 15\text{€/hora} = 1800\text{€}$$

^aCuánto gana un ingeniero informático,

<https://universidadeuropea.es/blog/cuanto-gana-un-ingeniero-informatico>

Costes de despliegue



tfm-tfhe

in [javier.junquera.sanchez](#) / 2 GB Memory / 25 GB Disk / NYC3 - Ubuntu 19.04 x64

ipv4: 167.71.249.100

ipv6: [Enable now](#)

Private IP: [Enable now](#)

Floating IP: [Enable now](#)

Graphs

Access

Power

Volumes

Resize

Networking

Backups

Snapshots

Kernel

History

Destroy

Tags

Recovery

Last updated just now.

CPU Usage ?

100%

66%

33%

Aug

Memory usage

100%

08-20-2019 11:01 AM

CPU USAGE

● Total %

LOAD AVERAGE

● 1 min

● 5 min

● 15 min

MEMORY USAGE

● Used %

DISK USAGE

● Local %

DISK I/O

● Read MB/s

● Write MB/s

BANDWIDTH

Aug 25

08-27-2019 06:40 PM

CPU USAGE

● Total 100.00 %

LOAD AVERAGE

● 1 min 2.00

● 5 min 2.00

● 15 min 2.00

MEMORY USAGE

● Used 53.51 %

DISK USAGE

● Local 11.92 %

DISK I/O

● Read 0.00 MB/s

● Write 0.00 MB/s

BANDWIDTH

100%

Aug 11

Ejecución en cloud

Una semana en Digital Ocean^a:

- Un día, un núcleo de CPU: a 5\$/mes
- 6 días, 2 CPUs (al 100 %): a 15\$/mes

El coste final ha sido de 5,07\$ (~ 4,6€)

^a<https://www.digitalocean.com/>

- Medidas basadas en riesgo



- Solucionar problemas irresolubles sin HE, no problemas ya resueltos

/Rooted[®]CON INICIO NOTICIAS EQUIPO FORMACIONES ROOTEDCON VLC 2019 ARCHIVO ROOTEDCON 2019

Sharing privacy. From 0 to Private Set Intersection

Idioma: es

La tendencia actual, tecnologías zero-trust, asume que redes y dispositivos informáticos están comprometidos por un atacante sigiloso. En este escenario, la criptografía puede intentar minimizar el impacto en el robo o manipulación de información. Por desgracia, no es una solución perfecta, ya que puede esquivarse con el uso de malware o ataques de canal lateral, en resumen por el robo de las claves criptográficas. En esta charla, se profundizará en la criptografía homomórfica completa, computación de datos cifrados, analizando su seguridad actual y el ejemplo con diversas librerías de como poder realizar PoCs funcionales. Se mostrará con demos como es posible mezclar esos principios con diversas propuestas, como el protocolo PSI (Private Set Intersection), para añadir mayores niveles de privacidad a aplicaciones de uso común. Se enseñará, por ejemplo, como poder descubrir contactos comunes en una aplicación de mensajería instantánea "segura" sin ceder nuestra agenda de contactos y, por tanto, manteniendo la privacidad.

Alfonso Muñoz Muñoz

PhD in Telecommunications Engineering by Technical University of Madrid (UPM) and postdoc researcher in network security by Universidad Carlos III de Madrid (UC3M). He has been a senior security researcher for more than 10 years and has published more than 60 academic publications (IEEE, ACM, JCR, hacking conferences, X books and numerous security tools). He has also worked in

Conclusiones

- Actualmente, cualquier implementación tiene que ser “bajo supervisión”



Username : admin
Password : admin

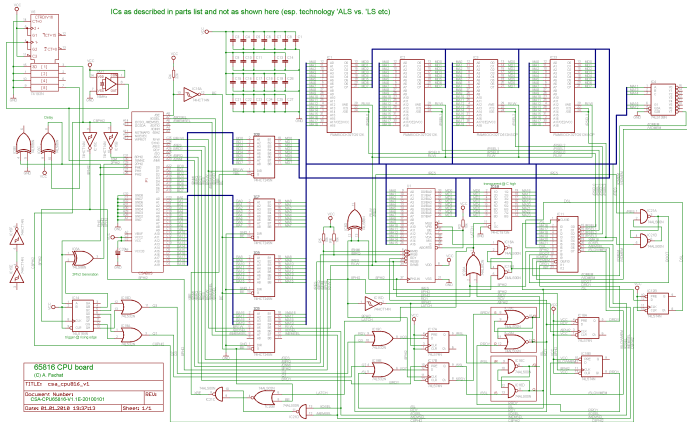
- Hacer implementaciones a medida

```
function out = processEnum(in)

switch in
    case 'north'
        out = 1;
    case 'east'
        out = 2;
    case 'south'
        out = 3;
    case 'west'
        out = 4;
    otherwise
        error('this can not happen')
end
```

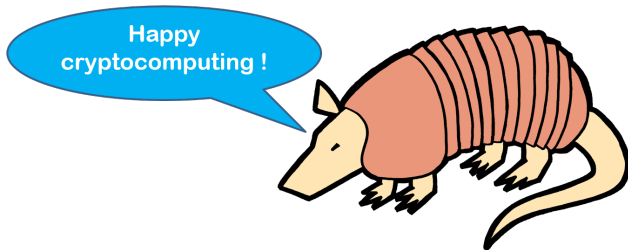
Conclusiones

- Hacer implementaciones a medida



Trabajos futuros

- cuHE¹
Paralelización con GPU
- Cingulata² + ABY³
Compilador de C++ a código HE & Repositorio con circuitos lógicos



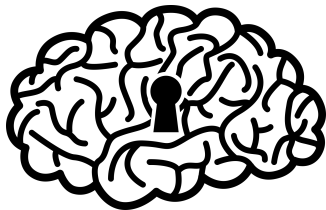
¹<https://github.com/vernamlab/cuHE>

²<https://github.com/CEA-LIST/Cingulata>

³<https://github.com/encryptogroup/ABY>

- Chillotti, I., Gama, N., Georgieva, M., & Izabachène, M. (2016, agosto). *TFHE: Fast Fully Homomorphic Encryption Library*.
- contributors., W. (2019a, julio). Criptografía. Page Version ID: 117340181. Recuperado el 10 de septiembre de 2019, desde <https://es.wikipedia.org/w/index.php?title=Criptograf%C3%ADa&oldid=117340181>
- contributors., W. (2019b, abril). Homomorphic encryption. Page Version ID: 894868556. Recuperado el 10 de mayo de 2019, desde https://en.wikipedia.org/w/index.php?title=Homomorphic_encryption&oldid=894868556
- Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. En *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing* (pp. 169-178). STOC '09. event-place: Bethesda, MD, USA. New York, NY, USA: ACM. doi:10.1145/1536414.1536440
- Junquera, J. (2019, agosto). *TFHE Math Library*. Recuperado desde <https://gitlab.com/junquera/tfhe-math>
- Microsoft SEAL (release 3.3). (2019, junio). Recuperado desde <https://github.com/Microsoft/SEAL>
- Regev, O. (2010, junio). The Learning with Errors Problem (Invited Survey). En *2010 IEEE 25th Annual Conference on Computational Complexity* (pp. 191-204). doi:10.1109/CCC.2010.26

No es difícil de procesar...



¡Es que está cifrado!