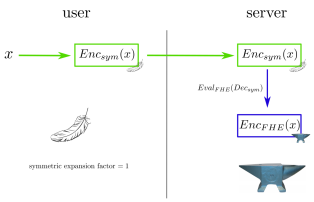
A Homomorphic AES Evaluation in Less than 30 Seconds by Means of TFHE

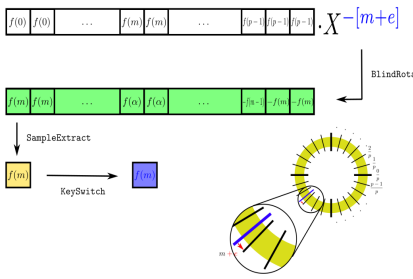
Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga and Renaud Sirdey
Université Paris-Saclay, CEA-List, Palaiseau, France
{name.surname}@cea.fr

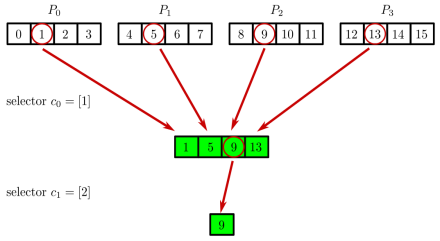


FHE ciphertexts are very far from having an expansion factor of 1.
To reduce the delay in sending FHE data to the server, we prefer the idea of transciphering from a symmetric ciphertext to a homomorphic one.
In this regard, the absolute symmetric standard that is the AES is a crucial benchmark for the research community working on transciphering. And especially since AES execution may be the application chosen by NIST in the FHE part of its pre-call for proposals on threshold encryption[1].

[1] See https://csrc.nist.gov/Projects/threshold-cryptography.

TFHE Programmable Bootstrapping





Tree-based Method

This method allows to perform TFHE programmable bootstrapping on several encrypted inputs. Here we perform the identity bootstrapping on the encrypted message $m = ([1], [2])$ which encrypts the integer 9, decomposed in two basis 4 digits.

Parameters for methods allowing bootstrapping on several incrypted inputs

	basis	n	N	l	B_g	B_{KS}	t	q	ϵ	TRLWE noise	TLWE noise
chaining method	4	850	2048	2	2048	1024	2	32	2^{-32}	9.6×10^{-11}	1.27×10^{-6}
	8	1024	8192	1	268435456	1024	2	128	2^{-27}	10^{-45}	5.6×10^{-8}
	16	1100	32768	1	4294967296	8192	2	512	2^{-30}	1.4×10^{-8}	10^{-248}
tree-based method	4	700	1024	5	16	1024	2	8	2^{-30}	5.6×10^{-8}	1.9×10^{-5}
	8	700	2048	2	2048	1024	2	16	2^{-23}	9.6×10^{-11}	1.9×10^{-5}
	16	1024	2048	3	256	1024	2	32	2^{-23}	9.6×10^{-11}	6.5×10^{-8}

This table compares the parameters needed depending on the method. B_g and l denote the basis and levels associated with the gadget decomposition, B_{KS} and t denote the decomposition basis and the precision of the decomposition of the `KeySwitch`. Finally, q denotes the size of the used plaintext size (meaning that the torus is discretized on q values), and ϵ is the error probability of one MVB evaluation or one evaluation of the chaining method.
As shown here, parameters for the chaining method are very large and do not allow an efficient functional bootstrapping evaluation. That is why we chose to work with the tree-based method, which requires smaller parameters and is thus more efficient.

As seen here, due to the small number of bootstrapping needed, it is the basis 16 that offers the best timing execution regarding a complete LUT evaluation.

basis	# XOR	# LUT
256	2	1
16	4	4
8	6	30
4	8	88

basis	single boot.	complete LUT eval.	n	N
256	1.5s	1.5 s	1024	32768
16	0.029s	**0.3 s**	1024	2048
8	0.015s	1.4 s	700	2048
4	0.007s	2.0 s	700	1024

Number of bootstrapping for one XOR or LUT evaluation (using MVB) depending on the decomposition basis. Unitary timings for bootstrapping and full LUT evaluation depending on basis and parameter choice.

Advanced Encryption Standard

An AES ciphertext is composed of 16 bytes such as $c = c_0 c_1 \cdots c_{15} \in (\mathbb{F}_{2^8})^{16}$ and incoded in the *state matrix* in the following way:

$$\begin{pmatrix} c_0 & c_4 & c_8 & c_{12} \\ c_1 & c_5 & c_9 & c_{13} \\ c_2 & c_6 & c_{10} & c_{14} \\ c_3 & c_7 & c_{11} & c_{15} \end{pmatrix}$$

We perform ten rounds of the following operations:

- `SubBytes`, a substitution consisting of an S-Box applied to the bytes of the stae matrix.
- `ShiftRows`, a byte transposition that cyclically shifts the rows of the state over different offsets.
- `MixColumns`, which operates on the state column by column via matrix multiplication.
- `AddRoundKey`, which modifies the state by combining it with a round key using the bitwise XOR operation.

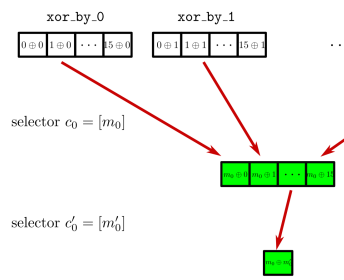
```
word8 mul(word8 a, word8 b) {
   /* multiply two elements of GF(256)
    * required for MixColumns and InvMixColumns
    */
   if (a && b) return Alogtable[(Logtable[a] + Logtable[b])%255];
   else return 0;
}
```

→

```
word8 mul_a(word8 b) {
return T_a[b];
}
```

$GF(256)$ Homomorphic Multiplication

The orignal AES code (seen on the left) is not optimal for a homomorphic evaluation, as it requires six operations in order to proceed to a simple $GF(256)$ multiplication.
We deal with this issue by observing that the parameter a only takes 6 different *plaintext* value. So we transform the bivariate ciphertext-ciphertext `mul` function into six univariate `mul_a` functions. To do so, we create six new tables giving the result of the multiplication of a ciphertext b by a cleartext value a.
This allows us to perform the $GF(256)$ multiplication with only one table indirection instead of three indirections, an addition, a modulo operation and an *if* condition as in the original implementation.



Here we want to perform the bitwise XOR of two 4-bit ciphertexts c_0 and c_1 encrypting m_0 and m_1. The first step is to create the polynomials that will be shifted during the bootstarpping. Each one of them must encode the values of a unary XOR. We then apply the bootstrapping to these polynomials with selector c_0, which gives us 16 new ciphertext encoding the values of the bitwise `XOR_by_m_0` polynomial.
By applying the bootstrapping on this new polynomial with selector m_1, we finally obtain an encryption of $m_0 \oplus m_1$.

Homomorphic Xor Example

Results

	execution time	time ratio
`i7-laptop` (1 thread)	4.5 mins = 270 secs	9. 4
`i7-laptop` (6 threads)	54.31 secs	1.9
`AMD-server` (1 thread)	5.7 mins = 342 secs	11.9
`AMD-server` (16 threads)	36.39 secs	1.3
`"i7-server"` (16 threads)	28.73 secs	1
Gentry et al.(1 thread)	18 mins	37.6
Mella and Susella (1 thread)	22 mins	45.9
Stracovsky et al. (16 threads)	4.2 mins = 252 secs	8.8

We used the OpenMP library to parallelize our TFHElib code and optimize the execution timings of our homomorphic AES.

[1] Sergiu Carpov, Malika Izabachène, and Victor Mollimard. 2018. New techniques for Multi-value input Homomorphic Evaluation and Applications. Cryptology ePrint Archive, Paper 2018/622. https://eprint.iacr.org/2018/622.
[2] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. August 2016. TFHE: Fast Fully Homomorphic Encryption Library. https://tfhe.github.io/tfhe/.
[3] Ilaria Chillotti, Marc Joye, and Pascal Paillier. 2021. Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks. In *Cyber Security Cryptography and Machine Learning*, Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander Schwarzmann (Eds.). Springer International Publishing, Cham, 1–19.
[4] Joan Daemen and Vincent Rijmen. 2002. *The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography)* (1 ed.). Springer.
[5] Craig Gentry, Shai Halevi, and Nigel P. Smart. 2012. Homomorphic Evaluation of the AES Circuit. In *Advances in Cryptology – CRYPTO 2012*, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 850–867.
[6] Antonio Guimarães, Edson Borin, and Diego F. Aranha. 2021. Revisiting the functional bootstrap in TFHE. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 2 (Feb. 2021), 229–253. https://doi.org/10.46586/tches.v2021.i2.229-253
[7] Silvia Mella and Ruggero Susella. 2013. On the Homomorphic Computation of Symmetric Cryptographic Primitives. In *Cryptography and Coding*, Martijn Stam (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 28–44.
[8] Roy Stracovsky, Rasoul Akhavan Mahdavi, and Florian Kerschbaum. 2022. Faster evaluation of AES using TFHE. Poster Session, FHE.Org.