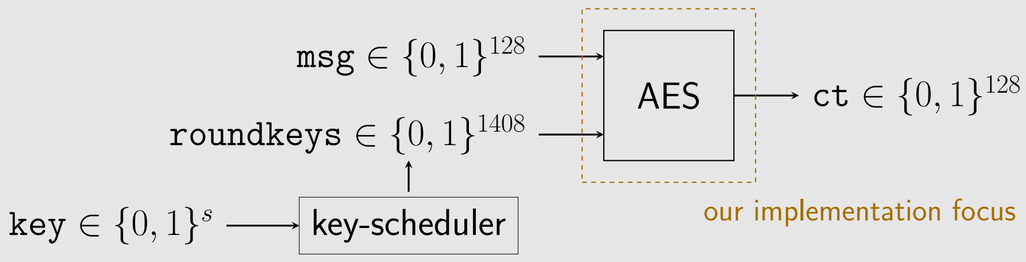
Faster Evaluation of AES with TFHE

Roy Stracovsky Rasoul Akhavan Mahdavi Florian Kerschbaum

University of Waterloo

Advanced Encryption Standard (AES)

AES is a symmetric block cipher that operates on an internal 4×4 matrix of bytes known as the state.



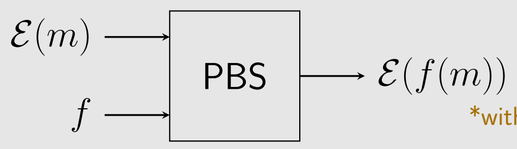
It applies the following round operations on the state for 10 rounds during encryption.

- **SubBytes:** Applies an s-box to each byte.
- **ShiftRows:** Swaps the positions of bytes.
- **MixColumns:** Performs linear combinations on each column.
- **AddRoundKey:** Adds the round key to the round key.

TFHE and Concrete

TFHE is a fully homomorphic encryption (FHE) scheme based on the learning with errors (LWE) problem which **allows for efficient programmable bootstrapping** (PBS).

Concrete implements Zama's variant of TFHE and operates on encrypted floats. Floats are encoded onto the real Torus $\mathbb{R}/\mathbb{Z}$ and special attention must be paid to padding and bit precision. Concrete operations include subtraction, multiple types of addition, and programmable bootstrapping (which we denote $\mathsf{PBS}_{f(\cdot)}(c)$).



These operations allow for other important operations. For example ct-ct homomorphic multiplication (which we denote $\odot$) leverages the identity that

$$ab = \left\lfloor \frac{(a+b)^2}{4} \right\rfloor - \left\lfloor \frac{(a-b)^2}{4} \right\rfloor.$$

Motivation

Encryptions of pseudo-random numbers (PRNs) are used in secure protocols with preprocessing. Usually these encrypted PRNs are generated by the client and sent to the server. Homomorphic evaluation of AES allows the server to directly generate PRNs, reducing communication costs.

In addition, evaluating AES acts as a litmus test for the practicality of FHE schemes and we aim to show that TFHE can be used in complex applications.

Experimental Results

Our implementation of AES shows the **fastest** single-evaluation (total/non-amortized) time of any existing implementation **with a fresh output which may be directly used in further computation**.

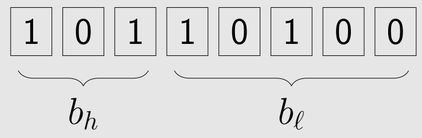
		Total Time	# of Blocks	Security	Key Sizes	Fresh
Single Eval.	Our Work ($k=4$)	**4.2m**	1	**128**	**365MB**	✓
	Our Work ($k=5$)	8.25m	1	94/128	658MB	✓
	MS13 (packed)	22m	1	80	-	✗
Amortized Eval.	GHS15 (w/ bs)	17.5m	180	123	3.7GB*	✓
	GHS15 (w/o bs)	4.1m	120	150	3GB*	✗
	DHS16	25h	1800	128	13GB	✗

The parameters we use in our implementations are as below.

	n	N	$\log q$	γ	ℓ	κ	t	Security
$k=4$	750	2048	64	7	3	2	7	128-bit
$k=5$	750	4096	64	24	1	1	20	128-bit

Implementing AES in Concrete

We encode and encrypt each state byte into two ciphertexts due to limitations on practical parameters for TFHE. For example



is encrypted as $(c_h, c_\ell) = (\mathcal{E}(b_h), \mathcal{E}(b_\ell)) = (\mathcal{E}(101), \mathcal{E}(10100))$. We use k to denote the number of bits in b_ℓ.

Homomorphic byte operations

- **XOR:** Computed using PBS to extract bits from the operands, bitwise XOR via addition/subtraction and PBS, and recombining bits into a byte. Concretely, we compute the XOR between two ciphertexts c_1, c_2 with

$$\sum_{i=0}^{k'} \mathsf{PBS}_{2^i \times (\cdot \bmod 2)} \left(\mathsf{PBS}_{\mathsf{eb}_i(\cdot)}(c_1) + \mathsf{PBS}_{\mathsf{eb}_i(\cdot)}(c_2) \right)$$

  which minimizes the effect of noise growth between bootstraps.
- **S-box:** Can be computed using PBS to evaluate $\mathsf{sbox}(2^k i \times c_\ell)$ for $0 \le i < 2^{8-k}$ and PBS, ct-ct multiplication, and addition to select the output corresponding to the correct input pair. Concretely, we implement s-box on an encrypted byte (c_h, c_ℓ) with

$$\left(\sum_{i=0}^{2^{8-k}-1} \mathsf{PBS}_{\mathsf{eq}_i}(c_h) \odot \mathsf{PBS}_{\mathsf{sbox}_h}(i, c_\ell), \sum_{i=0}^{2^{8-k}-1} \mathsf{PBS}_{\mathsf{eq}_i}(c_h) \odot \mathsf{PBS}_{\mathsf{sbox}_\ell}(i, c_\ell) \right)$$

  where $\mathsf{eq}_i(x)$ denotes the equality function which outputs 1 if $x = i$ or 0 otherwise, and sbox_h, sbox_ℓ receive a tuple (a, b) and output the appropriate higher/lower order bits of $\mathsf{sbox}(2^k \times a + b)$.
- **Mul2:** Consists of a left-shift with a conditional modulo. The left shift can be realized via PBS and adding the highest order bit of c_ℓ to c_h. The conditional modulo can be done by generating the appropriate XOR value via PBS and the homomorphic XOR from above. Concretely we compute,

$$\Big(\left(\mathsf{PBS}_{.<<1}(c_h) + \mathsf{PBS}_{\mathsf{eb}_{k-1}}(c_\ell) \right) \oplus \mathsf{PBS}_{\mathsf{eq}_1(\mathsf{eb}_{7-k}(\cdot))}(c_h), \mathsf{PBS}_{.<<1}(c_\ell) \oplus \mathsf{PBS}_{11 \times \mathsf{eq}_1(\mathsf{eb}_{7-k}(\cdot))}(c_h) \Big)$$

Homomorphic AES

The AES round operations following directly from the byte operations. As the round operations are naturally parallelizable, we update the value of each byte in the state matrix on a separate thread for further efficiency.

Related Work

- Gentry et al. [GHS12] implement AES using BGV, with and without bootstrapping, implemented in HElib with low amortized runtime.
- Mella and Susella [MS13] also implement AES using BGV, however they focus on low total time. They provide multiple implementations which partition and encrypt the state differently.
- Doroz et al. [DHS16] implement an amortized evaluation of AES using the LTV cryptosystem, and propose several optimizations to enhance the performance of LTV.

References

Yarkın Doröz, Yin Hu, and Berk Sunar. "Homomorphic AES evaluation using the modified LTV scheme". In: *Designs, Codes and Cryptography* 80.2 (2016), pp. 333–358.

Craig Gentry, Shai Halevi, and Nigel P Smart. "Homomorphic evaluation of the AES circuit". In: *Annual Cryptology Conference*. Springer. 2012, pp. 850–867.

Silvia Mella and Ruggero Susella. "On the homomorphic computation of symmetric cryptographic primitives". In: *IMA International Conference on Cryptography and Coding*. Springer. 2013, pp. 28–44.