Optimized Homomorphic Evaluation of Boolean Functions

Nicolas Bon, David Pointcheval, Matthieu Rivain

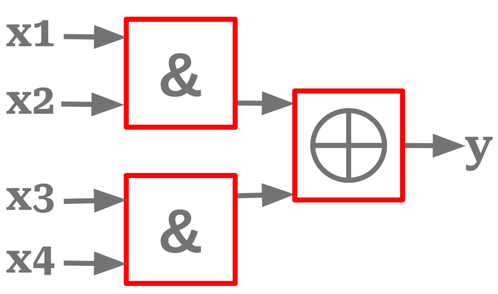
CryptoExperts, Paris

Ecole Normale Supérieure-PSL, Paris

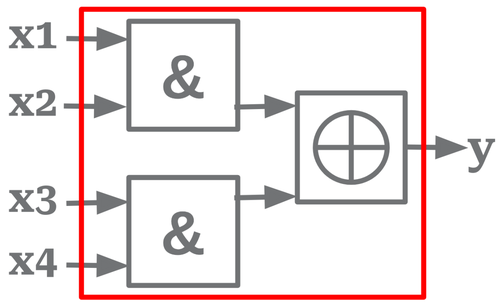
nicolas.bon@cryptoexperts.com

Gadgets are better than gates

To evaluate Boolean Functions with TFHE, the natural gate bootstrapping approach is inefficient.



Gate approach: 3 Bootstrapping required.



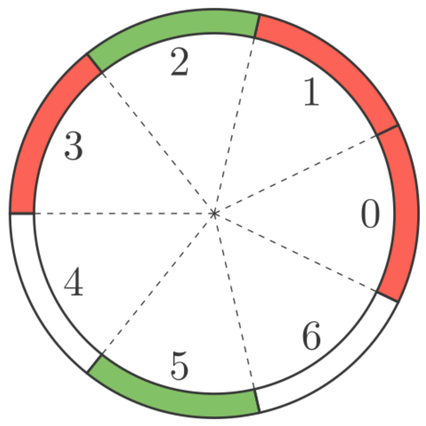
Gadget approach: 1 Bootstrapping required.

Our solution is to wrap Boolean Function into a **gadget** that is evaluated in **one single bootstrapping**, no matter the number of inputs.

New Encoding of bits into plaintexts

A bit $b \in \mathbb{Z}_2$ is encoded by a part of the discretized torus. Parts in red encode the Boolean ZERO and parts in green encode the Boolean ONE.

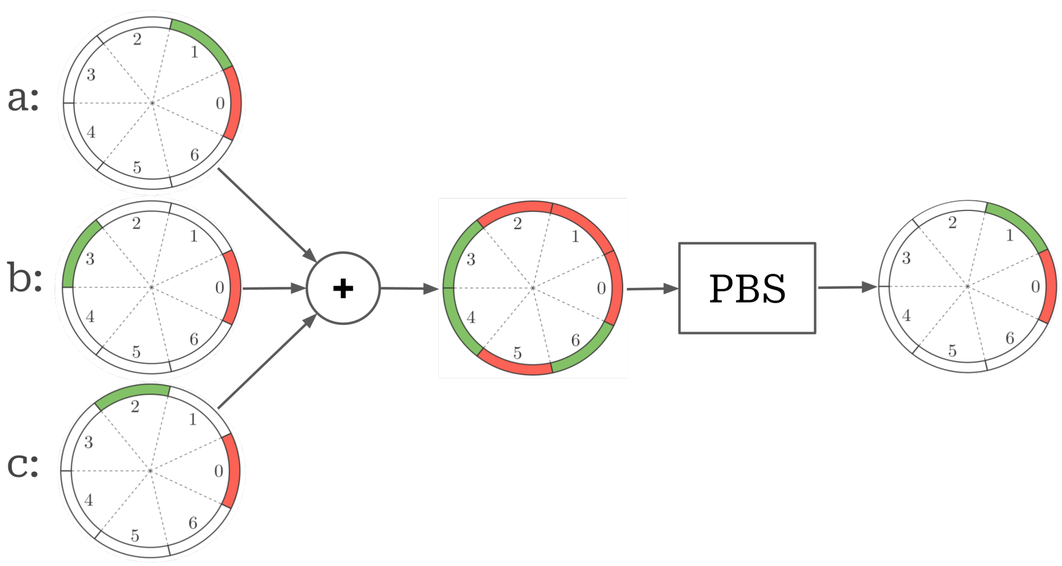
Such a mapping is called a p-**encoding**. RED and GREEN should never overlap.



Example of a 7-encoding.

Inner working of gadgets

With well-chosen p-encodings for the inputs, a simple sum followed by a bootstrapping allows to extract the result of the function. Figure 4 is an example for the multiplexing function: $f(a,b,c) = \begin{cases} a \text{ if } c = 1 \\ b \text{ if } c = 0 \end{cases}$.



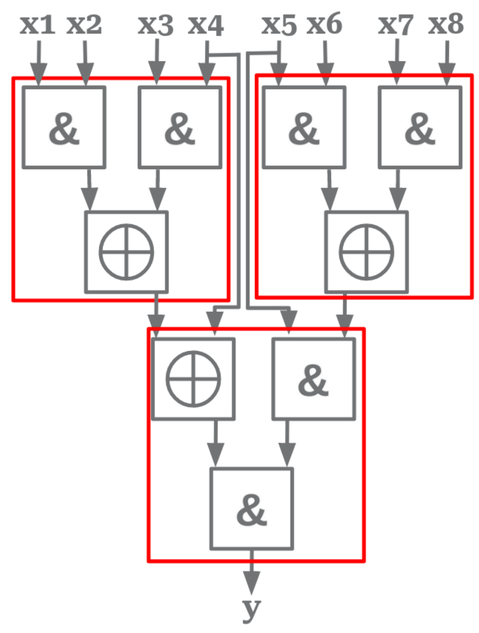
If the input encodings are well-chosen, red and green parts of the torus do not mix. Relabelling the intermediary torus is done using the truth table of f.

Generating the appropriate p-encodings for a given function f is done by an efficient **search algorithm** we designed. It constructs a valid gadget for a given value of p if such a gadget exists. Especially, functions with more than six inputs are unlikely to have a valid associated gadget for a p small enough to have good performances with the TFHE scheme.

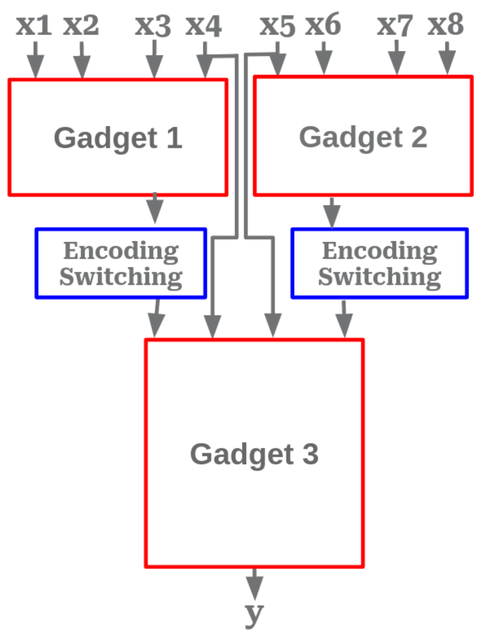
To select the right value for p, we also developed a **sieving heuristic**.

Graph of gadgets

Using **prime** values for p allows to homomorphically switch the encoding of a bit with a simple multiplication by a clear constant. Thus, we can evaluate a large Boolean circuit by segmenting it into smaller functions that can be evaluated with a gadget.



Segmentation of a large circuit



An encoding switching operator allows to plug gadgets into each other.

We have developed a **heuristic algorithm** to perform this segmentation for any graph. Especially, we apply it to the **s-box of AES**.

Experimental results

We applied this framework to some cryptographic primitives with specially tailored sets of parameters. Our results allow an improvement in performance over the state of the art. Our experiences have been carried on a laptop.

Primitive	Section or Other work	Timing
One full run of SIMON	Gate Bootstrapping	174 s
	[Ben+23] †	128 s
	Our work	**10 s**
One warm-up phase of Trivium (*)	Gate Bootstrapping	1498 s
	[BOS23] (estimation on our machine)	53 s
	Our work	**32.8 s**
One Full Keccak permutation $(*)$	Gate Bootstrapping	30.7 min
	Our work	**8.8 min**
One Ascon hashing $(*)$	Gate Bootstrapping	200s
	Our work	**92 s**
One full evaluation of AES-128 ($\epsilon = 2^{-23}$) on one thread	[GHS12] †	18 min
	[CLT14] †	5 min
	[Tra+23]	270 s
	Our work	**103 s**
One full evaluation of AES-128 ($\epsilon = 2^{-40}$) on one thread	Gate Bootstrapping	234 s
	Our work (Real implementation)	**135 s**
	Our work (Theoretical timing with two keys)	**105 s**

Table 1: Timings of evaluation of full primitives, and comparison with previous works when they exist. A star $(*)$ is added in the cells if our timing is not obtained from a full implementation but estimated from an implemented building block. Also, the security level of each implementation is $\lambda = 128$ and the default error probability is $\epsilon = 2^{-40}$. The concurrent works that do not indicates their ϵ are marked with †.

Full paper and slides