On the Minimal Number of Bootstrappings in Homomorphic Circuits

Tancrède Lepoint[1,2] and Pascal Paillier[1]

[1] CryptoExperts, France
[2] École Normale Supérieure, France
{tancrede.lepoint,pascal.paillier}@cryptoexperts.com

Abstract. We propose a method to compute the *exact* minimal number of bootstrappings required to homomorphically evaluate any circuit. Given a circuit (typically over $\mathbb{F}_2$ although our method readily extends to circuits over any ring), the maximal noise level supported by the considered fully homomorphic encryption (FHE) scheme and the desired noise level of circuit inputs and outputs, our algorithms return a minimal subset of circuit variables such that boostrapping these variables is enough to perform an evaluation of the whole circuit. We introduce a specific algorithm for 2-level encryption (first generation of FHE schemes) and an extended algorithm for $\ell_{\max}$-level encryption with arbitrary $\ell_{\max} \geqslant 2$ to cope with more recent FHE schemes. We successfully applied our method to a range of real-world circuits that perform various operations over plaintext bits. Practical results show that some of these circuits benefit from significant improvements over the naive evaluation method where all multiplication outputs are bootstrapped. In particular, we report that a circuit for the AES S-box put forward by Boyar and Peralta admits a solution in 17 bootstrappings instead of 32, thereby leading to a 88% faster homomorphic evaluation of AES for any 2-level FHE scheme.

Keywords: Fully Homomorphic Encryption, Bootstrapping, Boolean Circuits, AES S-box.

1 Introduction

Fully homomorphic encryption (FHE) allows a worker to evaluate any circuit on plaintext values while manipulating their encryption in a public fashion *i.e.* with no knowledge of the decryption key. Gentry's original proposal [13] introduced a design principle that was later followed by a lot of FHE schemes [13,12,20,9,5,10,8]. Inherent to this design principle is the property that ciphertexts contain some noise which grows with successive homomorphic multiplications; thus ciphertexts need to be *refreshed* to maintain a low level of noise and allow subsequent homomorphic operations. In order to refresh ciphertexts, Gentry's key idea, referred to as *bootstrapping*, consists in homomorphically evaluating the decryption circuit of the FHE scheme using the decryption key bits in encrypted form, thus resulting in a different encryption of the same plaintext but with reduced noise. One ensures

A.A. Adam, M. Brenner, and M. Smith (Eds.): FC 2013, LNCS 7862, pp. 189–200, 2013.

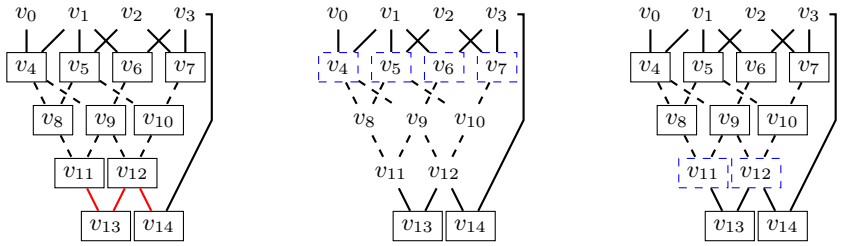
that the scheme parameters are such that the refreshed ciphertexts can handle at least one additional homomorphic multiplication. By repeating this procedure, the number of homomorphic operations becomes unlimited, thereby yielding a *fully* homomorphic encryption scheme.

Noise Levels. In all known FHE schemes, a ciphertext c_i contains a noise r_i which grows along with homomorphic multiplications and decryption is ensured as long as r_i does not exceed a given bound, *i.e.* $r_i < r_{\max}$. Without loss of generality, we can assume that the noise is lower-bounded by the noise after a bootstrapping operation[1]. We adopt a simplified approach by associating with each ciphertext c_i a *discretized noise level* $\ell_i = 1, 2, \ldots$, where 1 is the noise level of ciphertexts resulting from a bootstrapping operation. Let c_1 (resp. c_2) be a ciphertext with noise level ℓ_1 (resp. ℓ_2). Gentry-like FHE schemes are such that $c_3 = c_1 + c_2$ has noise level $\ell_3 = \max(\ell_2, \ell_1)$ and $c_3 = c_1 \times c_2$ has noise level $\ell_3 = \ell_1 + \ell_2$, where $+$ and $\times$ respectively denote homomorphic addition and multiplication. Therefore in these schemes, the noise level grows exponentially with the number of homomorphic multiplications: to evaluate a circuit with L sequential layers of multiplications, one must impose the maximum noise level $\ell_{\max}$ to be larger than 2^L. This is practically unacceptable even for small values of L and one must resort to bootstrapping periodically as the circuit is being evaluated.

Note that our definition of noise levels neglects the logarithmic increase of the noise size after a homomorphic addition. This approximation is often considered in the literature and remains valid as long as the proportion of additions does not become overwhelming in the circuit. Clearly, our simplified model would become invalid outside of this context.

Exponential vs. Linear FHE Schemes. For the purpose of this work, the above schemes will be referred to as being *exponential*. Recently, Brakerski, Gentry and Vaikuntanathan described a different framework where the ciphertext noise grows only linearly with the number of performed multiplications instead of exponentially [4]. This framework was used in several subsequent works [15,18,16] and even improved [3]. These FHE schemes are said to be *linear* throughout the paper. In linear schemes, homomorphic addition still outputs ciphertexts of level $\ell_3 = \max(\ell_1, \ell_2)$. However, a homomorphic multiplication $c_3 = c_1 \times c_2$ now results in a noise level $\ell_3 = \max(\ell_1, \ell_2) + 1$. Thus, to evaluate a circuit with L layers of multiplications, one only requires $\ell_{\max} \geqslant L$. However, when the depth of the circuit is not known at key generation time, this improvement is not strong enough to completely eliminate the need for intermediate bootstrappings.

[1] Notice that in most FHE schemes, freshly generated ciphertexts have a smaller noise than the noise obtained after a bootstrapping operation, allowing the circuit evaluator to save several bootstrappings at the beginning of the circuit. However, it is very likely that in *real-world* applications, data to be evaluated homomorphically will have been pre-processed and will not contain the smallest possible noise anymore.



(a) With no bootstrapping (b) Bootstrapping after each multiplication (c) Minimal solution

Fig. 1. Different bootstrapping solutions in a FHE scheme with $\ell_{\max} = 2$. Plain lines represent homomorphic multiplications while dashed lines represent homomorphic additions. The red lines in (a) reveal that the ciphertext noise will exceed the noise limit. Variables in a plain rectangle have a "large" noise ($\ell_i = \ell_{\max} = 2$) and the ones in a dashed blue rectangle are bootstrapped *i.e.* are re-encrypted to convert a "large" noise ($\ell_i = 2$) into a "small" noise ($\ell_i = 1$).

Minimizing Bootstrappings. Overall, both exponential and linear FHE schemes must resort to boostrapping in homomorphic circuit evaluation, either periodically or once in a while. However, the bootstrapping operation is reported as being the most drastic computational bottleneck in all known FHE implementations [14,9,10,16,8]. Worse, most of them merely perform a bootstrapping operation right after each multiplication, as suggested in [13,12]. It is easily seen though, as shown by the toy example depicted on Fig. 1, that this simple approach is often not optimal and that *fewer* bootstrappings may be sufficient to evaluate the whole circuit if positioned more judiciously.

Note that, even though finding a minimal solution is trivial and easily done by hand in Fig. 1, this optimization problem seems to become far more difficult with (even slightly) more complex circuits. Automated tools are therefore necessary to identify (one of) the smallest possible set of circuit variables whose bootstrapping will ensure a complete circuit evaluation in minimal time.

Contributions and Outline. We propose two efficient algorithms that automatically find an *exact* minimal solution for any given circuit *i.e.* output a minimal list of circuit variables to which bootstrapping can be applied to evaluate the circuit. Section 2 introduces a first algorithm specific to the case of FHE schemes with a maximum noise level set to $\ell_{\max} = 2$. This covers both exponential and linear schemes since the two categories collide in this particular case. In Section 3, we extend our algorithm to support exponential FHE schemes handling up to $\ell_{\max} \geqslant 3$ noise levels. We show that the same extended algorithm can also be used with linear schemes via a problem reformulation. Finally, Section 4 reports a number of experimental results on a range of real-world circuits, namely the benchmarking circuits for MPC and FHE proposed by Smart and Tillich [19], as well as circuits implementing the AES S-box suggested by Boyar and Peralta [2,1].

2 Homomorphic Schemes with 2 Noise Levels

In this section, we consider a FHE scheme that can only handle two levels of randomness in ciphertexts, *i.e.* level-1 ciphertexts can either be added (yielding a level-1 ciphertext) or multiplied (yielding a level-2 ciphertext); however only addition can be performed on ciphertexts with levels $(1,2),(2,1)$ or $(2,2)$ since the result of a multiplication would not be decryptable. As a result, the scheme can only handle a single multiplication after each bootstrapping operation. This framework was heavily considered [13,12,9,5,10,8] and implementations are available [11,7].

2.1 Stating the Problem

Let $\mathsf{C} = \mathsf{C}(n_1, n_2)$ be a Boolean circuit made of AND, XOR and NOT gates which takes as input n_1 bits and outputs n_2 bits. We denote by $\mathsf{C}^\dagger$ the same circuit as C where gates are replaced with homomorphic additions and multiplications[2]. Feeding $\mathsf{C}^\dagger$ with n_1 encrypted bits (under the FHE scheme), it will then output n_2 encrypted bits corresponding to the outputs of C applied on the same input bits in the clear. We denote by $\mathsf{V} = \{v_i : 1 \leqslant i \leqslant n\}$ the set of all single-assignment variables (ciphertexts) used in $\mathsf{C}^\dagger$ where $v_1, \ldots, v_{n_1}$ are the input variables and $v_{n-n_2+1}, \ldots, v_n$ the output variables. Now we assign a noise level $\ell_i \in \{1, 2, \ldots\}$ to each v_i as follows: the noise levels $\ell_1, \ell_2, \ldots, \ell_{n_1} \in \{1, 2\}$ are already fixed by the input variables $v_1, \ldots, v_{n_1}$. Using the two rules $\ell_{i_3} = \max(\ell_{i_1}, \ell_{i_2})$ when $v_{i_3} = v_{i_1} + v_{i_2}$ and $\ell_{i_3} = \ell_{i_1} + \ell_{i_2}$ when $v_{i_3} = v_{i_1} \times v_{i_2}$, we let noise levels automatically propagate throughout the circuit down to some output levels $\ell_{n-n_2+1}, \ldots, \ell_n$. Note that the noise levels of intermediate and output variables are left totally unbounded during that initial propagation and may therefore exceed by far the maximum level $\ell_{\max} = 2$ supported by the FHE scheme, meaning that the corresponding variables are in fact not decryptable. However, bootstrapping some variable v_i resets ℓ_i to 1 and it is easily seen that bootstrapping all variables $v_1, \ldots, v_n$ makes them all decryptable again: we then say that $\mathsf{C}^\dagger$ is evaluatable. What we are after is a minimal subset $I \subseteq \{1, n\}$ such that bootstrapping v_i for all $i \in I$ has the same effect.

A Boolean Reformulation. To each $v_i \in \mathsf{V}$ is assigned a Boolean value $b_i \in \{\mathsf{True}, \mathsf{False}\}$ that tells whether v_i is to be bootstrapped or not when evaluating $\mathsf{C}^\dagger$. We also define a Boolean mapping $\mathsf{B}(v_i)$ such that

$$\mathsf{B}(v_i) = \mathsf{True} \quad \text{if and only if} \quad \ell_i = 1 \,.$$

We see that if $v_{i_3} = v_{i_1} + v_{i_2}$ then

$$\mathsf{B}(v_{i_3}) = b_{i_3} \vee \big(\mathsf{B}(v_{i_1}) \wedge \mathsf{B}(v_{i_2})\big) \,. \tag{1}$$

[2] XOR and NOT gates correspond to homomorphic additions and AND gates to homomorphic multiplications.

This is because $\ell_{i_3} = 1$ only if $\ell_{i_1} = \ell_{i_2} = 1$ or, as an alternate case, ℓ_{i_3} equals 2 when v_{i_3} is computed but bootstrapping v_{i_3} afterwards resets ℓ_{i_3} to 1. Moreover, if $v_{i_3} = v_{i_1} \times v_{i_2}$ then

$$\mathsf{B}(v_{i_3}) = b_{i_3} \ . \tag{2}$$

Indeed as the result of a multiplication v_{i_3} has level $\ell_{i_3} = 2$. The only way to get $\ell_{i_3} = 1$ is therefore to bootstrap v_{i_3} after computing it. We also see that $\mathsf{B}(v_i)$ is already determined for input variables since for $i = 1, \ldots, n_1$,

$$\mathsf{B}(v_i) = \begin{cases} \mathsf{True} & \text{if } \ell_i = 1, \\ b_i & \text{if } \ell_i \neq 1. \end{cases} \tag{3}$$

Overall, we see that the Boolean predicate B can also be propagated (as a multivariate Boolean expression) across the circuit using the above rules (1)–(3). This operation can be done statically given the description of the circuit and will result in a list of formal Boolean expressions for $\mathsf{B}(v_1), \ldots, \mathsf{B}(v_n)$ that only involve the "bootstrapping" variables $b_1, \ldots, b_n$.

We now capture the fact that $\mathsf{C}^\dagger$ is evaluatable or not as a Boolean predicate ϕ^2_{C}. In order to ascertain the correctness of all variables of $\mathsf{C}^\dagger$, one must just ensure that all variables entering a multiplication have noise level 1. Hence

$$\phi^2_{\mathsf{C}} = \bigwedge_{v_k = v_i \times v_j \in \mathsf{C}^\dagger} \big(\mathsf{B}(v_i) \wedge \mathsf{B}(v_j)\big) \ . \tag{4}$$

Obviously, ϕ^2_{C} is a predicate involving $b_1, \ldots, b_n$ (or a subset thereof) and can be computed once B has been propagated throughout the circuit. All in all, evaluating $\mathsf{C}^\dagger$ with a minimal number of bootstrappings is reformulated as a Boolean satisfiability problem: ϕ^2_{C} must be satisfied with a minimal number of variables $b_1, \ldots, b_n$ set to True.

DNF and Monotone Predicates. We observe that the Boolean predicate $\phi^2_{\mathsf{C}} = \phi^2_{\mathsf{C}}(b_1, \ldots, b_n)$ is monotone since no negated literal $\neg b_i$ appears in ϕ^2_{C}. A monotone predicate is trivially satisfiable by setting all its variables to True. What we want, however, is to satisfy ϕ^2_{C} with as few b_i's set to True as possible. An exact solution to our problem would be to represent ϕ^2_{C} in Disjunctive Normal Form (DNF) *i.e.* as an OR of ANDs. Given a DNF representation of ϕ^2_{C}, it is easy to identify an AND involving a minimal number of variables, thus providing a minimal bootstrapping configuration for $\mathsf{C}^\dagger$. However, noting $\mu(\phi^2_{\mathsf{C}}) \in [1, n]$ this minimal number, even just deciding whether $\mu(\phi^2_{\mathsf{C}}) \leqslant t$ for some $t \in [1, n]$ is a priori intractable:

Theorem 1 ([17], Th. 3.4). *Let ϕ be an n-variate Boolean monotone predicate and $t \in [1, n]$. Let $\mu(\phi)$ be the size of its smallest prime implicant. Deciding whether $\mu(\phi) \leqslant t$ is NP-complete.*

We therefore circumvent this obstacle by adopting a heuristic approach and further validate its effectiveness experimentally as reported later in the paper.

2.2 A Heuristic Solver

We observe that ϕ^2_{C} is computed in Eq. 4 as an accumulated conjunction: thus when propagating B across $\mathsf{C}^\dagger$, we systematically put each $\mathsf{B}(v_i)$ in minimal Conjunctive Normal Form (min-CNF) *i.e.* as an AND of ORs with as few terms as possible. Obviously $\mathsf{B}(v_i)$ becomes more complex (involves more b_i's) as the variable v_i is taken deeper in the circuit. However, the complexity increase remains incremental from $\mathsf{B}(v_{i_1}), \mathsf{B}(v_{i_2})$ to $\mathsf{B}(v_{i_3})$ for $v_{i_3} = v_{i_1}$ op v_{i_2} and computing the min-CNF of $\mathsf{B}(v_{i_3})$ given the min-CNF of $\mathsf{B}(v_{i_1}), \mathsf{B}(v_{i_2})$ therefore requires a moderate computational effort. ϕ^2_{C} is then aggregated along the way as a min-CNF of other min-CNFs, which is easy to program. Once we are done collecting parts and putting together the multivariate predicate ϕ^2_{C}, we apply heuristic transformations on its min-CNF until it becomes small enough to allow a conversion to DNF using a standard algorithm. A minimal bootstrapping configuration is then selected from one of the smallest conjunctive clauses in the resulting DNF.

We apply 3 independent transformations on the min-CNF of ϕ^2_{C}:

1. **Bootstrap required variables:** if $\phi^2_{\mathsf{C}} = (\cdots) \wedge b_i \wedge (\cdots)$ for some b_i then set $b_i = \mathsf{True}$ and repeat the operation until no longer applicable;
2. **Remove redundant variables:** a variable b_i is *redundant* w.r.t. a variable b_j if every occurrence of b_i in a clause of ϕ^2_{C} appears together with an occurrence of b_j (but the converse might not be true). In other words, any clause c containing b_i is of the form $c = (\cdots) \vee b_i \vee (\cdots) \vee b_j \vee (\cdots)$. Setting $b_i = \mathsf{True}$ would of course lead to $c = \mathsf{True}$ but this will only remove all such clauses c from ϕ^2_{C}, whereas setting $b_j = \mathsf{True}$ instead might induce additional simplifications in other clauses of ϕ^2_{C}. Therefore, we set $b_i = \mathsf{False}$, propagate simplifications in the CNF of ϕ^2_{C}, repeat the operation until no longer applicable and restart with Step 1;
3. **Maintain minimal CNF:** Eliminate any clause that is tautologically implied by another clause of ϕ^2_{C}; repeat the operation until no longer applicable and restart with Step 1.

In practice, these transformations are reasonably efficient and allow us to reduce the min-CNF of ϕ^2_{C} in such proportions that converting it to DNF afterwards is either immediate or unnecessary (depending on the circuit C, ϕ^2_{C} sometimes reduces to True by itself along the way, which terminates our algorithm). Therefore, even though our method is unproven, we validated its practical effectiveness. We refer to Sections 4 and 4.2 for experimental results.

Remark 1. Note that one might may also want to ensure that some output variables v_{n-n_2+j} for $j \in J \subseteq [1, n_2]$ have noise level 1 instead of 2. Now, resolving ϕ^2_{C} and bootstrapping these output variables might not yield a minimal solution. To address this case, we simply accumulate the predicates $\mathsf{B}(v_{n-n_2+j})$ for $j \in J$ into ϕ^2_{C} and apply the exact same strategy as above.

3 Extension to FHE Schemes with Many Noise Levels

Assume we are now given a FHE scheme that can handle $\ell_{\max} \geqslant 2$ levels of noise. Let c_1, c_2 and c_3 be ciphertexts with noise levels ℓ_1, ℓ_2 and ℓ_3 respectively. As discussed earlier, there exists essentially two different formulas for ℓ_3 when $c_3 = c_1 \times c_2$:

- $\ell_3 = \ell_1 + \ell_2$: this corresponds to the settings of exponential schemes [13,12,9,10,5,3]. In these schemes, the modulus remains the same after a multiplication but the noise increase depends on the amount of initial noises in the input ciphertexts[3]. At most $\log_2(\ell_{\max})$ layers of homomorphic multiplications can be evaluated before resorting to bootstrapping;
- $\ell_3 = \max(\ell_1, \ell_2)+1$: this corresponds to linear FHE schemes found in [6,4,10]. The noise grows negligibly after a homomorphic multiplication, but the modulus is modified after each multiplication (therefore the relative amount of noise increases). This technique is known as modulus switching, wherein $\ell_{\max}$ different moduli are used to evaluate $\ell_{\max}$ layers of homomorphic multiplications without bootstrapping. Moreover two ciphertexts can only be added or multiplied when they have exactly the same noise level so that their underlying rings become identical. In the following, we assume that the cost of modulus switching for a variable v_i, *i.e.* incrementing its noise level, is negligible compared to the cost of a bootstrapping operation.

We generalize the method of Section 2 to FHE schemes with $\ell_{\max} \geqslant 2$ noise levels: Section 3.1 focuses on a extended algorithm that works with exponential schemes, and we show in Section 3.2 how to slightly modify $\mathsf{C}^\dagger$ in order to reuse the very same algorithm as a black-box to address linear schemes.

We recall that our goal is to minimize the number of bootstrappings needed to homomorphically evaluate the circuit $\mathsf{C}^\dagger$ on input $(v_i, \ell_i)_{1 \leqslant i \leqslant n_1}$. As above, we associate to every circuit variable $v_i \in \mathsf{V}$ a Boolean variable $b_i \in \{\mathsf{True}, \mathsf{False}\}$ that tells whether v_i is to be bootstrapped or not. Again, we construct a Boolean predicate $\phi_{\mathsf{C}}^{\ell_{\max}}$ as a function of $b_1, \ldots, b_n, \ell_1, \ldots, \ell_{n_1}$ that tells whether $\mathsf{C}^\dagger$ is evaluatable. We then rely on our heuristic solver of Section 2 to issue a minimal set $I \subseteq [1, n]$ such that $b_i = \mathsf{True}$ for all $i \in I$ implies $\phi_{\mathsf{C}}^{\ell_{\max}} = \mathsf{True}$.

3.1 Extension to Exponential FHE Schemes

To any variable $v_i \in \mathsf{V}$, we now associate a vector $\mathbf{B}(v_i) = (\mathsf{B}_{i,1}, \ldots, \mathsf{B}_{i,\ell_{\max}-1})$ with $(\ell_{\max} - 1)$ Boolean coefficients such that $\ell_i = j$ if and only if $B_{i,j}$ is the first coefficient set to True as j ranges from 1 to $\ell_{\max} - 1$, and $\ell_i = \ell_{\max}$ if none of the coefficients is True. We make use of the Boolean vector $\mathbf{B}(v_i)$ to encode the noise level ℓ_i of v_i and propagate it throughout the circuit as we did with $\mathsf{B}(v_i)$ in the binary case $\ell_{\max} = 2$. Let us describe in more detail how $\mathbf{B}(v_i)$ evolves when being propagated across the circuit:

[3] Notice that the order of noise increase is quite different between [13,12,9,10,5] and [3], but this does not change our high-level description.

– for $1 \leqslant i \leqslant n_1$ *i.e.* for input variables, set

$$\mathsf{B}_{i,j} = \mathsf{False} \quad \text{for } j \neq \ell_i \qquad \text{and} \qquad \mathsf{B}_{i,\ell_i} = \mathsf{True} \quad \text{if } \ell_i < \ell_{\max} .$$

– when $v_k = v_i + v_j$, set

$$\mathbf{B}(v_k) = \begin{pmatrix} b_k \vee \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,1}\big) \\ \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,2}\big) \vee \big(\mathsf{B}_{j,1} \wedge \mathsf{B}_{i,2}\big) \vee \big(\mathsf{B}_{i,2} \wedge \mathsf{B}_{j,2}\big) \\ \vdots \end{pmatrix} , \tag{5}$$

Indeed, $\ell_k = 1$ if and only if v_k is bootstrapped or $(\ell_i, \ell_j) = (1,1)$, otherwise $\ell_k = 2$ if $(\ell_i, \ell_j) \in \big\{(1,2),(2,1),(2,2)\big\}$, etc. All vector coefficients $\mathsf{B}_{k,3}, \ldots, \mathsf{B}_{k,\ell_{\max}-1}$ are formed in the same fashion.

– when $v_k = v_i \times v_j$, set

$$\mathbf{B}(v_k) = \begin{pmatrix} b_k \\ \mathsf{B}_{i,1} \wedge \mathsf{B}_{j,1} \\ \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,2}\big) \vee \big(\mathsf{B}_{j,1} \wedge \mathsf{B}_{i,2}\big) \\ \vdots \end{pmatrix} . \tag{6}$$

This multiplication expresses the fact that $\ell_k = \ell_i + \ell_j$. Indeed, $\ell_k = 1$ if and only if v_k is bootstrapped, $\ell_k = 2$ if and only if $\ell_i = \ell_j = 1$, and so forth.

Remark 2. Before explaining how to construct the Boolean formula $\phi_{\mathsf{C}}^{\ell_{\max}}$, let us give a couple of remarks on our representation. First of all, this representation does not imply that $\big(\mathsf{B}_{i,j} = \mathsf{True}$ and $\mathsf{B}_{i,m} = \mathsf{False}$ for $m \neq j\big) \iff \ell_i = j$, but that

$$\big(\mathsf{B}_{i,j} = \mathsf{True} \text{ and } \mathsf{B}_{i,m} = \mathsf{False} \text{ for } 1 \leqslant m < j\big) \iff \ell_i = j .$$

This allows us to simplify the formulas for homomorphic addition and multiplication as we do not need to check whether $\mathsf{B}_{i,m} = \mathsf{False}$ for $m > \ell_i$ (see $\mathsf{B}_{k,2}$ in Eq. (5) and $\mathsf{B}_{k,3}$ in Eq. (6)). Secondly, when all the elements of $\mathbf{B}(v_i)$ are False, this means that v_i is at the maximum level of noise $\ell_i = \ell_{\max}$. Therefore this representation nicely generalizes the one of Section 2.

We now construct the Boolean formula $\phi_{\mathsf{C}}^{\ell_{\max}}$ which tells whether the circuit is evaluatable by setting

$$\phi_{\mathsf{C}}^{\ell_{\max}} = \bigwedge_{v_i \in \mathsf{V}} (\ell_i \leqslant \ell_{\max}) = \bigwedge_{v_k = v_i \times v_j \in \mathsf{C}^\dagger} \left(\bigvee_{1 \leqslant m \leqslant \ell_{\max}} \mathsf{B}_{k,m} \right) .$$

Note that the clauses of $\phi_{\mathsf{C}}^{\ell_{\max}}$ encode the fact that to properly evaluate a homomorphic operation $v_k = v_i \ \mathsf{op} \ v_j$, one must just have $\ell_k \leqslant \ell_{\max}$. This is automatically guaranteed by induction for all additions; expressed on all multiplications, this constraint precisely gives the above expression. As before, we use minimal CNF representation to propagate $\mathbf{B}(v_i)$ throughout the circuit and aggregate all the clauses of $\phi_{\mathsf{C}}^{\ell_{\max}}$ on the way. This results in a min-CNF for $\phi_{\mathsf{C}}^{\ell_{\max}}$ to which we apply the same 3 simplifying transformations. We finally convert the resulting predicate to DNF (if necessary) to identify a minimal configuration.

Remark 3. Notice that one might want to ensure that (a subset of) the output variables have noise levels bounded by some $\ell \leqslant \ell_{\max}$. One then aggregates in $\phi_{\mathsf{C}}^{\ell_{\max}}$ the clauses $\bigvee_{i \leqslant \ell} \mathsf{B}_{n-n_2+j,i}$ for $j \in [1, n_2]$ before solving the system.

3.2 Extension to Linear FHE Schemes

In this section, we explain how to deal with the case where $\ell_3 = \max(\ell_1, \ell_2) + 1$ when $c_3 = c_1 \times c_2$. Instead of adapting the previous method, we apply it as a black box to a modified version of the homomorphic circuit $\mathsf{C}^\dagger$. The modified circuit will no longer be consistent with its specification but can be treated by our algorithm regardless. The key idea is to see that one can simulate the linear framework in the exponential framework by replacing every homomorphic multiplication $c_3 = c_1 \times c_2$ with a subcircuit $c_3 = (c_1 + c_2) \times c_{1,2}$ where $c_{1,2}$ is a fixed ciphertext with noise level $\ell_{1,2} = 1$. Indeed, we get

$$\ell_3 = \max(\ell_1, \ell_2) + \ell_{1,2} = \max(\ell_1, \ell_2) + 1,$$

which is the wanted value in linear schemes. As mentioned, the correctness of the modified circuit as a homomorphic version of C is destroyed, but our extended algorithm remains applicable to it and will compute a minimal bootstrapping configuration in an oblivious fashion.

Note however that we need to slightly twitch the extended solver, otherwise solutions might suggest to bootstrap the newly introduced variables $v_{i,j}$. This would not make any sense as these variables have no real existence and only serve as helper variables in our simulation. We can easily circumvent this by not assigning a Boolean $b_{i,j}$ (or equivalently by forcing it to be False in $\mathsf{B}_{i,j}$) to the variables $v_{i,j}$. This eliminates the undesired collateral effect of seeing these variables being bootstrapped when solving $\phi_{\mathsf{C}}^{\ell_{\max}}$. We then successfully compute a minimal bootstrapping configuration from $\phi_{\mathsf{C}}^{\ell_{\max}}$ as previously described.

4 Practical Experiments

In this section, we discuss practical results obtained by applying our algorithms on several circuits (see Table 1). We implemented our basic and extended solvers using Mathematica 9 running on a 2.6 GHz Intel Core i7 with 16 GB of RAM. Although we did not specifically measure execution times, these range from a few seconds to a few hours depending on the circuit size and $\ell_{\max}$ (timings tend to grow exponentially with $\ell_{\max}$). We focused on the benchmarking circuits for MPC/FHE proposed by Smart and Tillich [19], and on circuits put forward by Boyar and Peralta for the AES S-box [2,1]. For each circuit, we computed the minimal number of bootstrappings needed to evaluate homomorphically that circuit with an exponential FHE scheme supporting $\ell_{\max} = 2$ or $\ell_{\max} = 4$ noise levels and with level-1 inputs and outputs *i.e.*

$$\ell_1 = \cdots = \ell_{n_1} = 1 \qquad \text{and} \qquad \ell_{n-n_2+1} = \cdots = \ell_n = 1.$$

Table 1 reports the results we obtained by applying our algorithms to the selected circuits.

Table 1. Minimal number of bootstrappings with level-1 inputs and outputs

Circuit $\mathsf{C}^\dagger$	$\ell_{\max}$	Number of hom. multiplications in $\mathsf{C}^\dagger$	Exact minimal number of bootstappings
Adder 32 bits [19]	2	127	127
Adder 32 bits [19]	4	127	64
Comparator 32 bits [19]	2	150	146
Comparator 32 bits [19]	4	150	74
DES (expanded key) [19]	2	18175	18041
DES (expanded key) [19]	4	18175	8997
AES S-box [2]	2	32	19
AES S-box [2]	4	32	12
AES S-box [1]	2	32	17
AES S-box [1]	4	32	12

4.1 MPC/FHE Benchmark Circuits

Our results show that circuits given as reference by [19] tend to be disappointing when $\ell_{\max} = 2$ as we find that the minimal number of bootstrapping required to evaluate them is nearly equal to the number of homomorphic multiplications, thus being very close to the (trivial) upper bound. This can be explained by the fact that these circuits are automatically generated from hardware components, and clearly not optimized: they were not constructed to be small in terms of gate count, or have a significantly smaller depth, etc. Their linear parts were not optimized either [2]. Also note that setting $\ell_{\max} = 4$ instead of 2 divides the number of required bootstrappings by a factor nearly two.

4.2 The Boyar-Peralta AES S-Box

To the best of our knowledge, the first real-life circuit evaluated by a fully homomorphic encryption scheme is a circuit for AES encryption proposed by Gentry, Halevi and Smart [16]. However the authors decided to completely get rid of the bootstrappings by choosing a FHE scheme with $\ell_{\max} = 100$ so that the entire circuit can be evaluated at once. The drawback of this choice is that the public key becomes *prohibitively large* and required a server with 256GB of RAM to run the implementation and issue performance benchmarks. The authors suggested that bootstrapping might certainly be used as an optimization, *i.e.* as a way to balance the running time and the memory requirements.

The non-linear part of AES, computing the S-box, cannot be performed by table lookups in an homomorphic implementation. We considered circuits for the AES S-box already optimized by Boyar and Peralta with respect to gate count or depth [2,1]. Our practical results are detailed on Table 1. Contrarily to the circuits of [19], the Boyar-Peralta circuits were optimized and we found that their minimal number of bootstrappings is nearly half the number of homomorphic multiplications when $\ell_{\max} = 2$. As a result, homomorphically evaluating an AES encryption with a 2-level FHE scheme can be boosted by a factor 1.88 by just

choosing the circuit from [1] and use our 17-bootstrapping optimal configuration $\{t_{21}, t_{22}, t_{23}, t_{24}, t_{26}, t_{29}, t_{33}, t_{36}, t_{40}, s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ as described in [8].

However, when $\ell_{\max}$ grows, the gain of minimal bootstrapping operations with respect to the case $\ell_{\max} = 2$ is smaller than for the circuits of [19] (and even lower bounded by 8) due to the structure of these circuits[4]. Since the output variables are required to have a minimal noise level, the last reduction phase implies that the minimal solution consists in bootstrapping these output variables.

5 Conclusion

We introduced a method that computes the exact minimal number of bootstrappings required to homomorphically evaluate any circuit using any known FHE scheme. When $\ell_{\max} = 2$, the number of homomorphic multiplications is a strict upper bound on the minimal number of bootstrappings but significantly better figures can be found using our approach as exemplified by the circuit from [1]. We see, however, that most commonly used circuits are disappointingly unoptimized with respect to their "bootstrapping complexity". As an avenue for future research, we suggest to explore algorithmic strategies to *build* bootstrapping-efficient circuits *i.e.* to decrease their boostrapping complexity by a specific design effort. Finally, it would be interesting to refine our definition of noise levels to take into account the additional logarithmic effects induced by homomorphic operations, especially in the case of linear FHE schemes.

References

1. Boyar, J., Matthews, P., Peralta, R.: Logic minimization techniques with applications to cryptology. Journal of Cryptology 26(2), 280–312 (2013)
2. Boyar, J., Peralta, R.: A depth-16 circuit for the AES s-box. Cryptology ePrint Archive, Report 2011/332 (2011), `http://eprint.iacr.org/`
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, pp. 309–325. ACM (2012)
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 97–106. IEEE Computer Society (2011)
6. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
7. Brenner, M., Perl, H., Smith, M.: Implementation of the fully homomorphic encryption schemes of Gentry and Smart and Vercauteren, `https://hcrypt.com/`

[4] Notice that these circuits are composed of three phases: top linear transformations, shared non-linear component, and bottom linear transformations.

8. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
9. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)
10. Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012)
11. Coron, J.-S., Tibouchi, M.: Implementation of the fully homomorphic encryption scheme over the integers with compressed public keys in sage, `https://github.com/coron/fhe`
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
13. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), `http://crypto.stanford.edu/craig`
14. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
15. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
16. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
17. Goldsmith, J., Hagen, M., Mundhenk, M.: Complexity of DNF and isomorphism of monotone formulas. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 410–421. Springer, Heidelberg (2005)
18. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, pp. 1219–1234. ACM (2012)
19. Smart, N.P., Tillich, S.: Circuits of basic functions suitable for MPC and FHE, `http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/`
20. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)