Conditionals in Homomorphic Encryption and Machine Learning Applications

Diego Valerio Chialva and Ann Dooms

Abstract

Homomorphic encryption has the purpose to allow computations on encrypted data, without the need for decryption other than that of the final result. This could provide an elegant solution to the problem of privacy preservation in data-based applications, such as those provided and/or facilitated by machine learning techniques, but several limitations and open issues hamper the fulfillment of this plan. In this work we assess the possibility for homomorphic encryption to fully implement its program without the need to rely on other techniques, such as multiparty computation, which may be impossible in many actual use cases (for instance due to the high level of communication required). We proceed in two steps: i) on the basis of the well-known structured program theorem [28] we identify the relevant minimal set of operations homomorphic encryption must be able to perform to implement any algorithm; and ii) we analyse the possibility to solve -and propose an implementation for- the most fundamentally relevant issue as it emerges from our analysis, that is, the implementation of conditionals (which in turn require comparison and selection/jump operations) in full homomorphic encryption. We show how this issue has a serious impact and clashes with the fundamental requirements of homomorphic encryption. This could represent a drawback for its use as a *complete* solution in data analysis applications, in particular machine learning. It will thus possibly require a deep re-thinking of the homomorphic encryption program for privacy preservation.

We note that our approach to comparisons is novel, and for the first time completely embedded in homomorphic encryption, differently from what proposed in previous studies (and beyond that, we supplement it with the necessary selection/jump operation). A number of studies have indeed dealt with comparisons, but have not managed to perform them in pure homomorphic encryption. Typically their comparison protocols do not utilise homomorphic encryption for the comparison itself, but rely on other cryptographic techniques, such as secure multiparty computation, which *a)* require a high level of communication between parties (each single comparison in a machine learning training and prediction process must be performed by exchanging several messages), which may not be possible in various use cases, and *b)* required the data owner to decrypt intermediate results, extract significant bits for the comparison, re-encrypt and send the result back to the other party for the accomplishment of the algorithm. Such "decryption" in the middle foils the purpose of homomorphic encryption. Beside requiring only homomorphic encryption, and not any intermediate decryption, our protocol is also provably safe (as it shares the same safety as the homomorphic encryption schemes), differently from other techniques such as OPE/ORE and variations, which have been proved not secure.

Index Terms

homomorphic encryption, machine learning

I. Introduction

Machine learning, data mining, and predictive data analytics represent an ensemble of techniques and algorithms (which for simplicity we will in the following indicate simply as "machine learning") that allow systems to act and make predictions without being explicitly programmed in full detail to do so, but by leveraging their input data and inference techniques. They have nowadays an overwhelming number of practical applications providing us with an unprecedented level of comfort and services, from tailored suggestion systems, to "personalized medicine", and several other services.

However, these advantages typically come at the price of loosing individual privacy, as personal or valuable information is used by the algorithms and the third parties operating them. This issue has spawn research activity at different levels. Very roughly speaking we can divide the developed privacy-preservation techniques in two classes: those that work by modifying the data themselves, and those that modify the representation of the data but not the actual data content.

Techniques of the first class act on the datasets holding the privacy-concerned data and can be divided in a few subclasses [1]. Common to all of them is the distinction between identifier, quasi-identifier and anonymous data. Such techniques require only comparatively minor (conceptual) changes to the applications algorithms acting on the data, but they have significant drawbacks. Indeed, they impose a trade-off between the degree of preserved privacy and the usefulness of the data: a "privacy budget", which has been shown to be quite limited [2]. Moreover, such techniques appear to be beatable by the algorithms themselves and database crossing attacks (that is, the use by an attacker of other, public or stolen, databases to "complete" or infer the relevant distorted information in the database of interest) [2].

The other class of techniques has been proposed within cryptography. Among the different research lines we recall: secure multi-party computation, homomorphic encryption, functional encryption and program obfuscation, see for instance [3]–[6]. These approaches differ on several aspects, including the set of functions that can be computed on the encrypted data and stage

D. Chialva (diego-valerio.chialva AT ec.europa.eu) is with the ERCEA (European Research Council Executive Agency), and A. Dooms (ann.dooms AT vub.be) is with the Department of Mathematics, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium.

Disclaimer. The views expressed in this paper by Diego Chialva are the author's. They do not necessarily reflect the views or official positions of the European Commission, the European Research Council Executive Agency or the ERC Scientific Council.

of development. Homomorphic encryption has appeared in particular interesting for privacy preservation (including algorithm protection) in learning applications, and has been actively pursued in the latest years, e.g [7]–[12].

Homomorphic encryption aims at enabling the computation of arbitrary (in the case of *fully homomorphic encryption*) or classes (in the case of *partial homomorphic encryption*) of functions on encrypted data without having the need to decrypt them first and *limiting decryption to the very final result only*. Several open issues make homomorphic cryptosystems still unsuited for the vast majority of machine learning algorithms. Those that have been identified in the literature mainly are: **memory footprint**, **computational complexity**, **limited representable data** (only integers and finite precision floats) and a **restricted set of operations** (only polynomial operations: addition and multiplication).

Such problems can however be divided in two classes. The first class comprises issues such as the memory footprint and the computational complexity, which could be hoped to be trivially solved by technological (hardware) advancements, similarly to what has happened in deep learning. On the other hand, the second class of the above-listed problems, like the limited types of representable data and the lack, for instance, of comparison operations, must find a solution at the theoretical and cryptography level. It is this second class of issues that we are interested in.

In this work we will reanalyse and investigate the minimum set of operations necessary to implement any algorithm. This is a fundamental question and a necessary step: in order to assess the possibility for homomorphic encryption to accommodate all algorithms (in particular machine learning ones) whatever their complexity, one identifies a minimum number of basic operations and studies their implementations. Proving the impossibility of implementing such operations, would entail the impossibility to implement the more complex algorithms within homomorphic encryption[1].

We proceed on the basis of the well-known structured program theorem [28], we identify the minimal basic operations of interest to model any machine learning algorithm, and then discuss those that are not yet achievable in homomorphic encryption, which turn out to be comparisons and "jump/selections" to a subset of the compared elements. In fact, it is the combination of the two that allow the evaluation of conditionals. We will study both classes of operations.

Concerning comparisons, our approach is novel and for the first time completely embedded in homomorphic encryption. Several proposal, indeed, have been made concerning comparison operations in encrypted setting, but they did not succeeded in being performed in full-fledged homomorphic encryption. Typical, well-studied approaches have been

1) the so-called Order-Preserving-Encryption and its variants such as Modular-OPE and Order-Revealing-Encryption, see for instance [29]–[34]), which do not belong to the homomorphic encryption class and, more seriously, have been proven to be not secure (for recent proofs, see for instance [35], [53]);
2) secure-multiparty computation (for recent works see [55], [56]), which, although secure, require a high level of communication between parties (each single comparison in a machine learning training and prediction process must be performed by exchanging several messages), which may not be always possible;
3) combinations of homomorphic encryption with other cryptographic techniques such as secure multiparty computation used to perform the comparisons [38], [54], [57], [58]. These approaches however did not manage to perform the comparison exclusively on encrypted messages, as the data owner was required by the protocol to decrypt intermediate results, extract the significant bits for the comparison, re-encrypt and send the result back to the other party for the accomplishment of the algorithm. Such "decryption" in the middle frustrates the purpose of homomorphic encryption (also, the need for high level of communication between parties due to the use of secure multiparty computation may be impossible in a number of actual practical use cases).

While we manage to individuate a technique for achieving comparisons in purely homomorphic encryption (that is, with no need for communication between parties and acting exclusively on encrypted messages with no need for intermediate nor partial decryption), we will show that when trying to implement full conditionals, which are the key in practical applications, one hits a rather fundamental issue in homomorphic cryptosystems, namely the cryptographic requirement of semantic security. This could represent a serious drawback in using homomorphic encryption for data analysis applications and for implementing algorithms in general, and could force to revisit (or abandon) that plan in its more ambitious formulation.

The article is organized as follows: we provide a brief introduction to homomorphic encryption in Section II, after which we study conditionals and comparisons in a homomorphic setting in Section . In section IV we present our methodology and test results. We finally discuss applications to machine learning in Sections V, V-A, where we highlight and provide precise and exhaustive examples of how the fundamental, general issues of homomorphic encryption that our analysis has revealed impact on the program of implementing machine learning algorithms in such scenario. In our conclusions we also briefly touch upon the consequences of our work in applications different than machine learning.

II. Homomorphic encryption

A cryptosystem consists of a plaintext P, ciphertext C and key space K together with an encryption function Encr and decryption function Decr. Although in the literature Encr is called encryption *function*, it is not exactly a function in the strict mathematical sense for most of the encryption schemes, because an element of (pseudo)randomness is involved such that

[1]Clearly, in the opposite case where the fundamental analysis is positive, one still need to assess the effectiveness of the implementation, which may still condemn homomorphic encryption to be impractical.

applying it more than one time to the same key and plaintext, one obtains different ciphertexts. Such probabilistic encryption schemes are favoured because they provide *semantic security*[2], which is equivalent to *ciphertext indistinguishability*[3] [16], [17]. This needed randomness has a huge relevance in homomorphic encryption, as we will see.

Encryption schemes are further distinguished by the relation between the encryption and decryption key. If the decryption key can be easily computed from the encryption one (in the typical case they are in fact identical), one speaks of a *symmetric* cryptosystem, while if not, one speaks of an *asymmetric* cryptosystem. Typical asymmetric systems also distinguish between public (for encryption) and private (for decryption) keys k_p, k_s.

In modern cryptology the adversaries are conceived as having finite computational resources and a cryptosystem is considered secure if its breaking is unfeasible with attack algorithms that are probabilistic in nature and running in polynomial time. The running of the cryptosystems functions and adversary algorithms are all measured as a function of the so-called *security parameter* λ, which measures the complexity of the computational problem.

A cryptosystem is *homomorphic* for an operation $*$ acting on P if there is a corresponding operation $\circ$ acting on C with

$$\text{Decr}(k_s, \text{Encr}(k_p, m_1) \circ \text{Encr}(k_p, m_2)) = m_1 * m_2 \tag{1}$$

for $m_1, m_2 \in P$.

Note that this is not in general a true group homomorphism, because

$$\text{Encr}(k_p, m_1) \circ \text{Encr}(k_p, m_2)) \neq \text{Encr}(k_p, m_1 * m_2). \tag{2}$$

due to the (pseudo)randomness of the encryption scheme. However, while mathematically this lack of identity holds, there is a strong definition of homomorphic cryptosystems that reconciles with the group-homomorphism-like identity, in the statistical or computational senses, see [18].

Defining a *homomorphic encryption system* $(P, C, K, \text{Encr}, \text{Decr}, \text{Ev})$ then consists in specifying the evaluation function Ev that performs the homomorphically preserved operation on the ciphertexts:

$$\text{Ev} : C^n \times \mathcal{C} \times K \to C : (\vec{c}, O, p_k) \to c' \tag{3}$$

and $\mathcal{C}$ is some family of circuits that the homomorphic cryptosystem can evaluate. An homomorphic cryptosytem is defined *correct* if it correctly decrypts ciphertexts both coming from a circuit evaluation (sometimes called "evaluated ciphertexts"), and from direct encryption of a plaintext (also dubbed "fresh ciphertexts"). Trivial homomorphic cryptosystems are excluded by requiring *strong homomorphicity* and *compactness* for the cryptosystems, see [18].

Homomorphic cryptosytem can be further distinguished in

- *partial homomorphic*: allow only one type of operation for an unlimited number of times,
- *somewhat homomorphic*: allow summing and multiplication but only for a limited number of times (the size of the ciphertext depends on circuit depth),
- *leveled homomorphic*: allow summing and multiplication but only for a limited number of times specified as an input parameter (here the size of the ciphertext does not depend on the maximal allowed circuit depth, but the size of the public key does),
- *fully homomorphic* allow summing and multiplication for an unlimited number of times, and thus arbitrary functions expressible as arithmetic circuits composed of summing and multiplication.

The evaluation of each operation in a circuit increases the noise component (given by the randomness necessary for semantic secure schemes) of the ciphertexts. When the noise is above a certain limit decryption is no longer possible. Fully homomorphic systems, as constructed first by Gentry, see [14], can cope with this issue thanks to a procedure, called *bootstrapping* that allows to extend specific somewhat homomorphic cryptosystems (called *bootstrappable*) to systems where unlimited number of operation evaluations are possible. Later realisations are, for example, [19]–[27]. For the purpose of this work, it is important to note that the noise increase in addition and multiplication is different. Typically the one incurred by with multiplications is much larger.

Moreover, and quite relevantly, in practical applications the computational complexity (and hence slowness) of homomorphic operations has prompted to use *leveled* homomorphic systems. This clearly affects the algorithms that can be successfully implemented at the practical level.

III. Conditionals and Comparison Operations in Homomorphic Cryptosystems and Minimal Necessary Operations in machine Learning

As mentioned in the introduction, we begin by identifying the minimal necessary building blocks to cope in general with the implementation of algorithms (or computable functions). The theorem of structured programming [28] shows that any

[2]The fact that no polynomial time probabilistic algorithm can derive information about a plaintext m given its length, ciphertext and encryption algorithm, more than any other polynomial time probabilistic algorithm that has not access to the ciphertext.

[3]Given two plaintext chosen by the adversary, and the ciphertext of one of them chosen by us, the adversary cannot distinguish which of the plaintexts has been encrypted with a probability (significantly) larger than $1/2$, see [15].

algorithm representable as a flow chart (and, thus, machine learning algorithms) can compute any computable function by combining a series of minimal necessary operations: 1) sequencing subprograms, 2) iterating a subprogram conditioned on the value of a boolean expression, 3) selecting one or another subprogram based on a boolean expression.

Homomorphic encryption aims at computing any computable function on encrypted data *without recurring to intermediate, not even partial, decryption*, and it has been highly regarded as a possible to make privacy-safe machine learning algorithms. So far it has been proven that homomorphic encryption alone[4] allows only to compute general polynomial operations (modelable with circuits of addition and multiplication), but in order to prove that homomorphic encryption can be applied at least conceptually to machine learning algorithms one still need to prove, on the basis of the structured programming theorem, that it can provide for comparison operations, as well as selections (the two forming what are called *conditionals*).

We will show that some of the involved aspects pose rather crucial problems for the program of homomorphic encryption. We will propose a solution to implement comparisons. We will also have to deal with other open issues in homomorphic encryption, such as the ability to perform divisions among ciphertexts.

A. State of the Art

Comparison and order relations among ciphertexts in an encryption system have been a long-sought objective given the several practical use cases. Limiting ourselves to the most recent works, in the last fifteen years the field of "order-preserving encryption" (OPE) has been revived, starting from an article by Agrawal in 2004 [29], and with a rapid development, see [30]–[33]. Unfortunately all variations on the techniques (from OPE, to MOPE, and so on) have failed with respect to security, as they would leak much of the original plaintext, from half to the totality of it, see for example [53]. Recently a new approach within this paradigm has been proposed, dubbed "order revealing encryption" (ORE) [34], which tries to abandon structuring the ciphertext space and rather focuses on developing a (pseudo)function returning an order judgement between couples of ciphertexts. However, also the security of this technique has been shown to be flawed, see for recent works [35], [53]. In conclusion, on the basis of the studies in the literature, OPE/ORE and their variants do not allow to implement secure comparison operations on encrypted data.

Moving to the domain of homomorphic encryption there is a claim that comparison would not be feasible in pure homomorphic encryption, see for example the comments in the reviews such as [36]. In fact, in the literature there are, to our knowledge, no proposals for comparison within homomorphic encryption alone: all proposed techniques require a combination of techniques (typically of homomorphic encryption and secure multiparty computation) with the necessity to decrypt the homomorphically encrypted messages (or at least some significant bits) in order to accomplish the comparison. For example, the reader can consider [37], [38], [54], [57], [58], whose protocols require the data owner to decrypt intermediate results, extract the significant bits for the comparison, re-encrypt and send the result back to the other party for the accomplishment of the algorithm. Such "decryption" in the middle foils the purpose of homomorphic encryption, which aimed at avoiding intermediate, even partial, decryption and allowing entirely running an algorithm by a third party.

Moreover, secure multiparty computation requires a high level of communication between parties, even in advanced protocols for comparisons, which may not be always possible (in fact, each single comparison in a machine learning training and prediction process must be performed by exchanging several messages, see for recent works [55], [56]).

We will therefore follow a different route.

B. Elements of Conditionals: Comparison and Jump

As mentioned in section III, an analysis of the minimal requirements for implementing flow-chart representable (essentially all) algorithms points to the necessity to evaluate *conditionals*. We can distinguish two steps in conditionals. The first one consists of weighting different options or paths and finding their ranking or order (we are typically looking for evaluating a major/minor/equal relation): this is the *comparison step*. The second one consists of selecting that option: this is the *selection/jump step*. This second step will turn out to be highly sensitive, and, as we will discuss in Section III-D, will be in strong tension with (homomorphic) encryption. We begin by implementing comparisons in purely homomorphic encryption.

C. Implementing Comparison Operations in Homomorphic Cryptosystems

Our idea is as follows: we avoid trying to implement comparison operations as basic/elementary features of our homomorphic system (as attempted in previous research), but rather find convenient ways to model them as circuits that a homomorphic system can correctly and efficiently evaluate.

We start by the definition of comparisons as a map

$$\texttt{Comp} : C \times C \to \mathcal{S} = \{0, \pm 1\} \tag{4}$$

with C in our case the ciphertext space. We tackle the problem by trying to find a representation of this map in terms of elements of the circuit family that the homomorphic system can evaluate. However, `Comp` is **not** straightforwardly representable in terms of the circuit family that can be evaluated by the existing homomorphic systems, which allow sums and multiplications (that is,

[4]We will discuss in section III-A the advantages and disadvantages in combining homomorphic encryption with other crytography techniques, as it has been done in the literature, explaining also why this combination may not be possible in several concrete use cases and situations.

polynomial operations). In fact Comp is discontinuous, and indeed typically implemented as a sign or Heaviside function, by mapping the sign positive or negative of the difference of its argument to 0 or respectively ± 1. Being discontinuous, the (Stone)-Weierstrass theorem concerning polynomial approximations does not apply, and insisting on such an approximation requires using many polynomials of high degree, while the approximations are still of bad quality because of Gibb's phenomenon. High polynomial order means high number of consecutive multiplications in the homomorphic system, which is problematic for the levelled or somewhat homomorphic schemes to which one is limited in practice as we have explained before.

We can however cope with these relevant issues and obtain a satisfactory definition and modelling.

Solution to the problem. We propose here our solution, allowing: 1) to use only polynomial operations, 2) to compute comparison in an efficient way in pure homomorphic encryption.

As we have remarked, Comp is typically implemented, as a sign or Heaviside function. We note that these are distributions, also called generalized function[5], and this allows us to base our solution on the representation of distributions as the *weak limit* of *sequences of locally integrable functions*. This has the advantage that we can select suitable locally integrable functions admitting more convenient polynomial approximations that are amenable to homomorphic encryption.

Performing the weak limit is on the other hand problematic in the homomorphic encryption setting and in general when using (polynomial) approximations, which are typically defined only over restricted intervals. We will solve this problem by selecting a class of locally integrable functions that have specific and suitable characteristics enabling to calculate such limit in a sufficiently accurate way by mapping the values calculated over the restricted interval(s) to values at points outside such interval(s). A key-point will be keeping the number of consecutive operations sufficiently small (thus also keeping the necessary polynomials to be of a low degree).

After this general introduction to our solution, we now pass to its concrete illustration, **in three key points** hereby indicated as 1, 2 and 3.

1) Choice of the sequence of locally integrable functions.

As we said, the comparison function (4) is a *sign* or equivalently Heaviside (generalized) function $H(x)$. It can be obtained as the weak limit of several sequences of locally integrable functions, but in order to be able to effective perform the weak limit in homomorphic encryption with lower polynomials, we will use the sequence

$$\{\tanh(kx)\} \tag{5}$$

where $\tanh(x)$ is the hyperbolic tangent, such that the weak limit becomes

$$H(x) = \lim_{k\to\infty} \frac{1}{2}(1 + \tanh(kx)). \tag{6}$$

Using the sequence of hyperbolic tangent functions will be crucial to allow us to effectively compute the weak limit in homomorphic encryption, as we will now explain. First of all, note that homomorphic encryption only allows polynomial operations, hence we need to polynomially approximate the functions $\tanh(z)$. Importantly, an approximation is necessarily only valid (that is, accurate) over a restricted interval. In fact for performance reasons in our case, the interval will be $[0, 0.25)$ in order to use the lowest possible order in polynomial approximations of the functions, as we will explain in point ***3)***. However, computing the weak limit means calculating the function over large intervals ($z = kx, k \gg 1$). We will manage to do so precisely because of the so-called bisection property of $\tanh(z)$, which will permit us to map values calculated over the restricted interval to values at points outside such interval and obtain the weak limit rather efficiently (only low order polynomials will be necessary).

2) Definition and calculation of the weak limit.

As said, the weak limit in equation (6) requires mapping the (approximate) calculated value of $\tanh(z)$ for $|z| \in [0, 0.25)$ to much larger z: effectively $z = kx, k \gg 1$ (as $k \to \infty$). In order to do so, we employ the bisection equation:

$$\tanh(2z) = \frac{2\tanh(z)}{1 + \tanh^2(z)}. \tag{7}$$

After r applications of the bisection formula the hyperbolic tangent initially calculated at $z = x$ is now calculated at $z = 2^r x$. Thus $k = 2^r$ and the limit $k \gg 1$ corresponds to $r \gg 1$. We will discuss later on what values of r are achievable and/or efficient in practice, and what effect this has on the accuracy of the final comparison result.

The issue of divisions. Note that equation (7) involves calculating a division, which is not possible in the present homomorphic encryption schemes. We solve this issue by using specific polynomial approximations for the function $\frac{1}{x}$, where x can then be generalized to functions of our ciphertexts, once we understand how to approximate the reciprocal function for a variable x.

Obviously the smaller the degree of the polynomial, the less accurate is the approximation, but, as said, we have to consider small degree polynomials because of the limitations of the leveled homomorphic cryptosystems we have to deal with in practice.

[5]For an introduction see [39].

We have considered the two following minimax[6] approximations (which can be found for example using the Matlab minimax algorithm [40]):

$$\frac{1}{x} \approx 2.871320 - 3.029870x + 1.392785x^2 - 0.235498x^3 \tag{8}$$

$$\frac{1}{x} \approx 1.4571 - 0.5x \tag{9}$$

both for $x \in [1, 2]$. The first polynomial provides a better approximation (accuracy[7] $\mu = 9.62$ bits, compared to 4.5 bits of the second one), but its degree is three times bigger, which will finally make it a less convenient candidate for homomorphic cryptosystems.

Coming back to equation (7), we note that for all z, $0 \leq \tanh^2(z) \leq 1$, hence we must polynomially approximate the function $\frac{1}{1+x}$ for $x \in [0, 1]$ with $x = \tanh^2(z)$. This is easily done by observing that we can obtain the approximation of the function $\frac{1}{1+x}$ for $x \in [0, 1]$ by shifting $x \to x+1$ in the approximation (9) of the reciprocal function $\frac{1}{x}$ for $x \in [1, 2]$. We thus get

$$\frac{1}{1+x} \approx 0.9571 - 0.5x \qquad x \in [0, 1]. \tag{10}$$

and we can hence write the approximate bisection formula as

$$\tanh(2z) \sim \tanh(z)(1.9142 - \tanh(z)^2). \tag{11}$$

3) Polynomial approximation of the locally integrable functions.

We finally address the polynomial approximation of $\tanh(z)$ itself. Our choice for the explicit polynomial must also be guided by the fact that we are constrained in practice by the maximum number of consecutive operations that the leveled homomorphic system we are limited to can sustain before the need to bootstrap. Luckily, $\tanh(z) \sim z$ for $|z| \in [0, 0.25)$ with already quite good accuracy (≥ 7.6 bits).

In practical applications with concrete datasets, this implies that datapoints must be preprocessed and in particular *normalized* such that the values we want to compare fall in the interval $\sim [-0.12, 0.12]$ in order to apply the algorithms with the above described approximation.

D. Implementing Jump (Selection) operations in Homomorphic Cryptosystems

As we discussed in Section III homomorphic encryption will be able on first principles to compute any computable function or algorithm when it will be able to implement comparisons *as well as* selections/jumps to a given option/path once the result of the comparison is obtained.

The selection operation is *particularly difficult in an homomorphic setting*, and this is a crucial realisation of our analysis. Indeed cryptography requires *semantic security*, which is equivalent to *ciphertext indistinguishability*. The indistinguishability requirement is evidently in tension with the necessity to select a path (one or more ciphertexts) at run time, that is, before the decryption, which is supposed to occur only at the end.

We propose as best operational solution to this issue an *"implicit selection" by weighting*. This is in fact not an actual selection, as we will explain, so that it fully respects semantic security. The idea is not to truly select, but to map the two subsets (the one of elements we want to select, and the one of elements not to select) into two different subspaces, choosing those spaces in a way that this map will keep them separate in the subsequent parts of any algorithm and will allow to recover at the end the selected-for part. This is achieved by collapsing all elements of the "not-to-select" subset into the zero element of the ciphertext space, while the elements that we want to select will be preserved without change (that is, they will be mapped in themselves). We recall that in the case of homomorphic cryptosystems defined on polynomial rings the zero element is the zero polynomial.

The mapping procedure consists in re-scaling the compared data ciphertexts x_1, x_2 with suitable weights that depend on the result of the comparison. There are different ways to implement such "selection" weights, differing in what comparison operation one wants to implement ($>$, $<$, ...) and what are the constraints on the number of consecutive operations. We present in the next section a number of possible implementation of such maps and full-fledged conditionals.

[6]A minimax polynomial is a polynomial that approximates uniformly a function minimizing the maximum error (typically the maximum absolute difference between the function and the approximation over the finite interval of interest).

[7]The accuracy μ is related to the error ϵ as $\mu = -\log_2 \epsilon$.

E. Implementing full conditionals (comparison + selection) in Homomorphic Cryptosystems

We present implementations of a series of *comparison* and *selection* functions, each of which realized as an algorithm:

$$\texttt{Comp} : C \times C \to \{0, \pm 1\}, (x_1, x_2) \to w_{12} \tag{12}$$

$$\texttt{Select}_{>\frac{1}{2}} : C \times C \to C \times C, (x_1, x_2) \to (s_{12}x_1, s_{21}x_2) \tag{13}$$

$$\texttt{Select}_{<\frac{1}{2}} : C \times C \to C \times C, (x_1, x_2) \to (s_{12}x_1, s_{21}x_2) \tag{14}$$

$$\texttt{Select}_{=} : C \times C \to C \times C, (x_1, x_2) \to (\overline{\overline{s}}x_1, \overline{\overline{s}}x_2) \tag{15}$$

$$\texttt{Select}_{>} : C \times C \to C \times C, (x_1, x_2) \to (\widetilde{s}_{12}x_1, \widetilde{s}_{21}x_2) \tag{16}$$

$$\texttt{Select}_{<} : C \times C \to C \times C, (x_1, x_2) \to (\widetilde{s}_{12}x_1, \widetilde{s}_{21}x_2) \tag{17}$$

with

$$s_{ij} = \frac{1 + w_{ij}}{2}, \quad \overline{\overline{s}} = 1 + w_{12}w_{21}, \quad \widetilde{s}_{ij} = w_{ij}\frac{1 + w_{ij}}{2}. \tag{18}$$

Note that, although $w_{21} = -w_{12}$, it is more convenient in the homomorphic cryptosystem scenario to calculate w_{12} and w_{21} independently, so that they have the same (lower) noise content, rather than w_{21} having higher one due to being the negation of w_{12}. This will improve accuracy and precision of the algorithms allowing more operations on the ciphertexts, but at the expense of time efficiency.

The $\texttt{Select}_{>/<\frac{1}{2}}$ and the $\texttt{Select}_{>/<}$ algorithms differ in how they map the case $x_1 = x_2$: the former map $x_{1,2} \to 0.5x_{1,2}$, the latter map $x_{1,2} \to 0$. Note that although the former algorithms do not implement exactly the $>$ and $<$ relations, they are convenient because they use less operations, and for some practical applications their treatment of the case $x_1 = x_2$ is not very problematic.

Finally, in Algorithm 1 we provide a detailed description using $\texttt{Select}_{>\frac{1}{2}}$ as an example (the algorithms for $\texttt{Select}_{>/<\frac{1}{2}}$ and the $\texttt{Select}_{>/<}$ can be easily derived herefrom).

Algorithm 1 Algorithm $\texttt{Select}_{>\frac{1}{2}}$, encrypted version.

Input: Integer r and encrypted $z_c = x_{c1} - x_{c2}$, where x_{c1}, x_{c2} are encryptions of $x_1, x_2 \in [-0.12, 0.12]$ encoded using fractional encoder
Constants coefficient list $[-1.9142, 1.0, 0.5]$
Output: Binary values $\{0, 1\}$ with accuracy of about 3.65 bits
Algorithm
for $b \in$ coefficient list **do do**
$b_e \leftarrow \texttt{Enc}_f(b)$
end for
for $i = 0$ to r **do do**
Compute: $y_c \leftarrow z_c * z_c$
Add plain: $u_c \leftarrow -1.9142_e + y_c$
Multiply: $t_c \leftarrow z_c * u_c$
Assign: $z_c \leftarrow t_c$
end for
if $r\%2 == 1$ **then**
Negate: $z_c \leftarrow -t_c$
Add plain: $z_c \leftarrow z_c + 1.0_e$
Multply: $z_c \leftarrow z_c * 0.5_e$
else
Add plain: $z_c \leftarrow z_c + 1.0_e$
Multply: $z_c \leftarrow z_c * 0.5_e$
end ifreturn z_c
Data owner $\to$ decrypt z_c into z_e
Data owner $\to$ decode z_e into z_f
Data owner calculates the final result $(z_f + 1) * 0.5$.

We end this section with some comments on the specific features of the mechanism we have proposed to implement the selection/jump operations. First of all we stress the main difference with an actual selection/jump: while the latter operating on a certain set of element returns in general a subset of it (typically, but of course not always, with fewer elements), our proposed mechanisms projects the unwanted elements onto the zero element, while preserving the elements one wants to select.

However, both the w_{ij}'s and the elements will be encrypted in the homomorphic case, thus we will not be able to discern which elements have been mapped to the zero element and which have been preserved (selected). Therefore, one will have to carry over all elements until the moment of decryption. This clearly has implications for the efficiency of practical applications with large datasets. We explore some of the consequences of this in Section V-A.

An important figure of merit for the functions we have defined is the maximum number of consecutive operations they require, because the efficiency required in practical applications forces us to avoid bootstrapping, and thus allows only a limited number of consecutive operations. We will discuss this in detail in Section IV-C2.

IV. Tests and Results

A. Methodology

The general results presented in this work are *agnostic* for what concerns the choice of (fully) homomorphic cryptosystem. Nevertheless, to concretely implement our models and algorithms, we have chosen to adopt the scheme of Fan and Vercauteren (FV) [25] for a series of reasons.

a) ***Efficiency****:* the FV scheme is an efficient implementation of the scheme in [41], one of the most remarkable second generation homomorphic systems.

b) ***Comprehension of the operating range for the cryptosystem parameters****:* determining the correct operating range of parameters for the various homomorphic cryptosystems is one of the active topics of research and it is unclear in all schemes. Other cryptosystems beside the Fan-Vercauteren one have been studied under this point of view, but their good parameter ranges are much less clear than the already incomplete one in Fan-Vercauteren's, as one can for example see mentioned and discussed in [42], see also [43] when speaking of the popular scheme of [24]. The FV scheme has been subject to a few more studies and experiments, as for example can be seen in the documentation of libraries such as SEAL [44], and [8].

c) ***State of software libraries****:* this is the point where the FV scheme is particularly valuable, with examples such as [44]–[46]. In particular, `SEAL` [44] is evolving towards more explicit software engineering standards. We have been using its version 2.1, as latest updates have implemented modifications in the FV homomorphic schema to improve the speed of calculations, but making it less simple to explore suitable ranges of parameters, see [44].

One important remark is that all libraries we know of do not actually implement the *fully* homomorphic cryptosystem, because they do not implement the bootstrapping, thus reducing the cryptosystem to only its somewhat homomorphic version. This will effectively limit the maximum number of consecutive operations we can evaluate.

As our work is agnostic concerning homomorphic schemes, knowing the details of the FV one is not essential. It is however relevant to have a picture of the scheme's parameters, as they affect, for instance, the size of encodable data, the size of evaluable circuits, and so on. They are:

- the plaintext modulus t, for the plaintext space $R_t \equiv \frac{\mathbb{Z}_t[x]}{f(x)}$
- the ciphertext modulus $q \ll t$, for the ciphertext space $R_q \equiv \frac{\mathbb{Z}_q[x]}{f(x)}$
- the degree $d = 2n$ of the monic irreducible polynomial modulus $f(x) = x^d + 1$ (even degree and specific form chosen by SEAL for efficiency reasons).

We will also encode data in plaintexts using the so-called fractional encoding of [48], by expanding our finite precision floats u in a basis b as $u = \sum_{i=0}^{r} u_i b^i + \sum_{j=1}^{s} u_j b^{-i}$, with u_i, u_j then mapped to plaintext polynomial coefficients. This encoding depends on three parameters:

- the basis b
- the number of polynomial coefficient reserved for the fractional part $n_f = \max \text{ allowed } s$
- the number of polynomial coefficient reserved for the integer part $n_i = \max \text{ allowed } r$.

B. Datasets

As we have discussed in the introduction, only integers and finite precision floats (that is, rational numbers) are representable in the existing homomorphic cryptosystems. The datasets we have been using in our analysis are random datasets obtained from a uniform distribution of float values, and normalized according to the specifics we will illustrate in Sections III-C and III-D.

C. Empirical Study

We now turn to the empirical study of the algorithms leading to the functions in equations (12 - 17), which all depend on the single parameter r, the number of iterations to compute in the weak limit approximation derived from equation (7). We want to determine for which values of r and range of data arguments x_1, x_2 the algorithm is sufficiently accurate and this will involve studying the algorithms both in their unencrypted and encrypted form.

The tests are run over different datasets, specified in the respective subsections. The evaluation of the algorithm performances are based on the Mean Absolute Error (MAE):

$$\text{MAE}(X, Y) = \sum_{y \in Y} \frac{|x - y|}{|Y|} \tag{19}$$

where X is the set of expected values x, Y the set of obtained ones (from the algorithms) and $|Y| = |X|$ denotes its cardinality.

	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
MAE(r) for $\texttt{Select}_{>\frac{1}{2}}$	0.38	0.28	0.15	0.062	0.027	0.017	0.026	0.019
MAE(r) for $\texttt{Select}_{>}$	0.47	0.39	0.22	0.10	0.056	0.034	0.051	0.036

TABLE I: Values for the mean absolute error defined in Equation 19 for a number of iterations values r calculated over a set of points $x_t = 0.05 * s$, $-4 \leq s \leq 4$.

*1) **Evaluation of algorithm parameters: unencrypted form of the algorithm**:* We first study the algorithms in unencrypted form to establish the dependence of the results on r. We have considered a set X_t of samples x in the interval $|x| \in [0, 0.25)$ where the algorithms (12- 17) can operate. For each sample we have run the algorithm several times, for an increasing number of iterations r, starting from $r = 1$. We have then evaluated the accuracy of the algorithm in the unencrypted form by calculating the MAE.

We present in Figure 1 a series of illustrative plots. We have chosen to report here plots concerning $\texttt{Select}_{>\frac{1}{2}}(x, 0)$ with little loss of generality concerning the illustrative purpose, as those for $\texttt{Select}_{>}(x, 0)$ are quite similar, and as the algorithms for the $<$ operations are the same up to an intermediate sign. The plots in blue (rows one and three) show the returned results from $\texttt{Select}_{>\frac{1}{2}}(x, 0)$ against the number of iterations r. The plots in red (row two and four) show the value of the simple error defined as

$$\text{Simple\_Error}(x, r) = H(x) - \texttt{Select}_{>\frac{1}{2}}(x, 0) \tag{20}$$

where $H(x)$ is the Heaviside function. We have chosen to plot the results for some of the values x we have considered. In particular we plot for points with values x going from -0.20 to 0.20 in steps of 0.05 in order to provide a consistent coverage of the working interval of the algorithms.

We also present in Table I the values of the MAE, equation 19, for those set of values for x and for the tested number of iterations r in the cases of $\texttt{Select}_{>\frac{1}{2}}(x, 0)$ and $\texttt{Select}_{>}(x, 0)$. The results in the table show that, in the former, for $r = 4$ the MAE has dropped at around 6%, and for $r = 5$ around 3%, while in the latter the values are slightly bigger. The low degree of the approximating polynomial from equation (11), and thus low number of necessary consecutive multiplications, make this an interesting result for applications in homomorphic encryption.

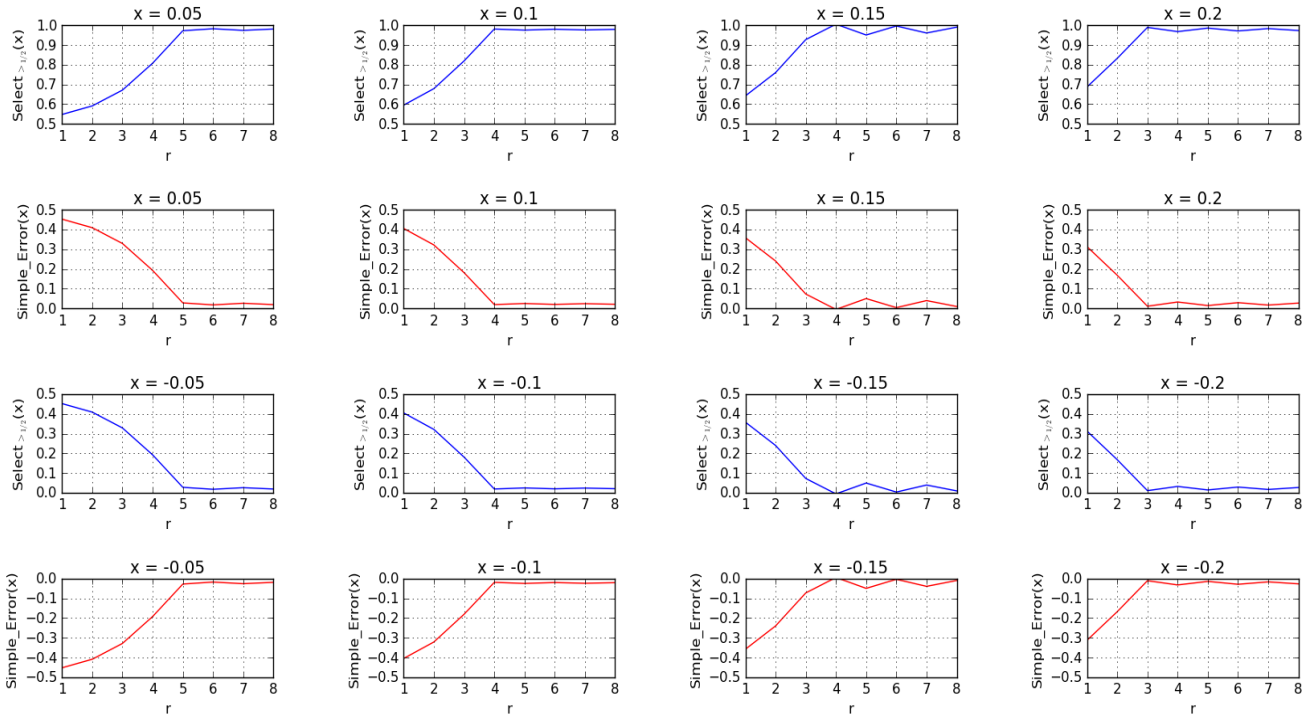

Fig. 1: Plots in rows one and three (blue color) show the value of the returned results from $\texttt{Select}_{>\frac{1}{2}}(z, 0)$ against the number of iterations r. The plots in row two and four(red color) show the value of the simple error defined in Equation 20.

*2) **Selection of algorithm and homomorphic scheme parameters: encrypted form of the algorithm**:* We move now to a series of tests with a carefully chosen artificial dataset to establish in the best r and FV cryptosystem parameters; where *best* means leading to the smallest errors over the maximum possible datavalue interval.

The value of r determines the number of consecutive operations the algorithms *must* sustain, while the parameters of the homomorphic scheme determine the number of consecutive operations the *encrypted* algorithm *can* sustain without bootstrapping. In the case of algorithms (13 - 17), the number of (noise-dominating) consecutive multiplications we need to be able to perform to run r iterations is

$$2r + 1_p \qquad \text{for the} \quad \texttt{Select}_{>/<\frac{1}{2}} \quad \text{algorithms} \tag{21}$$

$$2r + 1 + 1_p \qquad \text{for the} \quad \texttt{Select}_{>/</=} \quad \text{algorithms} \tag{22}$$

where 1_p is a multiplication with a plaintext coefficient[8], and $2r$ or $2r+1$ are ciphertext multiplications. The total count of operations is of course higher, when including additions and relinearizations, but as they generate less noise, we will neglect them. Moreover, note that the ciphertext multiplications involve multiplying recursively the *same* ciphertext, which means that successive multiplications are more costly for the noise growth, as they involved already noise-grown ciphertexts.

A number of consecutive operation such as, for instance, 8+1 (for $r = 4$) may seem not huge, but it must be put in relation with the cryptosystem parameters necessary to accommodate it. As said before, the analysis concerning the choice of parameters is still an active field of research, and there are in fact different partial results in the literature. For example [49] estimates the cryptosystem parameters for a scheme like the FV one, finding that already to perform 10 multiplications (of different, and thus with minimal noise, ciphertexts) requires a polynomial modulus degree of $d = 8192$ and a plaintext modulus of at least 2^{243} for a fractional encoding in base $b = 3$, which in turns implies a value for the ciphertext modulus of about 2^{226} to have 123 bits security as estimated in [50]. However, the work [12] claims that much higher values are actually necessary to be able to perform 4 subsequent multiplications, already in the case of a much simpler integer encoding ($t = 131702,\ q > 2^{159},\ d = 81920$. Finally, the recent paper [51] claimed necessary values of the form $t \gtrsim 2^{107}, d \leq 368$ when dealing with the case of a graph (representing an instance of Ivakhnenko's group method of data handling), whose evaluation along a path from input to output comported ≈ 6 consecutive multiplications (plus a similar quantity of additions).

In summary, two things appear from the literature:

- the parameter choice bounds are coarsely estimated for similar experiments
- the number of consecutive multiplications implemented in existing literature is very low, thus our $\sim 8+1$ one appears to be the highest ever tried.

We now present the results for the algorithms, $\texttt{Select}_{>/<\frac{1}{2}}$ and $\texttt{Select}_{>/<}$ (given the number of operations required by these, the results also apply to the case $\texttt{Select}_{=}$). We have been testing with parameters ranging over a number of possible values, in particular

$$\begin{aligned}
d &\in \{8192, 16384, 32768\} \\
q &\in \{2^{116} - 2^{18} + 1, 2^{226} - 2^{26} + 1, \\
&\qquad 2^{435} - 2^{33} + 1, 2^{829} - 2^{54} - 2^{53} - 2^{52} + 1\} \\
t &\in \{4096, 16384, 65536\} \\
b &\in \{3, 5, 7, 9\} \\
n_f &\in \{6, 8, 10, 24, 32\} \\
n_i &\in \{8, 16, 32, 64\}
\end{aligned} \tag{23}$$

where the parameters and their notation have been defined in Section IV-A. The time of key generation and storage overhead following from these choices of parameters are the known ones for the Fan-Verkauteren scheme and the SEAL implementation library, and we refer the reader to the literature, see the original articles [25], [44]. We also recall, and stress, that our results, at least the theoretical ones, are agnostic for what concerns the choice of homomorphic scheme, and thus any performing scheme may be used in practical applications.

The range of values for q, t, d were chosen by taking advantage of the sets of values indicated by the SEAL team in their testings of the library versions 2.1 and 2.2. Instead, SEAL version 2.3.0 uses a modification of the Fan-Vercauteren scheme to increase time efficiency, but which allow somewhat less flexibility and "ease" in the choice of parameters. In particular, the available values of the parameter q in version 2.3.0 have proven in our case to yield sub-optimal results.

We have run the algorithm $\texttt{Comp}(z, 0)$ over a small dataset

$$z \in \{-0.20, -0.15, -0.10, -0.05, 0, 0.05, 0.10, 0.15, 0.20\}$$

capable however to cover sufficiently uniformly the allowed instance space $|z| \in [0, 0.25)$ from Sections III-C and III-D.

We list in Table II the best results for each tested value of r, where best indicates smallest MAE for the selection weights $s_{ij}, \overline{\overline{s}}_{ij}, \widetilde{s}_{ij}$ as defined in Equation (18).

[8]We distinguish mixed plaintext/ciphertext multiplications because the noise level estimates are different than pure ciphertext multiplications in our implementation based on SEAL, see table 3 in [43].

Iterations	Results
$r = 3$	Smallest parameters where result still achieved: $d = 16384,\ q = 2^{435} - 2^{33} + 1,\ t = 65536,\ w = 7,\ n_i = 8,\ n_f = 8$
$r = 4$	Not accomplished correctly (error less than 1 at least) by any parameter value in 23. "Best results" for $d = 16384,\ q = 2^{435} - 2^{33} + 1,\ t = 65536,\ w = 7,\ n_i = 8,\ n_f = 8$

TABLE II: Values for the mean absolute error defined in equation 19 for a number of iterations values r calculated over a set of points $x_t = 0.05 * s,\ -4 \leq s \leq 4$.

$d = 16384,\ q = 2^{435} - 2^{33} + 1,\ t = 65536,\ w = 7,\ n_i = 8,\ n_f = 8$						
Iterations	$\texttt{Select}_{>\frac{1}{2}}$		$\texttt{Select}_{>}$		$\texttt{Select}_{=}$	
	MAE(s_{ij})	MAE($s_{ij}x$)	MAE($\overline{\overline{s}}_{ij}$)	MAE($\overline{\overline{s}}_{ij}x$)	MAE($\widetilde{s}_{ij}$)	MAE($\widetilde{s}_{ij}x$)
$r = 3$	0.26	0.021	0.41	0.023	0.52	0.057
$r = 4$	1.7	0.28	4.9	0.55	2.6	0.30

TABLE III: Errors for the comparison and full conditional algorithms defined in 12, 13, 15, 16. We have tested the algorithms on randomly generated datasets with batches of 60 couples of datapoints to be compared and present here the average results for $r = 3$, while for $r = 4$ we present the result for the best batch (since anyway the case $r = 4$ is affected by error of decryption due to too many consecutive operations performed, see text.

From our analysis, the rationale behind the effectiveness of encryption scheme parameters emerges as follows. First of all we need small t and large q since $\frac{q}{t}$ mostly determines the maximum noise bound, see [43]. Secondly, we need to keep the number of coefficients reserved to the fractional part in encoding (n_f) as small as possible because during multiplication the number of coefficients occupied by the fractional part will increase rapidly. The number of coefficients reserved to the integer part (n_i) is of less concern, because all the normalized test data instances x are smaller than one.

The basis b used for the fractional encoding, see Section IV-A also played an important role. One would like to have as small a basis as possible, to avoid the "wrapping up" of the modulo t during computations. However, smaller basis also means that more coefficients of the plaintext polynomial will be non-zero, and so since the number of coefficients of the fractional part increase with multiplications, they can more easily cross over to the coefficients reserved for the integer part, and ruin decryption.

Finally, the degree d of the polynomial modulus is relevant because of two different reasons: on the one hand the experiment should take into account security bounds, which depend on d since long polynomial are more difficult to attack; on the second hand having a big number of coefficients also helps in avoiding that those reserved to the fractional part and those to the integer part cross over and mix up rapidly.

*3) **Full tests: encrypted form of the algorithm**:* Having estimated as discussed the best algorithm and scheme parameters, we have finally run full-fledged tests over randomized datasets to assess the accuracy of the algorithms.

We have studied the algorithms $\texttt{Select}_{>\frac{1}{2}}$, $\texttt{Select}_{>}$ and $\texttt{Select}_{=}$, since the algorithms for the $<$ (less than) relations are essentially the same as the ones for $>$ up to an (intermediate) sign, hence the test results apply to those as well. Our datasts consisted of couples of datapoints randomly generated in the range $[-0.12, 0.12]$ (so that the difference between datapoint values would fall in the valid range $[0, 0.25)$ to apply the algorithms, see Sections III-C and III-D). We have used several measures of accuracy and performance for the algorithms, to be able to provide a rigorous evaluation.

The results are presented in Table III. The simplified notation MAE(a) indicates the error calculated using equation (19) on the values $a = a_{\text{out}} - a_{\text{expected}}$. We have studied various tests, in particular we have considered a to be first $s_{ij}, \overline{\overline{s}}_{ij}, \widetilde{s}_{ij}$ and then the final full output of the algorithms (that is, $a = s_{ij}x_i$ and the analogous for $\overline{\overline{s}}_{ij}, \widetilde{s}_{ij}$).

The best performing algorithms are $\texttt{Select}_{>\frac{1}{2}}$ (and thus $\texttt{Select}_{<\frac{1}{2}}$ as it is the same up to an initial sign), achieving about 20% error on the selection weights and 2% on the final conditional (including the selection/jump) output. The error is dominated by the error value for datapoints that are very close to each other. In fact, we have run the same tests with datapoints with a fixed minimal distance in order to check variations depending on this, and the error rate drops rapidly in function of the inter-distance of points (already with inter-distance higher than a few percent, for instance 3%, the error rate on selection weights drop at about 12%).

The algorithm $\texttt{Select}_{=}$ deserves a special comment: the comparison weights w_{ij} are exactly 1 when $x_i = x_j$ and different from 1 when $x_i \neq x_j$ (and closer to 0 as the difference/distance between x_i and x_j is larger), so that if in a simple application one let the dataowner simply decrypt the comparison weights and pick the datapoints corresponding to weights equal to 1 to perform the selection part of the conditional, one would have perfect accuracy. This however cannot be done when the algorithm must be inserted in a longer pipeline of algorithms and the "selection" must be performed on the encrypted parts and carried over to further steps of the pipeline. The results relative to $\texttt{Select}_{=}$ that we show in Table III are therefore to be intended for this case.

Average timing per instance in seconds			
$d = 16384,\ q = 2^{435} - 2^{33} + 1,\ t = 65536,\ w = 7,\ n_i = 8,\ n_f = 8$			
Iterations	$\texttt{Select}_{>\frac{1}{2}}$	$\texttt{Select}_{>}$	$\texttt{Select}_{=}$
$r = 3$	17.4 s	21.5 s	21.1 s
$r = 4$	30.5 s	31.5 s	31.2 s

TABLE IV: Values for the timing for runs of the conditionals algorithms in seconds per instance.

We finally present in Table IV the result for the timing of the comparison algorithm. Our work has not focused on achieving the best performance, as it has been more centered on the proof-of-concept and the practical implementation of the algorithms, as well as to the discussion of the novel issues concerning homomorphic encryption and applications such as machine learning (see Section V-A). We have however measured the timings when running our tests, and report in the table the average timing per (x_1, x_2) instance for the algorithms (13), (15), (16).

A direct comparison with the results reported in the literature is however not straightforward, because:

- there are very few works implementing similarly complex algorithms in a homomorphic cryptosystem,
- often the experiments in the literature have been performed on powerful computer clusters, see for instance [12],
- only few among the works with complex algorithms report full algorithm timings. [9]

D. Improvements, other possible implementations and overall comments

We have presented here above a series of algorithms to evaluate comparisons and conditionals in homomorphic encryption settings. The algorithms have been explicitly tested in a concrete implementation of the Fan-Vercauteren encryption scheme in a levelled form. The limitation on total number of consecutive operations has the strongest influence on the accuracy of the algorithms.

Such limitations, and hence inaccuracies, would simply be absent for an implementation in a fully homomorphic scheme, or, possibly, using schemes that although limited can tolerate a larger number of consecutive operations (we estimated 9 multiplications would already guarantee accuracy at per cent or sub per cent level).

It would be also interesting to assess what effects new plain- and ciphertext encodings such as [8] would have, possibly in alleviating some of the accuracy loss due to crossing of the integer and fractional parts of the standard encoder, see Section IV-C3.

V. Applications

A. Machine learning and specific issues

As mentioned in the introduction, part of the present interest in homomorphic encryption stems from the potential to permit privacy preservation to coexist with the nowadays ubiquitous machine learning/data mining/predictive analytics[10] without incurring in the limits of the privacy/data usefulness budget of other approaches [2].

In the literature of the past few years there has been a certain, limited number of implementations of machine learning algorithms[11] preserving privacy combined with homomorphic cryptography techniques, see for instance [7]–[12].

The main problem is that very often machine learning techniques are not amenable to homomorphic encryption due to various limitations and it is therefore interesting to re-think the actual machine learning algorithms.

*1) **Lightning introduction to relevant aspects of machine learning**:*

Developing machine learning/data mining/predictive analysis systems typically involves a series of different steps[12]: training, validation, testing, prediction. The core issue is coined as solving a "learning problem", which comes down to finding within a certain solution space (essentially delimited by the inference bias) a function or generalization thereof that maps input data to a correctly inferred output (be it classification or regression). To this end, the algorithm uses the input data to assess the relevance of different hypothesis in the solution space, building them against the available training data, and validating and testing them against independent pieces of data. The tested algorithm can then be used with other data for prediction.

[9]The others, for example [51], report "time per operation" where "operation" indicates addition, multiplication or encryption. However, also other routines such as relinearizations are part of the algorithms and finding the overall algorithm timing is not straightforward.

[10]Again, for brevity of expression in the following we will use "machine learning" as an umbrella term for all these different but related approaches.

[11]Typically for prediction only, that is having the training part all in unencrypted form.

[12]Not always: consider for example instance-based methods, such as k-nearest neighbours, which do not need an actual training, validation and testing phase.

*2) **Problematic points of machine learning in homomorphic settings: heuristics and stopping criterion**:*

We will now elaborate on certain ingredients of the machine learning inference process that clash in a particularly relevant way with fundamental constraints of homomorphic encryption. In a large class of machine learning algorithms two element are paramount: the *stopping criterion* and *heuristics*.

- **Stopping criterion**. Typically machine learning algorithms terminate when a stopping criterion is met, generally when an extremal condition is reached. This means that the algorithm must be able to evaluate a condition (the criterion) and select one of the options (essentially, continue or stop) while running in its encrypted form.
  This represents a clear example of evaluating a conditional, with its two steps (comparison and selection/jump), as we discussed in Section III-B.
  In Section III-D we explained that the selection/jump step conflicts with the fundamental requirement of semantic security in (homomorphic) encryption and we proposed some "selection by weighting" algorithm that allows mapping the selected-for and unselected-for subsets to specific subsets (in particular the zero element for the unselected subset) that at decryption will provide the desired result.
  The bigger problem in this case is that the "selection by weighting" does not really signal that a selection has been made, but all mapped results (both "selected" and "rejected" ones) are still encrypted and carried over. There is no way at run time to determine that the stopping criterion has thus been met and the procedure must be stopped, until decryption occurs. Unfortunately, and importantly, not stopping the training of an algorithm precisely at the stopping point does entail overfitting and thus suboptimal learning models.
- **Heuristics**. In order to efficiently explore the instance (data) and problem space, and make useful inference, several machine learning algorithms operate heuristic choices at run time. Again, the clash between the need to make selections and the requirement of semantic security of the (homomorphic) encryption pops up. Differently from the case of the stopping criterion, here our "selection by weighting" would not create loss of accuracy in the algorithms, but, of course, in the case of large datasets it would entail carrying over the full dataset all along, hence affecting the efficiency of the algorithm, and in certain cases its whole inference capabilities, as we will discuss at the end of this section.
  Also another issue can arise: the comparison weights w_{ij} are not exactly 1 or 0 but some other numbers (float) close to that, because of the limitations in the number of consecutive operations of the levelled homomorphic system one has to use in practice, which limits the value of the algorithm parameter r and thus its accuracy. This effectively transforms a machine learning algorithm into a weighted version of itself. In some cases this does not significantly affect the accuracy, sensitivity and precision of the algorithms, but in other cases it does, also in an adverse way. The studies in this respect are scarce in the literature, see e.g. [52] for what concerns clustering.

The two issues here presented are quite fundamental and could seriously complicate, if not make impossible, the implementation of privacy-preserving machine learning using purely homomorphic encryption. The stopping criterion issue, in particular, implies that the training of correctly performing algorithms (that is, not overfitted ones) does not seem achievable without decryption at runtime, which goes against the aim itself of homomorphic encryption. While this drawback affects only the training of models, private data are also used in training (even more than in the prediction runs after training) and should therefore be protected as well.

The heuristics issue instead seems to represent a secondary problem, only affecting performance and thus possibly solved by, for example, hardware evolution. However, that is not the case. To be able to make actual inferences several machine learning algorithms do need to operate with heuristics on the data/problem space. If that is not possible, the algorithms cannot proceed with meaningful inferences. Again, this appears to be a relevant obstacle on the road to make machine learning privacy-friendly by using homomorphic encryption.

B. Applications to algorithms different than machine learning

Algorithms different than machine learning or similar predictive analytic techniques that do not need to make inferences and avoid overfitting as discussed in the previous section are not in such a relevant clash on general grounds with homomorphic encryption.

This means that operations such as pure database searches, for instance, even including conditionals could be effectively performed taking advantage of the techniques and algorithms we have developed in this work and their future improvements.

VI. Conclusions

Homomorphic encryption could provide an elegant solution to the problem of privacy preservation in data-based applications, such as those provided and/or facilitated by machine learning techniques, but several limitations hamper the implementation of such program. In this work we have identified, on the basis of the structured programming theorem, the set of minimal operations that guarantee to be able to compute any computable function or algorithm. We have then focused on those that are still lacking in homomorphic encryption, namely comparisons and conditionals. We have discovered rather fundamental clashes between the necessity to implement those operations and the basic requirements of (homomoprhic encryption). We have also proposed practical implementations for those operations or their closest possible forms in homomorphic encryption.

The limitation on the total number of consecutive operations, due to the use of leveled homomorphic encryption schemes without using bootstrapping (a practical limitation we have to face), has had the strongest influence on the accuracy of our algorithms. Percent accuracy (and better) can be obtained however for datapoints which are sufficiently inter-spaced. Moreover, such limitations, and hence inaccuracy, would not occur in a fully homomorphic scheme, or, possibly, using schemes and/or encodings that can tolerate a larger number of consecutive operations even if only somewhat homomorphic (we estimate from 9 multiplications onward).

We have also analysed the specific situation arising in machine learning/predictive analytic applications. We have pointed out at least two main sources of tension with the use of homomorphic encryption to fully guarantee privacy preservation in machine learning due to the newly found above-mentioned issues. These two sources of tensions are the stopping criterion and heuristics. They are present and paramount in most machine learning algorithms, and clash with (homomorphic) encryption in that they require performing selection/jump operations at run time, which in its turn clashes with semantic security, as we have studied in this work.

Two options are open under this respect. On the one hand it might be possible to find new classes and families of learning algorithms that operate without "choices at run time". On the other, we could reconsider the use of homomorphic encryption. Maybe some other technology, such as for example functional encryption (see for example [6]) may be capable to avoid the need for high level of communication and intermediate decryption of other techniques (such as multiparty computation ones)? Functional encryption could allow to limit the computations to some agreed level, while preserving the rest of the privacy of the data or algorithm.

Note however that the class of algorithms which do not need to make inferences and where overfitting can be avoided, are not in conflict with the general grounds of homomorphic encryption. Thus, operations such as pure database searches, for instance, including conditionals could be performed taking advantage of the techniques and algorithms we have developed in this work and their future improvements. We believe that further exploring the use of homomorphic encryption in algorithms for privacy preservation is paramount.

VII. References

[1] C. Aggarwal, P. Yu, "Privacy-Preserving Data Mining: Models and Algorithms", Kluwer Academic Publishers Boston/-Dordrecht/London, 2008.

[2] Fredrikson M., Lantz E., Jha S., Lin S., Page D., Ristenpart T., "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing", 23rd USENIX Security Symposium (USENIX Security 14), 2014.

[3] Wee H., "Functional Encryption and Its Impact on Cryptography", SCN 2014: Security and Cryptography for Networks pp 318-323.

[4] D. Boneh, A. Sahai, B. Waters "Functional encryption: definitions and challenges" In proceedings of TCC'11, LNCS 6597, pp. 253-273. eprint.iacr.org/2010/543.pdf

[5] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang, "On the (Im)possibility of Obfuscating Programs", Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings

[6] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, "Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits". SIAM J. Comput. 45(3): 882-929 (2016)

[7] T. Graepel, K. Lauter, M. Naehrig "ML Confidential: Machine learning on encrypted data", in T. Kwon, M.-K. Lee and D. Kwon, eds, 'Information Security and Cryptology (ICISC 2012)', Vol. 7839 of Lecture Notes in Computer Science, Springer, pp. 1–21.

[8] C. Bonte, C. Bootland, J. W. Bos, W. Castryck, I. Iliashenko, and F. Vercauteren, "Faster Homomorphic Function Evaluation using Non-Integral Base Encoding," In Cryptographic Hardware and Embedded Systems - CHES 2017, Lecture Notes in Computer Science 10529, W. Fischer, and N. Homma (eds.), Springer-Verlag, pp. 579-600, 2017.

[9] K. Lauter, M. Naehrig, V. Vaikuntanathan, "Can homomorphic encryption be practical?", in 'Proceedings of the 3rd ACM workshop on Cloud computing security workshop', ACM, pp. 113–124, 2011, https://eprint.iacr.org/2011/405.pdf.

[10] J. W. Bos, K. Lauter, M. Naehrig, "Private predictive analysis on encrypted medical data", Journal of Biomedical Informatics 50, 234–243, 2014.

[11] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy", Proceedings of The 33rd International Conference on Machine Learning, pp. 201–210, 2016.

[12] L. J. M. Aslett, P. M. Esperança, C. C. Holmes, "Encrypted statistical machine learning: new privacy preserving methods", Technical report, University of Oxford, 2015.

[13] R. L. Rivest, L. Adleman, M. L. Dertouzos "On data banks and privacy homomorphisms", Foundations of Secure Computation 4(11), 169–180, 1978.

[14] C. Gentry, "A fully homomorphic encryption scheme", PhD thesis, Stanford Uni- versity, 2009. URL: crypto.stanford.edu/craig

[15] Fontaine C., Galand F., "A survey of homomorphic encryption for nonspecialists", EURASIP Journal on Information Security archive Volume 2007, January 2007 Article No. 15.

[16] Goldwasser S., Micali S., "Probabilistic encryption & how to play mental poker keeping secret all partial information", Proceedings of the fourteenth annual ACM symposium on Theory of computing, p.365-377, May 05-07, 1982, San Francisco, California, USA [DOI 10.1145/800070.802212].

[17] Goldreich O., "A uniform complexity treatment of encryption and zero-knowledge," Journal of Cryptology, vol. 6, no. 1, pp. 21-53, 1993.

[18] Halevi, S., Lindell, Y. "Homomorphic Encryption", chapter in the book "Tutorials on the Foundations of Cryptography: Dedicated to Oed Goldreich.", pages 219-276, Springer 2017.

[19] N.P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Public Key Cryptography – PKC 2010, Lecture Notes in Comput. Sci. 6056, 420–443, 2010.

[20] D. Stehlé, R. Steinfeld, "Faster fully homomorphic encryption", in Advances in Cryptology-ASIACRYPT 2010', Springer, pp. 377–394, 2010.

[21] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, "Fully homomorphic encryption over the integers", in Advances in Cryptology, EUROCRYPT 2010, Springer, pp. 24-43, 2010.

[22] Cheon J.H., Kim J., Lee M.S., Yun A., "CRT-based fully homomorphic encryption over the integers", Information Sciences, Volume 310, 20 July 2015, Pages 149-162.

[23] Brakerski, Z. and Vaikuntanathan, V. (2011a), Efficient fully homomorphic encryption from (standard) LWE, in 2011 IEEE 52nd Annual Symposium on Foundations of Com- puter Science', IEEE, pp. 97–106.

[24] Brakerski Z., Gentry C., Vaikuntanathan V., "(Leveled) Fully homomorphic encryption without bootstrapping", in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference', ACM, pp. 309-325, 2012.

[25] J. Fan, F. Vercauteren, "Somewhat practical fully homomorphic encryption", IACR Cryptology ePrint Archive eprint.iacr.org/2012/144.

[26] C. Gentry, A. Sahai, B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based", in Advances in Cryptology CRYPTO 2013', Springer, pp. 75–92, 2013.

[27] A. Lopez-Alt, E. Tromer, V. Vaikuntanathan "On-the- fly multiparty computation on the cloud via multikey fully homomorphic encryption", Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, pages 1219–1234. ACM, 2012, eprint.iacr.org/2013/094.pdf.

[28] Bohm, C.; Jacopini G., "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". Communications of the ACM. 9 (5): 366–371 (May 1966).

[29] Agrawal R., Kiernan J., Stikant R., Xu Y., "Order preserving encryption for numeric data", Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 563-574.

[30] Boldyreva A., Chenette N., Lee Y., Neill A.O., "Order-preserving symmetric encryption", in: A. Joux (Ed.), Advances in Cryptology EUROCRYPT 2009, Vol. 5479 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 224-241 .

[31] Boldyreva A., Chenette N., Neill A.O., "Order-preserving encryption revisited: improved security analysis and alternative solutions", in: P. Rogaway (Ed.), Advances in Cryptology C CRYPTO 2011, Vol. 6841 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 578-595 .

[32] Popa R., Li F., Zeldovich N., "An ideal-security protocol for order-preserving encoding", in: Security and Privacy (SP), 2013 IEEE Symposium on, vol. 465, 2013, pp. 463-477 .

[33] Mavroforakis C., Chenette N., Neill A.O., Kollios G., Canetti R. "Modular Order-Preserving Encryption, Revisited", Proceeding SIGMOD '15 Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data Pages 763-777.

[34] Lewi K., Wu D., "Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds" Proceeding CCS '16 Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security Pages 1167-1178

[35] Bëtul Durc F., DuBuisson T. M., Cash D., "What lse is revealed by Order-revealing encryption?", published in ACM CCS 2016.

[36] Armknecht F., Boyd C., Carr C., Gjøsteen K., Jäschke A., Reuter C., Strand M., "A Guide to Fully Homomorphic Encryption", Cryptology ePrint Archive, Report 2015/1192, 2015.

[37] Togan M., Plasca C., "Comparison-Based computations over fully homomorphic encrypted data", COMM 2014 International Conference.

[38] Bost R., Ada Popa R., Tu S., Goldwasser S., "Machine Learning Classification over Encrypted Data, NDSS 2015, Cryptology ePrint Archive, Report Report 2014/331, version 2015.

[39] Kanwal, R. P. Generalized Functions: Theory and Technique, 2nd ed. Boston, MA: Birkhäuser, 1998.

[40] https://www.mathworks.com/products/matlab .

[41] Zvika Brakerski 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In Advances in Cryptology–CRYPTO 2012. Springer, 868–886.

[42] Halevi S., Shoup V. "Bootstrapping for helib", in Oswald and Fischlin Oswald E., Fischlin M., editors. "Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques", Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, volume 9056 of Lecture Notes in Computer Science. Springer, 2015, pages 641–670.

[43] Chen H., Laine K., Player P., "Simple Encrypted Arithmetic Library - SEAL v2.1", Cryptology ePrint Archive, Report

2017/224, https://eprint.iacr.org/.
[44] https://www.microsoft.com/en-us/research/project/ simple-encrypted-arithmetic-library/.
[45] L. J. M. Aslett (2014), HomomorphicEncryption: Fully Homomorphic Encryption. R package version 0.2. URL: http: // www. louisaslett. com/ HomomorphicEncryption/
[46] https://github.com/CryptoExperts/FV-NFLlib/blob/master/FV.hpp
[47] S. Halevi, V. Shoup, (2014), 'Helib', https://github.com/shaih/HElib.
[48] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, "Manual for Using Homomorphic Encryption for Bioinformatics", Published in: Proceedings of the IEEE (Volume: 105, Issue: 3, March 2017)
[49] Chenette N., Lewi K., Weis S. A., Wu D. J., "Practical order-revealing encryption with limited leakage", 2015, Cryptology eprint archive: 2015/1125.
[50] Chen H., Laine K., Player P., "Simple Encrypted Arithmetic Library - SEAL v2.2", https://www.microsoft.com/en-us/research/wp-content/uploads/2017/06/sealmanual_v2.2.pdf.
[51] Bos J. W., Castryck W., Iliashenko I., Vercauteren F., "Privacy-friendly forecasting for the Smart Grid using Homomorphic Encryption and the Group Method of Data Handling", 2016, Cryptology eprint archive: 2016/1117.
[52] Ackerman M., Ben-David S., Branzei S., LokerD., "Weighted clustering". In Proc. 26th AAAI Conference on Artificial Intelligence, 2012.
[53] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, T. Ristenpart, "Leakage-Abuse Attacks against Order-Revealing Encryption", IEEE 2017 Symposium on Security and Privacy : 655-672.
[54] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-Preserving Face Recognition". PETS 2009: 235-253.
[55] J. Liu, M. Juuti, Y. Lu, N. Asokan. "Oblivious Neural Network Predictions via MiniONN Transformations". ACM CCS 2017: 619-631.
[56] D. Demmler, T. Schneider, M. Zohner. 2015. "ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation". In 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015.
[57] P. Mohassel, Y. Zhang. "SecureML: A System for Scalable Privacy-Preserving Machine Learning". IEEE S&P 2017: 19-38.
[58] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. Usenix Security 2018: 1651-1669.

Diego Chialva is with the ERCEA (European Research Council Executive Agency), where he is in charge of policy data analytics and data management. This article is based on work performed when he was member of the Department TONA of the Vrije Universiteit Brussel (VUB), where he lead the software develoments, algorithmic design and implementations for the design research group. He holds a Master (Laurea Specialistica) in physics (University of Turin, Italy), a Master in Applied Computer Science (Vrij Universitet Brussel, Belgium) and Ph.D. in Theoretical and Elementary Particle Physics (SISSA-ISAS, Trieste, Italy). He has held researcher and visiting positions also at the University of Uppsala (Sweden), Nordita (Sweden), IHES (France), UCL (Belgium). His resarch activity, both in and outside academy, has crossed several fields of research, starting from theoretical and mathematical physics, and then specializing in machine learning, data mining, information systems, information retrieval, privacy-preservation techniques and homomorphic encryption.

Ann Dooms is professor at the Department of Mathematics (DWIS) of the Vrije Universiteit Brussel (VUB), where she leads the research group Digital Mathematics (DIMA). She received the degree and PhD in Mathematics in 2000 and 2004 respectively, both from the VUB. She specializes in the mathematical foundations for digital data acquisition, representation, analysis, communication, security and forensics. Dooms was elected as a member of the Royal Flemish Young Academy of Sciences and Arts and the IEEE Technical Committee in Information Forensics and Security.