Private AI:
Machine Learning on Encrypted Data

Kristin E. Lauter

Abstract As the world adopts Artificial Intelligence (AI), the privacy risks are many. AI can improve our lives, but may leak or misuse our private data. Private AI is based on Homomorphic Encryption (HE), a new encryption paradigm which allows the cloud to operate on private data in encrypted form, without ever decrypting it, enabling private training and private prediction with AI algorithms. The 2016 ICML CryptoNets [26] paper demonstrated for the first time evaluation of neural net predictions on homomorphically encrypted data, and opened new research directions combining machine learning and cryptography. The security of Homomorphic Encryption is based on hard problems in mathematics involving lattices, a candidate for post-quantum cryptography. This paper gives an overview of my Invited Plenary Lecture at the International Congress of Industrial and Applied Mathematics (ICIAM), explaining Homomorphic Encryption, Private AI, and real-world applications.

1 Motivation: Privacy in Artificial Intelligence

These days more and more people are taking advantage of cloud-based artificial intelligence (AI) services on their smart phones to get useful predictions such as weather, directions, or nearby restaurant recommendations based on their location and other personal information and preferences. The AI revolution that we are experiencing in the high tech industry is based on the following value proposition: you input your private data and agree to share it with the cloud service in exchange for some useful prediction or recommendation. In some cases the data may contain extremely personal information, such as your sequenced genome, your health record, or your minute-to-minute location.

Kristin E. Lauter
Research Manager, Cryptography and Privacy Research, Microsoft Research, Redmond, USA, e-mail: klauter@microsoft.com

This quid pro quo may lead to the unwanted disclosure of sensitive information or an invasion of privacy. Examples during the year of ICIAM 2019 include the case of the Strava fitness app which revealed the location of U.S. army bases world-wide, or the case of the city of Los Angeles suing IBM's weather company over deceptive use of location data. It is hard to quantify the potential harm from loss of privacy, but employment discrimination or loss of employment due to a confidential health or genomic condition are potential undesirable outcomes. Corporations also have a need to protect their confidential customer and operations data while storing, using, and analyzing it.

To protect privacy, one option is to lock down personal information by encrypting it before uploading it to the cloud. However, traditional encryption schemes do not allow for any computation to be done on encrypted data. In order to make useful predictions, we need a new kind of encryption which maintains the structure of the data when encrypting it so that meaningful computation is possible. Homomorphic Encryption allows us to switch the order of encryption and computation: we get the same result if we first encrypt and then compute, as if we first compute and then encrypt.

The first solution for a Homomorphic Encryption scheme which can process any circuit was proposed in 2009 by Gentry [25]. Since then, many researchers in cryptography have worked hard to find schemes which are both practical and also based on well-known hard math problems. In 2011, my team at Microsoft Research collaborated on the Homomorphic Encryption schemes [8, 9] and introduced a practical encoding method and improvements [34] which are now widely used in applications of Homomorphic Encryption. In 2012, using this practical data encoding method, the "ML Confidential" paper [27] was the first to demonstrate *training* ML algorithms on homomorphically encrypted data and to show initial performance numbers for simple models such as linear means classifiers and gradient descent. Then in 2016, a surprise breakthrough in the influential CryptoNets paper [26] demonstrated for the first time that evaluation of neural network predictions was possible on encrypted data.

Thus began our Private AI project, the topic of my Invited Plenary Lecture at the International Congress of Industrial and Applied Mathematics in Valencia in July 2019. Private AI refers to our Homomorphic Encryption-based tools for protecting the privacy of enterprise, customer, or patient data, while doing Machine Learning (ML)-based AI, both learning classification models and making valuable predictions based on such models.

You may ask, "What is privacy?" Preserving "privacy" can mean different things to different people or parties. Researchers in many fields including social science and computer science have formulated and discussed definitions of privacy. My favorite definition of privacy is: a person or party should be able to control how and when their data is used or disclosed. This is exactly what Homomorphic Encryption enables.

1.1 Real-world applications

In 2019, the British Royal Society released a report on Protecting Privacy in Practice: Privacy Enhancing Technologies in Data Analysis. The report covers Homomorphic Encryption (HE) and Secure Multi-Party Computation (MPC), but also technologies not built with cryptography, including Differential Privacy (DP) and secure hardware hybrid solutions. Our Homomorphic Encryption project was featured as a way to protect "privacy as a human right" at the Microsoft Build world-wide developers' conference in 2018 [43]. Private AI forms one of the pillars of Responsible ML at Microsoft Research as part of our collection of Responsible AI research, and Private Prediction notebooks were released in Azure ML at Build 2020.

Over the last 8 years, my team has created demos of Private AI in action, running private analytics services in the Azure cloud. I showed a few of these demos in my talk at ICIAM in Valencia. Our applications include an encrypted fitness app, which is a cloud service which processes all your workout and fitness data and locations in the cloud in encrypted form, and displays your summary statistics to you on your phone after decrypting the results of the analysis locally. Another application shows an encrypted weather prediction app, which takes your encrypted zip-code and returns encrypted versions of the weather at your location to be decrypted and displayed to you on your phone. The cloud service never learns your location or what weather data was returned to you. Finally, I showed a private medical diagnosis application, which uploads an encrypted version of a chest X-ray image. The medical condition is diagnosed by running image recognition algorithms on the encrypted image in the cloud, and the encrypted diagnosis is returned to the doctor.

Over the years, my team[1] has developed other Private AI applications, enabling private predictions such as sentiment analysis in text, cat/dog image classification, heart attack risk based on personal health data, neural net image recognition of hand-written digits, flowering time based on the genome of a flower, and pneumonia mortality risk using intelligible models. All of these operate on encrypted data in the cloud to make predictions, and return encrypted results in a matter of fractions of a second.

Many of these demos and applications were inspired by collaborations with researchers in medicine, genomics, bioinformatics, and machine learning (ML). We worked together with finance experts and pharmaceutical companies to demonstrate a range of ML algorithms operating on encrypted data. The UK Financial Conduct Authority (FCA) ran an international hackathon in August 2019 to combat money-laundering with encryption technologies by allowing banks to share confidential information with each other. Since 2015, the annual NIH-funded iDASH competition has attracted teams from around the world to submit solutions to the Secure Genome Analysis Competition. Participants include researchers at companies such

[1] My collaborators on the SEAL team include: Kim Laine, Hao Chen, Radames Cruz, Wei Dai, Ran Gilad-Bachrach, Yongsoo Song, Shabnam Erfani, Sreekanth Kannepalli, Jeremy Tieman, Tarun Singh, Hamed Khanpour, Steven Chith, James French, with substantial contributions from interns Gizem Cetin, Kyoohyung Han, Zhicong Huang, Amir Jalali, Rachel Player, Peter Rindal, Yuhou Xia as well.

as Microsoft and IBM, start-up companies, and academics from the U.S., Korea, Japan, Switzerland, Germany, France, etc. The results provide benchmarks for the medical research community of the performance of encryption tools for preserving privacy of health and genomic data.

2 What is Homomorphic Encryption?

I could say, "Homomorphic Encryption is encryption which is homomorphic." But that is not very helpful without further explanation. Encryption is one of the building blocks of cryptography: encryption protects the confidentiality of information. In mathematical language, encryption is just a map which transforms plaintexts (unencrypted data) into ciphertexts (encrypted data), according to some recipe. Examples of encryption include blockciphers, which take sequences of bits and process them in blocks, passing them through an S-box which scrambles them, and iterating that process many times. A more mathematical example is RSA encryption, which raises a message to a certain power modulo a large integer N, whose prime factorization is secret, $N = p \cdot q$, where p and q are large primes of equal size with certain properties.

A map which is *homomorphic* preserves the structure, in the sense that an operation on plaintexts should correspond to an operation on ciphertexts. In practice that means that switching the order of operations preserves the outcome after decryption: i.e. *encrypt-then-compute* and *compute-then-encrypt* give the same answer. This property is described by the following diagram:

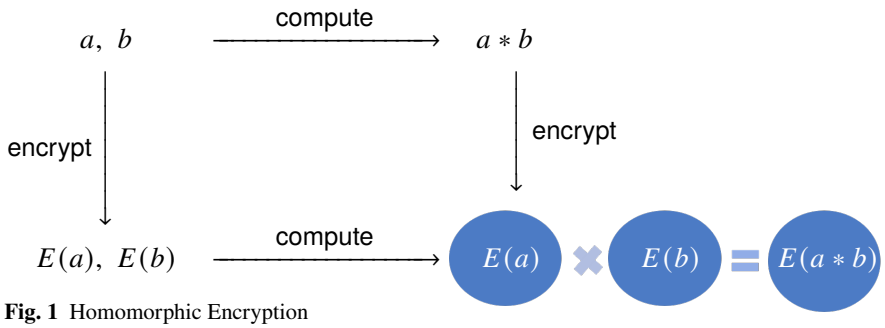

Fig. 1 Homomorphic Encryption

Starting with two pieces of data, a and b, the functional outcome should be the same when following the arrows in either direction, across and then down (*compute-then-encrypt*), or down and then across (*encrypt-then-compute*):

$$E(a+b) = E(a) + E(b).$$

If this diagram holds for two operations, addition and multiplication, then any circuit of AND and OR gates can be evaluated on data encrypted under the map E. It is

important to note that Homomorphic Encryption solutions provide for randomized encryption, which is a crucial property to protect against so-called dictionary attacks. This means that new randomness is used each time a value is encrypted, and it should not be computationally feasible to detect whether two ciphertexts are the encryption of the same plaintext or not. Thus the ciphertexts in the bottom right corner of the diagram need to be decrypted in order to detect whether they are equal.

The above description gives a mathematical explanation of Homomorphic Encryption by defining its properties. To return to the motivation of Private AI, another way to describe Homomorphic Encryption is to explain the functionality that it enables. Figure 2 shows Homer-morphic encryption, where Homer Simpson is a jeweler tasked with making jewelry given some valuable gold. Here the gold represents some private data, and making jewelry is analogous to analyzing the data by applying some AI model. Instead of accessing the gold directly, the gold remains in a locked box, and the owner keeps the key to unlock the box. Homer can only handle the gold through gloves inserted in the box (analogous to handling only encrypted data). When Homer completes his work, the locked box is returned to the owner who unlocks the box to retrieve the jewelry.

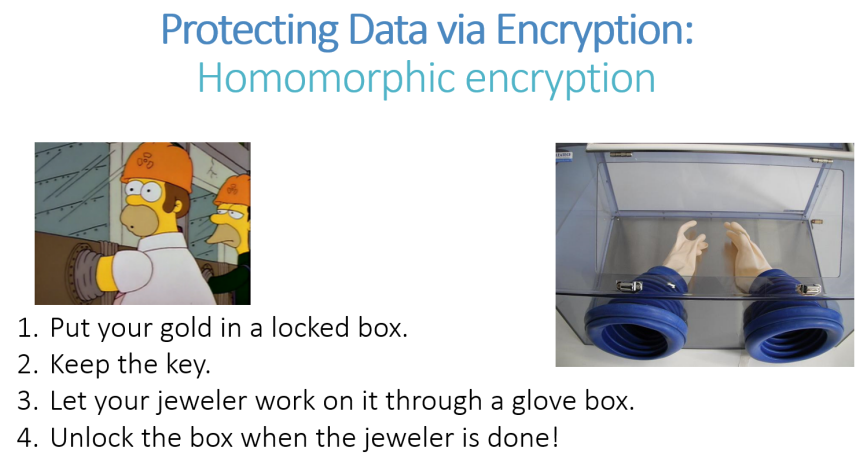

Fig. 2 Homer-morphic Encryption

To connect to Figure 1 above, outsourcing sensitive work to an untrusted jeweler (cloud) is like following the arrows down, across, and then up. First the data owner encrypts the data and uploads it to the cloud, then the cloud operates on the encrypted data, then the cloud returns the output to the data owner to decrypt.

2.1 History

Almost 5 decades ago, we already had an example of encryption which is homomorphic for one operation: the RSA encryption scheme [40]. A message m is encrypted by raising it to the power e modulo N for fixed integers e and N. Thus the product of the encryption of two messages m_1 and m_2 is $m_1^e m_2^e = (m_1 m_2)^e$. It was an open problem for more than thirty years to find an encryption scheme which was homomorphic with respect to two (ring) operations, allowing for the evaluation of any circuit. Boneh-Goh-Nissim [3] proposed a scheme allowing for unlimited additions and one multiplication, using the group of points on an elliptic curve over a finite field, along with the Weil pairing map to the multiplicative group of a finite field.

In 2009, Gentry proposed the first Homomorphic Encryption scheme, allowing in theory for evaluation of arbitrary circuits on encrypted data. However it took several years before researchers found schemes which were implementable, relatively practical, and based on known hard mathematical problems. Today all the major Homomorphic Encryption libraries world-wide implement schemes based on the hardness of lattice problems. A lattice can be thought of as a discrete linear subspace of Euclidean space, with the operations of vector addition, scalar multiplication, and inner product, and its dimension, n, is the number of basis vectors.

2.2 Lattice-based solutions

The high-level idea behind current solutions for Homomorphic Encryption is as follows. Building on an old and fundamental method of encryption, each message is *blinded*, by adding a random inner product to it: the inner product of a secret vector with a randomly generated vector. Historically, blinding a message with fresh randomness was the idea behind encryption via *one-time pads*, but those did not satisfy the homomorphic property. Taking inner products of vectors is a linear operation, but if Homomorphic Encryption involved only addition of the inner product, it would be easy to break using linear algebra. Instead, the encryption must also add some freshly generated noise to each blinded message, making it difficult to separate the noise from the secret inner product. The noise, or *error*, is selected from a fairly narrow Gaussian distribution.

Thus the security of the encryption is based on the hard problem called Learning-With-Errors (LWE), to find the secret vector, s given many samples of noisy inner products with the secret. It is a noisy decoding problem in a linear space, essentially Bounded Distance Decoding (BDD) or a Closest Vector Problem (CVP) in a lattice. Decryption is possible with the secret key, because the decryptor can subtract the secret inner product and then the noise is small and is easy to cancel.

Although the above high-level description was formulated in terms of lattices, in fact the structure that we use in practice is a polynomial ring. A vector in a lattice of n dimensions can be thought of as a monic polynomial of degree n, where the coordinates of the vector are the coefficients of the polynomial. Any number ring is

given as a quotient of $\mathbb{Z}[x]$, the polynomial ring with integer coefficients, by a monic irreducible polynomial $f(x)$. The ring can be thought of as a lattice in $\mathbb{R}^n$ when embedded into Euclidean space via the canonical embedding. To make all objects finite, we consider these polynomial rings modulo a large prime q, which is often called the ciphertext modulus.

The ring version of the Learning With Errors problem is called Ring Learning With Errors (RLWE). The first homomorphic encryption solution based on RLWE was proposed in [10], where they introduced a variant of RLWE called the Polynomial Learning With Errors (PLWE) problem. It was later shown that the Polynomial Learning With Errors (PLWE) problem is not hard for general polynomial rings and small error [22]. This attack was then extended to apply to the RLWE problem for many rings and small error [23, 14]. However none of these attacks apply to the 2-power cyclotomic polynomial rings which are used in practice.

2.3 Encoding data

When thinking about practical applications, it becomes clear that real data first has to be embedded into the mathematical structure that the encryption map is applied to, the *plaintext space*, before it is encrypted. This encoding procedure must also be homomorphic in order to achieve the desired functionality. The encryption function will be applied to elements in the polynomial ring with integer coefficients modulo q, so real data must be embedded into this polynomial ring.

In a now widely cited 2011 paper, "Can Homomorphic Encryption be Practical?" ([34, Section 4.1]), we introduced a new way of encoding real data in the polynomial space which allowed for efficient arithmetic operations on real data, opening up a new direction of research focusing on practical applications and computations. The encoding technique was simple: embed an integer m as a polynomial whose i^{th} coefficient is the i^{th} bit of the binary expansion of m (using the ordering of bits so that the least significant bit is encoded as the constant term in the polynomial). This allows for direct multiplication of real integers, represented as polynomials, instead of encoding and encrypting data bit-by-bit, which requires a deep circuit just to evaluate simple integer multiplication. When using this approach, it is important to keep track of the growth of the size of the output to the computation. In order to assure correct decryption, we limit the total size of the polynomial coefficients to t. Note that each coefficient was a single bit to start with, and a sum of k of them grows to at most k. We obtain the correct decryption and decoding as long as $q > t > k$, so that the result does not wrap around modulo t.

This encoding of integers as polynomials has two important implications, for performance and for storage overhead. In addition to enabling multiplication of floating point numbers via direct multiplication of ciphertexts (rather than requiring deep circuits to multiply data encoded bit wise), this technique also saves space by packing a large floating point number into a single ciphertext, reducing the storage overhead. These encoding techniques help to squash the circuits to be evaluated, and

make the size expansion reasonable. However, they limit the possible computations in interesting ways, and so all computations need to be expressed as polynomials. The key factor in determining the efficiency is the degree of the polynomial to be evaluated.

2.4 Brakerski/Fan-Vercauteren Scheme (BFV)

For completeness, I will describe one of the most widely used Homomorphic Encryption schemes, the Brakerski/Fan-Vercauteren Scheme (BFV) [7, 24], using the language of polynomial rings.

2.4.1 Parameters and notation.

Let $q \gg t$ be positive integers and n a power of 2. Denote $\Delta = \lfloor q/t \rfloor$. Define

$$R = \mathbb{Z}[x]/(x^n + 1),$$

$$R_q = R/qR = (\mathbb{Z}/q\mathbb{Z})[x]/(x^n + 1),$$

and $R_t = \mathbb{Z}/t\mathbb{Z}[x]/(x^n + 1)$, where $\mathbb{Z}[x]$ is the set of polynomials with integer coefficients and $(\mathbb{Z}/q\mathbb{Z})[x]$ is the set of polynomials with integer coefficients in the range $[0, q)$.

In the BFV scheme, plaintexts are elements of R_t, and ciphertexts are elements of $R_q \times R_q$. Let χ denote a narrow (centered) discrete Gaussian error distribution. In practice, most implementations of Homomorphic Encryption use a Gaussian distribution with standard deviation $\sigma[\chi] \approx 3.2$. Finally, let U_k denote the uniform distribution on $\mathbb{Z} \cap [-k/2, k/2)$.

2.4.2 Key generation.

To generate a public key, $\mathtt{pk}$, and a corresponding secret key, $\mathtt{sk}$, sample $s \leftarrow U_3^n$, $a \leftarrow U_q^n$, and $e \leftarrow \chi^n$. Each of s, a, and e is treated as an element of R_q, where the n coefficients are sampled independently from the given distributions. To form the public key–secret key pair, let

$$\mathtt{pk} = ([-(as + e)]_q, a) \in R_q^2,\ \mathtt{sk} = s$$

where $[\cdot]_q$ denotes the (coefficient-wise) reduction modulo q.

2.4.3 Encryption.

Let $m \in R_t$ be a plaintext message. To encrypt m with the public key $\mathtt{pk} = (p_0, p_1) \in R_q^2$, sample $u \leftarrow U_3^n$ and $e_1, e_2 \leftarrow \chi^n$. Consider u and e_i as elements of R_q as in key generation, and create the ciphertext

$$\mathtt{ct} = ([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q) \in R_q^2 .$$

2.4.4 Decryption.

To decrypt a ciphertext $\mathtt{ct} = (c_0, c_1)$ given a secret key $\mathtt{sk} = s$, write

$$\frac{t}{q}(c_0 + c_1 s) = m + v + bt ,$$

where $c_0 + c_1 s$ is computed as an integer coefficient polynomial, and scaled by the rational number t/q. The polynomial b has integer coefficients, m is the underlying message, and v satisfies $\|v\|_\infty \ll 1/2$. Thus decryption is performed by evaluating

$$m = \left\lfloor \frac{t}{q}(c_0 + c_1 s) \right\rceil_t ,$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

2.4.5 Homomorphic computation

Next we see how to enable addition and multiplication of ciphertexts. Addition is easy: we define an operation $\oplus$ between two ciphertexts $\mathtt{ct}_1 = (c_0, c_1)$ and $\mathtt{ct}_2 = (d_0, d_1)$ as follows:

$$\mathtt{ct}_1 \oplus \mathtt{ct}_2 = ([c_0 + d_0]_q, [c_1 + d_1]_q) \in R_q^2 .$$

Denote this homomorphic sum by $\mathtt{ct}_{\mathrm{sum}} = (c_0^{\mathrm{sum}}, c_1^{\mathrm{sum}})$, and note that if

$$\frac{t}{q}(c_0 + c_1 s) = m_1 + v_1 + b_1 t , \quad \frac{t}{q}(d_0 + d_1 s) = m_2 + v_2 + b_2 t ,$$

then

$$\frac{t}{q}(c_0^{\mathrm{sum}} + c_1^{\mathrm{sum}} s) = [m_1 + m_2]_t + v_1 + v_2 + (b_1 + b_2)t ,$$

As long as $\|v_1 + v_2\|_\infty < 1/2$, the ciphertext $\mathtt{ct}_{\mathrm{sum}}$ is a correct encryption of $[m_1 + m_2]_t$.

Similarly, there is an operation $\otimes$ between two ciphertexts that results in a ciphertext decrypting to $[m_1 m_2]_t$, as long as $\|v_1\|_\infty$ and $\|v_2\|_\infty$ are small enough. Since $\otimes$ is more difficult to describe than $\oplus$, we refer the reader to [24] for details.

2.4.6 Noise.

In the decryption formula presented above the polynomial v with rational coefficients is assumed to have infinity-norm less than $1/2$. Otherwise, the plaintext output by decryption will be incorrect. Given a ciphertext $\mathtt{ct} = (c_0, c_1)$ which is an encryption of a plaintext m, let $v \in \mathbb{Q}[x]/(x^n + 1)$ be such that

$$\frac{t}{q}(c_0 + c_1 s) = m + v + bt\,.$$

The infinity norm of the polynomial v called the noise, and the ciphertext decrypts correctly as long as the noise is less than $1/2$.

When operations such as addition and multiplication are applied to encrypted data, the noise in the result may be larger than the noise in the inputs. This noise growth is very small in homomorphic additions, but substantially larger in homomorphic multiplications. Thus, given a specific set of encryption parameters (n, q, t, χ), one can only evaluate computations of a bounded size (or bounded multiplicative depth).

A precise estimate of the noise growth for the YASHE scheme was given in [4] and these estimates were used in [5] to give an algorithm for selecting secure parameters for performing any given computation. Although the specific noise growth estimates needed for this algorithm do depend on which Homomorphic Encryption scheme is used, the general idea applies to any scheme.

2.5 Other Homomorphic Encryption Schemes

In 2011, researchers at Microsoft Research and Weizmann Institute published the (BV/BGV [8, 9]) Homomorphic Encryption scheme which is used by teams around the world today. In 2013, IBM released HELib, a Homomorphic Encryption library for research purposes, which implemented the BGV scheme. HELib is written in C++ and uses the NTL mathematical library. The Brakerski/Fan-Vercauteren (BFV) scheme described above was proposed in 2012. Alternative schemes with different security and error-growth properties were proposed in 2012 by Lopez-Alt, Tromer, and Vaikuntanathan (LTV [37]), and in 2013 by Bos, Lauter, Loftus, and Naehrig (YASHE [4]). The Cheon-Kim-Kim-Song (CKKS [16]) scheme was introduced in 2016, enabling approximate computation on ciphertexts.

Other schemes [18, 21] for general computation on bits are more efficient for logical tasks such as comparison, which operate bit-by-bit. Current research attempts to make it practical to switch between such schemes to enable both arithmetic and logical operations efficiently ([6]).

2.6 Microsoft SEAL

Early research prototype libraries were developed by the Microsoft Research (MSR) Cryptography group to demonstrate the performance numbers for initial applications such as those developed in [4, 5, 27, 33]. But due to requests from the biomedical research community, it became clear that it would be very valuable to develop a well-engineered library which would be widely usable by developers to enable privacy solutions. The Simple Encrypted Arithmetic Library (SEAL) [41] was developed in 2015 by the MSR Cryptography group with this goal in mind, and is written in C++. Microsoft SEAL was publicly released in November 2015, and was released open source in November 2018 for commercial use. It has been widely adopted by teams worldwide and is freely available online (`http://sealcrypto.org`).

Microsoft SEAL aims to be easy to use for non-experts, and at the same time powerful and flexible for expert use. SEAL maintains a delicate balance between usability and performance, but is extremely fast due to high-quality engineering. SEAL is extensively documented, and has no external dependencies. Other publicly available libraries include HELib from IBM, PALISADE by Duality Technologies, and HEAAN from Seoul National University.

2.7 Standardization of Homomorphic Encryption [1]

When new public key cryptographic primitives are introduced, historically there has been roughly a 10-year lag in adoption across the industry. In 2017, Microsoft Research Outreach and the MSR Cryptography group launched a consortium for advancing the standardization of Homomorphic Encryption technology, together with our academic partners, researchers from government and military agencies, and partners and customers from various industries: Homomorphic Encryption.org. The first workshop was hosted at Microsoft in July 2017, and developers for all the existing implementations around the world were invited to demo their libraries.

At the July 2017 workshop, participants worked in groups to draft three white papers on Security, Applications, and Schemes. We then worked with all relevant stakeholders of the HE community to revise the Security white paper [12] into the first draft standard for Homomorphic Encryption [1]. The Homomorphic Encryption Standard (HES) specifies secure parameters for the use of Homomorphic Encryption. The draft standard was initially approved by the HomomorphicEncryption.org community at the second workshop at MIT in March 2018, and then was finalized and made publicly available at the third workshop in October 2018 at the University of Toronto [1]. A study group was initiated in 2020 at the ISO, the International Standards Organization, to consider next steps for standardization.

3 What kind of computation can we do on encrypted data?

3.1 Statistical computations

In early work, we focused on demonstrating the feasibility of statistical computations on health and genomic data, because privacy concerns are obvious in the realm of health and genomic data, and statistical computations are an excellent fit for efficient HE because they have very low depth. We demonstrated HE implementations and performance numbers for statistical computations in genomics such as the chi-square test, Cochran-Armitage Test for Trend, and Haplotype Estimation Maximization [33]. Next, we focused on string matching, using the Smith-Waterman algorithm for edit distance [17], another task which is frequently performed for genome sequencing and the study of genomic disease.

3.2 Heart Attack Risk

To demonstrate operations on health data, in 2013 we developed a live demo predicting the risk of having a heart attack based on six health characteristics [5]. We evaluated predictive models developed over decades in the Framingham Heart study, using the Cox proportional hazard method. I showed the demo live to news reporters at the 2014 AAAS meeting, and our software processed my risk for a heart attack in the cloud, operating on encrypted data, in a fraction of a second.

In 2016, we started a collaboration with Merck to demonstrate the feasibility of evaluating such models on large patient populations. Inspired by our published work on heart attack risk prediction [5], they used SEAL to demonstrate running the heart attack risk prediction on one million patients from an affiliated hospital. Their implementation returned the results for all patients in about 2 hours, compared to 10 minutes for the same computation on unencrypted patient data.

3.3 Cancer patient statistics

In 2017, we began a collaboration with Crayon, a Norwegian company that develops health record systems. The goal of this collaboration was to demonstrate the value of SEAL in a real world working environment. Crayon reproduced all computations in the 2016 Norwegian Cancer Report using SEAL and operating on encrypted inputs. The report processed the cancer statistics from all cancer patients in Norway collected over the last roughly 5 decades.

3.4 Genomic Privacy

A growing community of researchers in bioinformatics and biostatistics concerned with patient privacy issues invited me to give a tutorial on Homomorphic Encryption at the 2014 Biological Data Sciences meeting at Cold Spring Harbor Labratories. This interdisciplinary community was interested in the development of a range of cryptographic techniques to apply to privacy problems in the health and biological sciences arenas. A manual for using SEAL for common tasks in bioinformatics was published in their special issue [20]. One measure of the growth of this community over the last five years has been participation in the iDASH Secure Genome Analysis Competition, a series of annual international competitions funded by the National Institutes of Health (NIH) in the U.S.. The iDASH competition has included a track on Homomorphic Encryption for the last five years 2015–2019, and our team from MSR submitted winning solutions for the competition in 2015 ([31]) and 2016 ([11]). The tasks were: chi-square test, modified edit distance, database search, training logistic regression models, genotype imputation. Each year, roughly 5-10 teams from research groups around the world submitted solutions for the task, which were bench-marked by the iDASH team. These results provide the biological data science community and NIH with real and evolving measures of the performance and capability of Homomorphic Encryption to protect the privacy of genomic data sets while in use. Summaries of the competitions are published in [42, 44].

3.5 Machine Learning: training and prediction

The 2012 "ML Confidential" paper [27] was the first to propose *training* ML algorithms on homomorphically encrypted data and to show initial performance numbers for simple models such as linear means classifiers and gradient descent. Training is inherently challenging because of the large and unknown amount of data to be processed.

Prediction tasks on the other hand, process an input and model of known size, so many models and tasks can be processed efficiently. For example, in 2016 we developed a demo using SEAL to predict the flowering time for a flower. The model processed $200,000$ SNPs from the genome of the flower, and evaluated a Fast Linear Mixed Model (LMM). Including the round-trip communication time with the cloud running the demo as a service in Azure, the prediction was obtained in under a second.

Another demo developed in 2016 using SEAL predicted the mortality risk for pneumonia patients based on 46 characteristics from the patient's medical record. The model in this case is an example of an intelligible model and consists of 46 degree 4 polynomials to be evaluated on the patient's data. Data from $4,096$ patients can be batched together, and the prediction for all $4,096$ patients was returned by the cloud service in a few seconds (in 2016).

These two demos evaluated models which were represented by shallow circuits, linear in the first case and degree 4 in the second case. Other models such as deep neural nets (DNNs) are inherently more challenging because the circuits are so deep. To enable efficient solutions for such tasks requires a blend of cryptography and ML research, aimed at designing and testing ways to process data which allow for efficient operations on encrypted data while maintaining accuracy. An example of that was introduced in CryptoNets [26], showing that the activation function in the layers of the neural nets can be approximated with a low-degree polynomial function (x^2) without significant loss of accuracy.

The CryptoNets paper was the first to show the evaluation of a neural net predictions on encrypted data, and used the techniques introduced there to classify hand-written digits from the MNIST [35] data set. Many teams have since worked on improving the performance of CryptoNets, either with hybrid schemes or other optimizations [19, 29, 39]. In 2018, in collaboration with Median Technologies, we demonstrated deep neural net predictions for a medical image recognition task: classification of liver tumors based on medical images.

Returning to the challenge of training ML algorithms, the 2017 iDASH contest task required the teams to train a logistic regression model on encrypted data. The data set provided for the competition was very simple and did not require many iterations to train an effective model (the winning solution used only 7 iterations [30, 32]). The MSR solution [13] computed over 300 iterations and was fully scalable to any arbitrary number of iterations. We also applied our solution to a simplified version of the MNIST data set to demonstrate the performance numbers.

Performance numbers for all computations described here were published at the time of discovery. They would need to be updated now with the latest version of SEAL, or can be estimated. Hardware acceleration techniques using state-of-the-art FPGAs can be used to improve the performance further ([38]).

4 How do we assess security?

The security of all Homomorphic Encryption schemes described in this article is based on the mathematics of lattice-based cryptography, and the hardness of well-known lattice problems in high dimensions, problems which have been studied for more than 25 years. Compare this to the age of other public key systems such as RSA (1975) or Elliptic Curve Cryptography ECC (1985). Cryptographic applications of Lattice-based Cryptography were first proposed by Hoffstein, Pipher, and Silverman [28] in 1996 and led them to launch the company NTRU. New hard problems such as LWE were proposed in the period of 2004-2010, but were reduced to older problems which had been studied already for several decades: the Approximate Shortest Vector Problem (SVP) and Bounded Distance Decoding.

The best known algorithms for attacking the Shortest Vector Problem or the Closest Vector Problem are called lattice basis reduction algorithms, and they have a more than 30-year history, including the LLL algorithm [36]. LLL runs in polynomial

time, but only finds an exponentially bad approximation to the shortest vector. More recent improvements, such as BKZ 2.0 [15], involve exponential algorithms such as sieving and enumeration. Hard Lattice Challenges were created by TU Darmstadt and are publicly available online for anyone to try to attack and solve hard lattice problems of larger and larger size for the record.

Homomorphic Encryption scheme parameters are set such that the best known attacks take exponential time (exponential in the dimension of the lattice, n, meaning roughly 2^n time). These schemes have the advantage that there are no known polynomial time quantum attacks, which means they are good candidates for Post-Quantum Cryptography (PQC) in the ongoing 5-year NIST PQC competition.

Lattice-based cryptography is currently under consideration for standardization in the ongoing NIST PQC Post-Quantum Cryptography competition. Most Homomorphic Encryption deployments use small secrets as an optimization, so it is important to understand the concrete security when sampling the secret from a non-uniform, small distribution. There are numerous heuristics used to estimate the running time and quality of lattice reduction algorithms such as BKZ2.0. The Homomorphic Encryption Standard recommends parameters based on the heuristic running time of the best known attacks, as estimated in the online LWE Estimator [2].

5 Conclusion

Homomorphic Encryption is a technology which allows meaningful computation on encrypted data, and provides a tool to protect privacy of data in use. A primary application of Homomorphic Encryption is secure and confidential outsourced storage and computation in the cloud (i.e. a data center). A client encrypts their data locally, keeps their encryption key(s) locally, then uploads the encrypted data to the cloud for long-term storage and analysis. The cloud processes the encrypted data without decrypting it, and returns encrypted answers to the client for decryption. The cloud learns nothing about the data other than the size of the encrypted data and the size of the computation. The cloud can process Machine Learning or Artificial Intelligence (ML or AI) computations, either to make predictions based on known or private models or to train new models, while preserving the client's privacy.

Current solutions for HE are implemented in 5-6 major open source libraries world-wide. The Homomorphic Encryption Standard [1] for using HE securely was approved in 2018 by HomomorphicEncryption.org, an international consortium of researchers in industry, government, and academia.

Today, applied Homomorphic Encryption remains an exciting direction in cryptography research. Several big and small companies, government agencies and contractors, and academic research groups are enthusiastic about the possibilities of this technology. With new algorithmic improvements, new schemes, an improved understanding of concrete use-cases, and an active standardization effort, wide-scale deployment of Homomorphic Encryption seems possible within the next 2-5 years. Small-scale deployment is already happening.

Computational performance, memory overhead, and the limited set of operations available in most libraries remain the main challenges. Most Homomorphic Encryption schemes are inherently parallelizable, which is important to take advantage of to achieve good performance. Thus, easily parallelizable arithmetic computations seem to be the most amenable to Homomorphic Encryption at this time and it seems plausible that initial wide-scale deployment may be in applications of Machine Learning to enable Private AI.

6 Acknowledgements

I would like to gratefully acknowledge the contributions of many people in the achievements, software, demos, standards, assets and impact described in this article. First and foremost, none of this software or applications would exist without my collaborators on the SEAL team, including Kim Laine, Hao Chen, Radames Cruz, Wei Dai, Ran Gilad-Bachrach, Yongsoo Song, John Wernsing, with substantial contributions from interns Gizem Cetin, Kyoohyung Han, Zhicong Huang, Amir Jalali, Rachel Player, Peter Rindal, Yuhou Xia as well. The demos described here were developed largely by our partner engineering team in Foundry 99: Shabnam Erfani, Sreekanth Kannepalli, Steven Chith, James French, Hamed Khanpour, Tarun Singh, Jeremy Tieman. I launched the Homomorphic Encryption Standardization process in collaboration with Kim Laine from my team, with Roy Zimmermann and the support of Microsoft Outreach, and collaborators Kurt Rohloff, Vinod Vaikuntanathan, Shai Halevi, and Jung Hee Cheon, and collectively we now form the Steering Committee of HomomorphicEncryption.org. Finally I would like to thank the organizers of ICIAM 2019 for the invitation to speak and to write this article.

References

1. Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin E. Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018. https://eprint.iacr.org/2019/939.
2. Martin Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
3. Dan Boneh, Eujin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC'05: Proceedings of the Second international conference on Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
4. Joppe W Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.
5. Joppe W Bos, Kristin E. Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.

6. Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Chimera: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes. Cryptology ePrint Archive. https://eprint.iacr.org/2018/758.
7. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.
8. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.
9. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Oct 2011.
10. Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference Proceedings*, pages 505–524, August, 2011.
11. Gizem S. Cetin, Hao Chen, Kim Laine, Kristin E. Lauter, Peter Rindal, and Yuhou Xia. Private queries on encrypted genomic data. *BMC Medical Genomics*, 10(45), July 2017.
12. Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Jeffrey Hoffstein, Kristin E. Lauter, Satya Lokam, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Security of homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA, July 2017.
13. Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin E. Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC Medical Genomics*, 11(81), October 2018.
14. Hao Chen, Kristin E. Lauter and Katherine E. Stange Attacks on the Search RLWE Problem with Small Errors SIAM J. Appl. Algebra Geometry, Vol. 1 (2017), pages 665–682, Society for Industrial and Applied Mathematics.
15. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 1–20, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
16. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.
17. Jung Hee Cheon, Miran Kim, and Kristin E. Lauter. Homomorphic computation of edit distance. In *International Conference on Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.
18. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33:34–91, 2020.
19. Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 142–156. ACM, 2019.
20. Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.
21. Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.
22. Kirsten Eisentraeger, Sean Hallgren, and Kristin E. Lauter. Weak Instances of PLWE. In *Selected Areas in Cryptography-SAC 2014*, Lecture Notes in Computer Science, volume 8781, pages 183–194, Springer, 2014.
23. Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of Ring-LWE. In *Advances in Cryptology - CRYPTO 2015*, Lecture Notes in Computer Science, volume 9215, pages 63–92, Springer, 2015.
24. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012. https://eprint.iacr.org/2012/144 (accessed on 9 April, 2018).

25. Craig Gentry. *A fully homomorphic encryption scheme*. Stanford University, 2009.
26. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.
27. Thore Graepel, Kristin E. Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.
28. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: a ring-based public key cryptosystem. In *Algorithmic number theory (Portland, OR, 1998)*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
29. Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018.
30. Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. Logistic regression model training based on the approximate homomorphic encryption. Cryptology ePrint Archive, Report 2018/254, 2018. https://eprint.iacr.org/2018/254.
31. Miran Kim and Kristin E. Lauter. Private genome analysis through homomorphic encryption. *BMC medical informatics and decision making*, 15(Suppl 5):S3, 2015.
32. Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure logistic regression based on homomorphic encryption. Cryptology ePrint Archive, Report 2018/074, 2018. https://eprint.iacr.org/2018/074.
33. Kristin E. Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.
34. Kristin E. Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.
35. Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits, 1998. http://yann.lecun.com/exdb/mnist/.
36. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.
37. Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of STOC*, pages 1219–1234. IEEE Computer Society, 2012.
38. M Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai. Heax: High-performance architecture for computation on homomorphically encrypted data in the cloud. *arXiv preprint arXiv:1909.09731*, 2019.
39. M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. XONN: Xnor-based oblivious deep neural network inference. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1501–1518, Santa Clara, CA, August 2019. USENIX Association.
40. Ron Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
41. Microsoft SEAL (release 3.2). https://github.com/Microsoft/SEAL, November 2018. Microsoft Research, Redmond, WA.
42. Haixu Tang, Xiaoqian Jiang, Xiaofeng Wang, Shuang Wang, Heidi Sofia, Dov Fox, Kristin E. Lauter, Bradley Malin, Amalio Telenti, Li Xiong, and Lucila Ohno-Machado. Protecting genomic data analytics in the cloud: state of the art and opportunities. *BMC Medical Genomics*, 9(63), 2016.
43. Jonathan Vanian. 4 Big Takeaways from Satya Nadella's talk at Microsoft Build, 2018. https://fortune.com/2018/05/07/microsoft-satya-nadella-build/.
44. Shuang Wang, Xiaoqian Jiang, Haixu Tang, Xiaofeng Wang, Diyue Bu, Knox Carey, Stephanie OM Dyke, Dov Fox, Chao Jiang, Kristin E. Lauter, Bradley Malin, Heidi Sofia, Amalio Telenti, Lei Wang, Wenhao Wang, and Lucila Ohno-Machado. A community effort to

protect genomic data sharing, collaboration and outsourcing. *npj Genomic Medicine*, 2(33), 2017.