Kristin Lauter
Wei Dai
Kim Laine *Editors*

Protecting Privacy through Homomorphic Encryption

Springer

Protecting Privacy through Homomorphic Encryption

Kristin Lauter • Wei Dai • Kim Laine
Editors

Protecting Privacy through Homomorphic Encryption

Springer

Editors

Kristin Lauter
West Coast Research Science
Facebook AI Research
Seattle, WA, USA

Wei Dai
Cryptography and Privacy Research Group
Microsoft Research
Redmond, WA, USA

Kim Laine
Cryptography and Privacy Research Group
Microsoft Research
Redmond, WA, USA

ISBN 978-3-030-77286-4 ISBN 978-3-030-77287-1 (eBook)
https://doi.org/10.1007/978-3-030-77287-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book is concerned with explaining methods for protecting privacy using Homomorphic Encryption. Privacy means different things to different people. In this volume, we will use the term privacy to refer to the notion defined by some social scientists as the guarantee that an individual or an organization *should have the right to control how their data is used or shared*. Privacy is not possible without tools from cryptography necessary to protect the security of data from unauthorized access or use.

Encryption is a tool for protecting data by transforming it using mathematical methods and the knowledge of a cryptographic key. Assuming a sound implementation of an encryption scheme and the hardness of the underlying mathematical problems, encryption can be used to protect both the security and the privacy of data. Traditional encryption schemes such as the US government standardized AES block cipher can be used to protect data while in transit or in storage. But to protect data while in use requires a new kind of encryption which allows for meaningful computation on ciphertexts without decryption. Such encryption is called Homomorphic Encryption (HE), because homomorphic is a common term in mathematics meaning to preserve structure. It means that the encryption map preserves the underlying algebraic structure of the data, resulting in the same output if the order of encryption and computation are exchanged.

The existence of a solution for Homomorphic Encryption was an open problem for more than three decades. A partially homomorphic encryption scheme was known already in the mid-1970s: RSA encryption allows for one operation on ciphertexts. But computation on today's (classical) computers is implemented as operations on bits described as circuits of AND and OR gates. So, two operations on encrypted data are required to implement general circuits for computation. The first blueprint for a solution was introduced by [1] in 2009, including the notion of bootstrapping to allow for arbitrary computation. The lattice-based solutions used in all the homomorphic encryption libraries today implement schemes based on the Ring Learning with Errors (RLWE) problem, which will be further explained in Part II. The first RLWE-based solution [2] was later extended to [3], and other proposed schemes followed, which will all be explained in Parts I and II. The first practical approach to computation on real data was introduced in [4], including the encoding of integers and real data as ciphertexts, replacing bitwise encryption. This led for example to techniques introduced in [5] for the first time to perform machine learning tasks on encrypted data, such as training models and using them for prediction, and eventually to the CryptoNets project [6] which demonstrated neural net predictions on encrypted data.

Any new proposal for cryptosystems based on hard mathematical problems must be thoroughly studied and reviewed by the scientific community before the public can be expected to adopt and trust it to protect the privacy and security of their data. New cryptographic proposals have typically seen at least a 10-year lag before widespread adoption in the industry, as was the case for Elliptic Curve Cryptography. Lattice-based cryptography was first introduced in the mid-1990s. There are no known efficient quantum attacks on general lattice-based schemes, so lattice-based key exchange and signature schemes are currently leading candidates in the ongoing 5-year National Institute of Standards and Technology (NIST) Post-Quantum Cryptography Standardization competition. But the parameters required for Homomorphic Encryption applications are quite a bit larger than for key exchange and signature schemes, and the protocols and applications are quite different. The idea of forming a community to standardize Homomorphic Encryption came out of a meeting between Kristin Lauter, Shai Halevi, Kurt Rohloff, Yuriy Polyakov, and Victor Shoup in New York City in April, 2015. Initial goals included developing common APIs to ensure interoperability of different implementations.

In 2017, Microsoft Research (MSR) Outreach funded the first Homomorphic Encryption Standardization Workshop, hosted at Microsoft in Redmond, WA, on July 13–14, 2017. The workshop was co-organized by Kristin Lauter and Kim Laine from MSR, Roy Zimmermann from MSR Outreach, Lily Chen (NIST), Jung Hee Cheon (Seoul National University), Kurt Rohloff (NJIT/Duality), and Vinod Vaikuntanathan (MIT), with input from Shai Halevi (IBM/Algorand). This group now forms the Steering Committee for the Homomorphic Encryption.org open community which grew out of this meeting. This first workshop was organized as a collaboration meeting, with 36 invited participants divided into three working groups of 12. The groups were led by the workshop organizers, to work on writing three whitepapers on Security, API design, and Applications over the course of 2 days. The whitepapers were made available publicly several weeks after the workshop, after some additional work and editing. The papers were posted on the workshop webpage and on the Homomorphic Encryption.org website, which was set up along with email lists and discussion groups to continue the conversation on standardization of HE.

First Homomorphic Encryption Standardization Workshop July 13–14, 2017, Microsoft Research, Redmond WA, USA

After the first workshop, it was decided that the Security whitepaper could form the basis of the first Homomorphic Encryption Standard, to assure basic agreement on secure parameter sets to be used for applications. Input from the wider international research community was solicited, and the revised Security whitepaper was circulated. After some updates, it was approved by the community at the Second Homomorphic Encryption Standardization Workshop, on March 15–16, 2018, at MIT, Cambridge MA, garnering more than 65 signatures from workshop attendees. The first two workshops both featured presentations of homomorphic encryption software by developers from all the leading HE libraries worldwide. The second workshop also included some research talks and panel discussions on the path forward for standardizing common APIs and Applications.

Second Homomorphic Encryption Standardization Workshop March 15–16, 2018, MIT, Cambridge MA, USA

Following further expert input from the community and the addition of some co-authors, the final version of the first Homomorphic Encryption Standard [7] was officially approved at the Third Homomorphic Encryption Standardization Workshop at the University of Toronto, October 20, 2018. The [7] Standard was posted online on the HomomorphicEncryption.org website and on the IACR eprint archive and appears here as Part II of this volume. The third workshop was co-located with the 25th ACM Conference on Computer and Communications Security (CCS) and the affiliated Workshop on Applied Homomorphic Cryptography (WAHC) and featured a poster session for related results. Stated goals were to build upon the API discussion from the second workshop and to present a draft API standard. The third workshop also included presentations from American and Canadian government agencies, including the Canadian Security Establishment (CSE), NIST, and the National Science Foundation (NSF). The second and third workshops also included reports on the Homomorphic Encryption track of the Annual iDASH Secure Genome Analysis Competition, co-funded by the National Institutes of Health (NIH).

Third Homomorphic Encryption Standardization Workshop October 20, 2018, Univ. of Toronto, Toronto, Canada

What started as a largely academic community of experts has grown to include many researchers and developers from industry. The Fourth Homomorphic Encryption Standardization Workshop was hosted by Intel in Santa Clara, CA, on August 17, 2019, co-located with the USENIX Security 2019 conference. In addition to sponsorship from Microsoft, Intel, Duality, and Samsung, the workshops have included participants, panelists, or organizers from IBM, Galois, SAP, Google, Intuit, Inpher, CryptoExperts, and CryptoLabs. The fourth workshop focused on introducing scheme-specific white papers and discussing protocol standardization for applications.

Homomorphic Encryption Standardization Workshop, August 17, 2019, Santa Clara CA, USA

The next two Homomorphic Encryption Standardization Workshops had already been planned: one for May 7—8, 2020, in Geneva, Switzerland, co-hosted by EPFL, Inpher and ITU and co-located with the UN AI for Good conference at the Geneva International Conference Centre; the second one was planned for December 2020 in Seoul, co-hosted by Seoul National University and Samsung and co-located with AsiaCrypt 2020. Both events had to be postponed due to the global pandemic.

In an effort to train more PhD students to work on and do research on HE, Microsoft Research hosted a Private AI Bootcamp in December 2019. More than 100 students and a few postdoctoral researchers applied, and more than 30 participants were supported to attend the workshop and work in 6 teams to develop novel privacy-preserving applications of Homomorphic Encryption. The six whitepapers written by the six teams are published here as Part IV of this volume.

Private AI Bootcamp – Microsoft Research, December 2–4, 2019

In February 2020, Microsoft Research again hosted a Strategic Planning meeting in Redmond to accelerate progress towards documenting schemes and specifying application protocols. Part I of this volume was written by the participants of the Schemes track at the February 2020 workshop. It contains an introduction to Homomorphic Encryption and descriptions of the main HE schemes in widespread use today. Part III of this volume was written by participants in the Applications track at the workshop and contains four whitepapers describing protocols for applications of HE, including data sharing, network traffic monitoring, private set intersection, and a trusted monitoring service. Parts III and IV should be of broad interest across many industries, as they contain 10 chapters presenting novel ways to protect privacy in applications using Homomorphic Encryption.

Homomorphic Encryption Strategic Planning Workshop February 6–7, 2020

The current volume is the result of these six workshops and the ongoing work of the HomomorphicEncryption.org community. The editors would like to thank all the organizers, participants, authors, and community members who have helped to make this volume possible through their contributions. We hope that the volume will serve as an accessible introduction to HE, providing guidance on how to use HE to preserve privacy in numerous ways.

References

1. Craig Gentry. A fully homomorphic encryption scheme. Thesis, Stanford University, 2009.
2.. 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.
3. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Proc. of ITCS, pages 309–325. ACM, 2012.
4. Kristin 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.

5. Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In International Conference on Information Security and Cryptology, pages 1–21. Springer, 2012.
6. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin 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.
7. Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, Vinod Vaikuntanathan, Homomorphic Encryption Standard, November 21, 2018. https://eprint.iacr.org/2019/939.pdf Published as Part 2 of this volume.

Abstract

With the explosion of the Internet and AI technologies, privacy protection has become a critical problem in society today. We need to inject our technologies with responsible measures for protecting privacy to better serve individuals. New legislation impedes collaboration between companies and governments even with the best intentions. Homomorphic encryption is one of the leading candidates for building privacy-preserving services. It allows processing protected data without access to the raw data. For example, a patient's health record or diagnostic images can be analyzed in encrypted form without decryption, and the result is only readable by the patient. To most people and policymakers, homomorphic encryption still sounds magical and impractical.

This book summarizes recent inventions, provides guidelines and recommendations, and demonstrates many practical applications of homomorphic encryption. This collection of papers represents the combined wisdom of the community of leading experts on Homomorphic Encryption. In the past 3 years, a global community consisting of researchers in academia, industry, and government has been working closely to standardize homomorphic encryption. This is the first publication of whitepapers created by these experts that comprehensively describes the scientific inventions, presents a concrete security analysis, and broadly discusses applicable use scenarios and markets. This book also features a collection of privacy-preserving machine learning applications powered by homomorphic encryption designed by groups of top graduate students worldwide at the Private AI Bootcamp hosted by Microsoft Research.

The book aims to connect non-expert readers with this important new cryptographic technology in an accessible and actionable way. Readers who have heard good things about homomorphic encryption but are not familiar with the details will find this book full of inspiration. Readers who have preconceived biases based on out-of-date knowledge will see the recent progress made by industrial and academic pioneers on optimizing and standardizing this technology. A clear picture of how homomorphic encryption works, how to use it to solve real-world problems, and how to efficiently strengthen privacy protection will naturally become clear.

Contents

Part I Introduction to Homomorphic Encryption

Part II Homomorphic Encryption Security Standard

Part III Applications of Homomorphic Encryption

Part IV Applications of Homomorphic Encryption

Part I
Introduction to Homomorphic Encryption

Introduction to Homomorphic Encryption and Schemes

Jung Hee Cheon, Anamaria Costache, Radames Cruz Moreno, Wei Dai, Nicolas Gama, Mariya Georgieva, Shai Halevi, Miran Kim, Sunwoong Kim, Kim Laine, Yuriy Polyakov, and Yongsoo Song

1 Introduction to Homomorphic Encryption

Homomorphic encryption (HE) enables processing encrypted data without decrypting it. This technology can be used, for example, to allow a public cloud to operate on secret data without the cloud learning anything about the data. Simply encrypt the secret data with homomorphic encryption before sending it to the cloud, have the

The original version of this chapter was revised: Revised Chapter 1 has been uploaded to Springerlink. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-77287-1_15

J. H. Cheon (✉)
Seoul National University, Seoul, Republic of Korea
e-mail: jhcheon@snu.ac.kr

A. Costache
Norwegian University of Science and Technology, Trondheim, Norway; Intel AI Research, San Diego, CA, USA

R. C. Moreno
Microsoft Research, Redmond, WA, USA
e-mail: radames.cruz@microsoft.com

W. Dai · K. Laine
Cryptography and Privacy Research Group, Microsoft Research, Redmond, WA, USA
e-mail: wei.dai@microsoft.com; kim.laine@microsoft.com

N. Gama · M. Georgieva
Inpher, Lausanne, Switzerland
e-mail: nicolas@inpher.io; mariya@inpher.io

S. Halevi
Algorand Foundation, Yorktown Heights, NY, USA
e-mail: shaih@alum.mit.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_1

cloud process the encrypted data and return the encrypted result, and finally decrypt the encrypted result. Here is a simplistic “hello world” example using homomorphic encryption:

```
# Every encryption needs a secret key. Let’s get one of those.
myEncryptionKey = generateEncryptionKey()

# Now we can encrypt some very secret data.
encrypted5  = encrypt(myEncryptionKey,  5)
encrypted12 = encrypt(myEncryptionKey, 12)
excrypted2  = encrypt(myEncryptionKey,  2)
# We have three ciphertexts now.

# We want the sum of the first two.
# Luckily we used homomorphic encryption, so we can actually
do this.
encrypted17 = addCiphertexts(encrypted5, encrypted12)

# Maybe we want to multiply the result by the 3rd ciphertext.
encrypted34 = multiplyCiphertexts(encrypted17, encrypted2)

# See that? We operated on ciphertexts without needing the key.

# But no matter what we compute, the result is always encrypted.
# To actually see the final result, we have to use the key.
decrypted34 = decrypt(myEncryptionKey, encrypted34)
print(decrypted34) # This should print ’34’
```

Homomorphic encryption today falls into the following common categories: partially homomorphic (weakest notion), leveled fully homomorphic, and fully homomorphic encryption (strongest notion). Partially homomorphic encryption supports only one type of operation, e.g. addition or multiplication. Leveled fully homomorphic encryption supports more than one operation but only computations of a predetermined size (typically multiplicative depth). Fully homomorphic encryption (FHE) supports arbitrary computation on encrypted data and is the strongest notion of homomorphic encryption.

M. Kim
Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea
e-mail: mirankim@unist.ac.kr

S. Kim
University of Washington Bothell, Bothell, WA, USA
e-mail: sunwoong@uw.edu

Y. Polyakov
Duality Technologies, Newark, NJ, USA
e-mail: polyakov@njit.edu

Y. Song
Seoul National University, Seoul, Korea
e-mail: y.song@snu.ac.kr

1.1 Plaintexts and Operations

Computation on encrypted data in homomorphic encryption preserves the same computation on the underlying plaintext. In the example above, we encrypted integers and then added and multiplied them. There are other types of data that we may want to encrypt and other operations that we may want to perform. For example:

- Encrypt bits and perform logical AND, OR, XOR operations on the ciphertexts.
  $0 \, AND \, 1 \rightarrow 0, 0 \, OR \, 1 \rightarrow 1, 1 \, XOR \, 1 \rightarrow 0$
- Encrypt small integers and perform addition and multiplication, as long as the result does not exceed some fixed bound, for instance, if the bound is 10,000
  $123 + 456 \rightarrow 579$, $12 \times 432 \rightarrow 5184$, $35 \times 537 \rightarrow$ *overflow*
- Encrypt 8-bit unsigned integers (between 0 and 255) and perform addition and multiplication modulo 256
  $128 + 128 \rightarrow 0$, $2 \times 129 \rightarrow 2$
- Encrypt fixed-point numbers and perform addition and multiplication with the result rounded to a fixed precision, for instance, two digits after the decimal point
  $12 + 42 + 1.34 \rightarrow 13.76$, $2.23 \times 5.19 \rightarrow 11.57$

Different homomorphic encryption schemes support different plaintext types and different operations on them.

1.2 Vectors and Special-Purpose Plaintext Data Types

Some homomorphic encryption schemes, such as BGV, BFV, and CKKS, support "packing" – or "batching" – many plaintexts into a single ciphertext. They encrypt vectors of elements, perform element-wise operations, and move elements around in the vector:

- Encrypt vectors of any of the above types and perform operations element-wise

$$(1, 0, 1, 0) \;\; AND \;\; (1, 0, 0, 1) \rightarrow (1, 0, 0, 0)$$

$$(1, 2) \times (3, 4) \rightarrow (3, 8)$$

$$(1.1, 2.2) + (5.5, 6.6) \rightarrow (6.6, 8.8)$$

- and rotation on element's positions

$$rotLeft1 \; (\; (1, 2, 3, 4) \;) \rightarrow (2, 3, 4, 1)$$

These element-wise and data-movement operations are often called *SIMD operations* (single-instruction multiple-data).

Many homomorphic encryption schemes also support various special-purpose plaintext data types. While not described in this document, we briefly list some of them below.

- Multi-precisions integers modulo a very large integer of the form $p^n + 1$;
- Vectors over finite fields (e.g. useful for evaluation of the AES cipher);
- Polynomials modulo $X^n + 1$(e.g. for convolution products).

Some of the most promising homomorphic encryption schemes today, such as BFV, BGV, CKKS, DM, and CGGI, are implemented in open-source libraries. All these schemes have unique advantages and drawbacks depending on the types of computation one wants to perform.

1.3 Ciphertexts

One thing that all contemporary homomorphic encryption schemes have in common is that in all of them each ciphertext is an *array of integers* of fairly high dimension (at least a few hundred integers, and sometimes many thousands).

1.4 Symmetric vs. Public-Key Homomorphic Encryption

In the example code from above we used the same key for encryption and decryption; this type of encryption is called *symmetric encryption*. In contrast, public-key encryption (also called asymmetric encryption), uses two different keys: a secret key for decryption and a public key associated to the secret key for encryption.

Homomorphic encryption can be instantiated as either symmetric encryption or public-key encryption, with encrypted computation capabilities. It provides the following fundamental operations:

Operation	Symmetric encryption	Public-key encryption
Key generation	secret key	secret key → public keys
Encryption	plaintext, secret key → ciphertext	plaintext, public key → ciphertext
Decryption	ciphertext, secret key → plaintext	ciphertext, secret key → plaintext
Operations	ciphertext (and plaintext) → ciphertext	

1.5 Parameters and Security

Instantiating any encryption scheme – homomorphic or otherwise – requires setting some parameters, for example, to determine the key size or the security level. For homomorphic encryption, the parameters influence not just security but also the

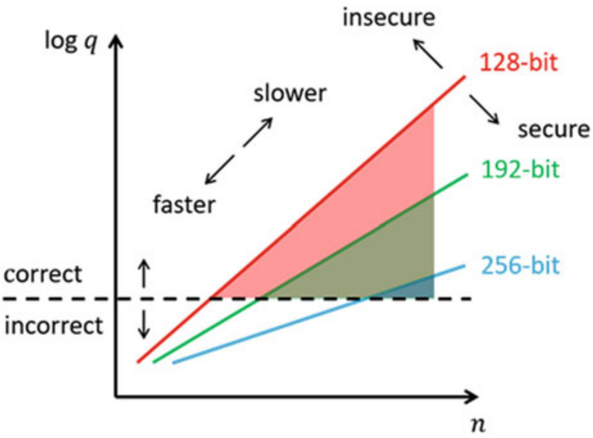


Fig. 1 Parameter selection

plaintext type and the computations that can be performed. The most prominent parameters that must be set for contemporary HE schemes are the following:

- Ciphertext dimension n corresponds roughly to the number of integers in each ciphertext;
- Ciphertext modulus q bounds the size of each integer in the ciphertext array.

In general, the security level *increases as n grows and decreases as q grows*. On the other hand, the larger q is, the more complex computations can be performed on ciphertexts of the encryption scheme: Ciphertexts in these encryption schemes contain a *noise component* (which is important for security), and that noise grows with each operation. The encrypted result can only be decrypted if the noise is smaller than q, hence using larger values of q imply that we can do more operations.

An illustration of the parameters n and q and their influence on the level of security is sketched in Fig. 1.

There are a few other parameters that influence security of lattice-based HE schemes, such as the distribution from which the secret key is selected (and a few others).

The security of lattice-based HE schemes is based on the hardness of a mathematical problem called *Learning with Errors* (LWE) or a variant of it called *Ring Learning-with-Errors* (RLWE). The (R)LWE problem is believed to be hard for both classical and quantum computers under appropriate parameters. The HE security document[1] contains tables indicating the security levels for various choices of n and q. Those tables are based on the best-known attacks against LWE.

The tables in the HE security document should be used as follows: Once the size of q is known (as well as the secret-key distribution), one needs to consult the

[1] Published in 2017 on http://homomorphicencryption.org/white_papers/security_homomorphic_encryption_white_paper.pdf. An updated version is included in Chapter "Homomorphic Encryption Standard".

table and find the smallest value of n that provides the desired security level for this q-size. For example, suppose we use a ternary secret-key distribution and want to achieve 128 bits of security with a 200-bit modulus q. The third part of Table 1 in the security document says that a value of $n = 8192$ can support q moduli of size up to 218 bits, but $n = 4096$ can only support q moduli of size up to 109 bits. Hence the smallest n that we can use is $n = 8192$.

2 The BGV and BFV Encryption Schemes

This section includes a simplified introduction to the *Brakerski-Gentry-Vaikuntanathan* (BGV) encryption scheme [34] and the encryption scheme due to *Brakerski and Fan-Vercauteren* (BFV) [3, 6]. For a more technical description of the schemes, we refer the reader to the Further Information subsection below.

BGV and BFV are homomorphic encryption schemes whose security is based on the hardness of the *Ring Learning with Errors* (RLWE) problem. The plaintext type in both schemes consists of vectors of integers, with modular SIMD operations as described below.

The BGV and BFV schemes involve several parameters that determine the security level, functionality, and the plaintext data type supported by the scheme. These parameters are:

- Plaintext modulus p;
- Ciphertext modulus q;
- Ciphertext dimension n.

The plaintext modulus $p$determines an upper bound for the integer components of the *plaintext vectors* that are encrypted in the BGV and BFV schemes. For example, setting the plaintext modulus to $p = 31$ means that computing the product of an encrypted 5 and an encrypted 7will overflow and produce the result $5 \times 7 - 31 = 4$.

The ciphertext modulus q is the main functional parameter that determines the encrypted computation capabilities of the scheme. A ciphertext in the BGV or BFV scheme consists of an array of $2n$ integers between 0 and $q - 1$. As explained in the introduction, the larger the parameter q of an instance is, the more operations can be performed on encrypted data in that instance.

For a given value of q, the ciphertext dimension n determines the security level of the scheme, with larger n meaning higher security. At the same time, the ciphertext dimension n also influences the size of the *plaintext vector* which is encrypted into each ciphertext. Often – but not always – the size of the plaintext vector is equal to n.

2.1 Homomorphic Operations

Operations over encrypted data preserve the same operations modulo p on vectors of integers and always produce a ciphertext as output. The main operations are:

Two-Argument Operations

- Ciphertext-Ciphertext addition;
- Ciphertext-Plaintext addition;
- Ciphertext-Ciphertext multiplication;
- Ciphertext-Plaintext multiplication;
- Ciphertext-Ciphertext subtraction;
- Ciphertext-Plaintext subtraction.

Unary Operations

- Negation;
- Vector rotation.[2]

2.2 *Parameter Selection*

Typically the first parameter to select is the plaintext modulus p, that determines the width of the plaintext data type. In some applications the plaintext modulus needs to be large enough to accommodate the desired computation without overflow, other times an overflow is desired. Selecting appropriate plaintext modulus depends on details of the application, and is beyond the scope for this document.

The next parameter to select is ciphertext modulus q, which is primarily determined by the multiplicative depth of the desired encrypted computation; a higher depth requires a larger ciphertext modulus, and is typically slower. Therefore, the computation should be made as low depth as possible. For example, computing a product of four encrypted numbers A, B, C, and D is better done as $(A * B) * (C * D)$ rather than $A * (B * (C * D))$, as the former has lower multiplicative depth, and hence requires a smaller ciphertext modulus.

Once q is determined, the ciphertext dimension n should be selected to achieve the desired security level, using the tables in [9]. The application developer is advised to use a library that implements the [9] standard. We note that choosing the right table from the [9] document requires knowing certain details of the implementation, such as the secret key distribution.

2.3 *A BGV/BFV Hello World Example*

```
# We first must set the parameters p,q, and n
p = 31
q = 65537
```

[2] In some cases we have more involved data-movement operations than just rotations. See the Further Information section for more details.

```
n = 16
# Warning: this setting is completely insecure!!
# To get any kind of security with q=65537 we need at least n=512

# With these parameters, the size of the plaintext vectors is 8

# Generate the keys for these parameters
myPublicKey, mySecretKey = generateBFVkey(n, p, q)

# Encrypt data, each plaintext is a vector of 8 elements
encrypted_a  = encrypt(myPublicKey, [5, 11, 2, 0, 20, 3, 8, 11])
encrypted_b = encrypt(myPublicKey, [12, 7, 14, 11, 1, 2, 3, 24])
excrypted_c  = encrypt(myPublicKey, [2, 10, 15, 13, 6, 3, 2, 1])
# We have three ciphertexts now.

# Compute the sum of the first two.
encrypted_d = addCiphertexts(myPublicKey, encrypted_a,
encrypted_b)
# Encryption of vector [17, 18, 16, 11, 21, 5, 11, 4]

# Maybe we want to multiply the result by the 3rd ciphertext.
encrypted_e = multiplyCiphertexts(myPublicKey, encrypted_c,
                                  encrypted_d)
# Encryption of vector [3, 25, 23, 19, 2, 15, 22, 4]

# Then rotate by 2 to the right
encrypted_f = rotateBy2(myPublicKey, encrypted_e)

# To actually see the final result we have to use the key.
decrypted = decrypt(mySecretKey, encrypted_f)
print(decrypted)
# This should print [22, 4, 3, 25, 23, 19, 2, 15]
```

2.4 Further Information

Maintenance Operations

The BGV and BFV schemes also include some operations that have no effect on the underlying plaintext, but are nonetheless sometimes needed for implementation reasons.

- Ciphertext-Ciphertext multiplications and cyclic vector rotations have a side-effect of requiring a different secret key to decrypt the result than what was needed before the operation. These operations are therefore followed by a *key switching* operation to restore the secret key back to the original one.[3] The

[3] In some applications, key switching operations are avoided or delayed for the sake of optimization.

key switching operation for Ciphertext-Ciphertext multiplication is also called *relinearization*;

- Bootstrapping, which "refreshes" a ciphertext and reduces the level of noise in it, to support more computations. This operation is very expensive, and hence it is not often used (and sometimes it is not even implemented).
- Modulus switching, which sometimes follows the multiplication operation. This is used more in BGV, where it is needed to control the level of noise in a ciphertext. (It is rarely used in BFV, except for bootstrapping.)

Evaluation Keys

The key switching operations require the evaluator to have access to special public *evaluation keys*. These evaluation keys are generated by the owner of the secret key. In the context of Ciphertext-Ciphertext multiplication, these keys are often called *relinearization keys*; and in the context of rotation, they are sometimes called *rotation* or *Galois keys*.

Data Encoding

Prior to encrypting data with the BGV or BFV scheme, a separate *encoding* operation is required, which transforms source data (e.g. vectors of integers) into a native plaintext format for the scheme. After decryption, a corresponding *decoding* operation is required.

Data Movement Operations

For some setting of the parameters p and n, the native data-movement operations supported by the scheme may differ from just cyclic rotations. For example, in some cases the plaintext elements are arranged in a matrix with 2 rows and $n/2$ columns, with native operations of row-rotate and column-rotate.[4] Even in these cases, it is always possible to implement cyclic rotations using the native row- and column-rotations, as described in [11].

References for the BFV Encryption Scheme

The BFV scheme is a ring variant of the scale-invariant LWE scheme proposed by Brakerski [3]. The "textbook" (multi-precision integer arithmetic) variant of BFV is described in [6, 13]. Note that [13] provides tighter noise constraints than the original paper [6].

[4] For some parameters we get even higher-dimension hypercube formats.

The most efficient variants of BFV used in practice represent large integers in the Residue Number System (RNS). The RNS representation has a number of practical advantages over the conventional multi-precision positional number system (PNS) representation:

1. RNS works with native (machine-word size) integers: faster (up to 5–10x) than PNS.
2. Runtime in RNS scales (quasi-)linearly with integer size.
3. RNS dramatically improves memory locality.
4. Computations are easily parallelizable, and hence RNS supports efficient GPU/FPGA hardware implementations.

Two RNS variants of BFV are known in literature: [2] (based on integer arithmetic) and [10] (based on both integer and floating-point arithmetic). A comparison of the RNS variants is provided in [4].

The encoding of vectors of integers into a BFV plaintext is described in Appendix A of [13]. This batching/packing encoding technique is discussed at a more advanced level in [7].

The bootstrapping for BFV is described in [5]. Note that BFV bootstrapping is rarely used in practice, and is not currently supported by any open-source homomorphic encryption library.

The following libraries have open-source implementations of BFV (the variants are indicated in parentheses):

- Microsoft SEAL [2]
- PALISADE [2, 6, 10]
- Lattigo [14]

References for the BGV Encryption Scheme

The BGV encryption scheme was first described in [34], improving on a previous construction from [35]. Here too it is desirable to represent large integers in the Residue Number System (RNS), for the same reason as for the BFV encryption scheme. This implementation was described in [7]. The BGV encryption scheme is implemented in the HElib and PALISADE libraries. Bootstrapping for BGV was described in [1, 8, 12], and is implemented in HElib.

3 The CKKS Encryption Scheme

This section includes a simplified introduction to the *Cheon, Kim, Kim, and Song (CKKS)* encryption scheme [15]. For a more technical description of the scheme, we refer the reader to the Further Information section below.

CKKS is a homomorphic encryption scheme whose security relies on the hardness of the *Ring Learning with Errors* (RLWE) problem. The plaintexts are vectors of real numbers, represented as a fixed-point type. The scheme natively supports fixed-point arithmetic between these vectors in a SIMD manner.

The CKKS scheme involves several parameters that determine the security level, functionality, and precision supported by the scheme. These parameters are:

- Number of fractional bits f, corresponding to the accuracy of the computation;
- (Maximal) Ciphertext modulus q;
- Ciphertext dimension n.

We assume that every plaintext value is represented as a binary fixed-point number which has f fractional bits after the radix point. The value of f for a ciphertext can be adjusted after performing computations using a so-called *rescaling* procedure, which is a distinctive feature of CKKS.

The ciphertext modulus q is the main functional parameter that determines the encrypted computation capabilities of the scheme. A ciphertext of the CKKS scheme consists of an array of $2n$ integers modulo q. The larger the parameter q is, the more operations can be performed on encrypted data and at a higher precision. For a given value of q, the ciphertext dimension n determines the security level of the scheme, with larger n meaning higher security. We refer the reader to the parameter selection section for more details.

As noted above, CKKS allows us to encrypt multiple fixed-point numbers in a single ciphertext. The ciphertext dimension n also determines the size of the plaintext vectors, which is $n/2$.

3.1 Homomorphic Operations

All computations involving at least one encrypted input produce encrypted outputs. The main operations are:

Two-Argument Operations

- Ciphertext-Ciphertext addition;
- Ciphertext-Plaintext addition;
- Ciphertext-Ciphertext multiplication;
- Ciphertext-Plaintext multiplication;
- Ciphertext-Ciphertext subtraction;
- Ciphertext-Plaintext subtraction.

Ciphertext-Ciphertext and Ciphertext-Plaintext multiplications return a ciphertext whose scaling factor is explicitly the product of scaling factors of inputs.

Ciphertext-Ciphertext and Ciphertext-Plaintext additions require the scaling factors of the inputs to match.

Unary Operations

- Negation;
- Cyclic vector rotation;
- Rescaling.

Rescaling, which almost always follows the multiplication operation, is a unary operation that divides the scaling factor of input ciphertext by a specific factor. It controls the magnitude of scaling factors during homomorphic computation. Ciphertext modulus decreases after the rescaling operation, and further multiplication is not allowed if a ciphertext modulus is too small.

3.2 Parameter Selection

The number of fractional bits and the supported depth of the encryption scheme are main parameters to be considered. Encrypted evaluation of a circuit can be performed if the circuit depth does not exceed the bound determined by the parameters.

Precision loss and overflow are two major issues of fixed-point arithmetic. Ciphertexts in CKKS have inherent error after encryption or computation, which is controlled by the parameter f. A larger f means more accurate result, but the computational cost grows as f increases. At the same time, the magnitude of encrypted values must be kept sufficiently smaller than the ciphertext modulus q to ensure that no overflow occurs during computation.

The maximal ciphertext modulus q is primarily determined by the multiplicative depth of the desired circuit to be evaluated, and by the accuracy parameter f; higher depth and larger accuracy require a larger ciphertext modulus, and is typically slower. Therefore, a common optimization technique is to represent a computational task as a circuit with minimal depth. For example, computing a product of four encrypted numbers A, B, C, and D is better done as $(A * B) * (C * D)$ rather than $A * (B * (C * D))$, as the former has lower multiplicative depth, and hence requires a smaller ciphertext modulus.

Once q is determined, a lower bound on the ciphertext dimension n is now determined to achieve a desired security level, using the tables in [9]. The application developer is advised to use a library that implements the [9] standard and automatically selects the correct table, as choosing the right table requires knowing certain details of the implementation, such as the secret key distribution.

3.3 A CKKS Hello World Example

```
# We first must set the parameters f,q, and n
f = 2
q = 65537
n = 8
# We use a decimal representation with f = 2 fractional digits

# Warning: this setting is completely insecure!!
# To get any kind of security with q=65537 we need at least n=512
# With these parameters, the plaintext vectors have size 4

# Generate the keys for these parameters
myPublicKey, mySecretKey = generateCKKSkey(n, q)

# Encrypt data, each plaintext is a vector of 4 elements
encrypted_a  = encrypt(myPublicKey, [1.53, -11.53, 0.02, -3.32])
encrypted_b = encrypt(myPublicKey, [12.29, 7.52, -14.47, 11.01])
excrypted_c  = encrypt(myPublicKey, [2.64, 10.78, -15.30, 13.34])
# We have three ciphertexts now.

# We want the sum of the first two.
# Luckily we used homomorphic encryption, so we can actually.
 do this.
encrypted_d = addCiphertexts(myPublicKey,
 encrypted_a, encrypted_b)
# encrypting the vector [13.82, -4.01, -14.45, 7.69]

# Maybe we want to multiply the result by the 3rd ciphertext.
encrypted_e = multiplyCiphertexts(myPublicKey, encrypted_c,
                                  encrypted_d)
# encrypting the vector [36.48, -43.23, 221.09, 102.58]

# Then rotate by 2 to the right
encrypted_f = rotateBy2(myPublicKey, encrypted_e)

# To actually see the final result, we have to use the key.
decrypted = decrypt(mySecretKey, encrypted_f)
print(decrypted)
# This should print [221.09, 102.58, 36.48, -43.23]
```

3.4 Further Information

Data Encoding

Prior to encrypting data with the CKKS scheme, a separate *encoding* operation is required. The CKKS encoding incurs some loss of precision, hence the plaintext vector must first be multiplied by a scaling factor (which is determined by the parameters of the scheme), to ensure that the encoded value retains enough precision. Then, the scaled vector is converted into a native plaintext format for the scheme. Ciphertexts implicitly store the scaling factor which may change during homomorphic computation. After decryption, a corresponding *decoding* operation is required.

The ciphertext modulus determines an upper bound for the components of the underlying *encoded plaintext* to guarantee its correct decryption. For example, setting the ciphertext modulus to $q = 1024$ means that an encryption of 12.34 with scaling factor of 32 is correctly decryptable but encrypting the same value with scaling factor 256 will result in overflow.

Maintenance Operations

The CKKS scheme also uses some operations that do not change the underlying plaintext (beyond some possible precision loss) but are nonetheless needed for implementation reasons.

- Ciphertext-Ciphertext multiplication and cyclic vector rotation have a side-effect of requiring a different secret-key to decrypt the result than what was needed before the operation. These operations are therefore followed by a *key switching* operation to convert the secret key back to the original one. The key switching operation for Ciphertext-Ciphertext multiplication is also called *relinearization*.
- *Bootstrapping*, which "refreshes" a ciphertext and raises the ciphertext modulus in it, to support more computations. This operation is expensive, and hence it is not often used (and sometimes it is not even implemented).

Evaluation Keys

The key switching operations require the evaluator to have access to special public *evaluation keys*. The evaluation key generation must be done by the secret key owner. In the context of Ciphertext-Ciphertext multiplication, these keys are often called *relinearization keys*; and in the context of rotation, they are sometimes called *rotation* or *Galois keys*. The bootstrapping procedure also requires such evaluation keys.

References for the CKKS Scheme

The CKKS encryption scheme was first proposed in [15]. For the same reason as for the BFV scheme, it is desirable to represent large integers in the RNS. Several RNS variants of the CKKS scheme have been proposed and implemented, including [17, 19, 20], and [21]. Modern HE libraries typically implement a combination of these RNS variants and often add their own optimizations/usability improvements. Bootstrapping for CKKS was described in [16, 18, 21].

Reference Implementations

The following libraries have open-source implementations of CKKS (the variants are indicated in parentheses):

- HEAAN/RNS-HEAAN
- HElib
- Lattigo
- Microsoft SEAL
- PALISADE

4 The DM (FHEW) and CGGI (TFHE) Schemes

4.1 Basic Concepts

This section includes a simplified introduction to the *Ducas-Micciancio* (*DM)* and *Chillotti-Gama-Georgieva-Izabachene CGGI schemes, based on* [25–29, 31, 32]. The DM scheme is often referred to as the FHEW scheme in literature, and the CGGI scheme is often referred to as the TFHE scheme. To distinguish the underlying schemes from their implementations in the FHEW and TFHE libraries, we adopt the naming convention based on authors' initials. For a more technical description of these schemes, we refer the reader to the Further Information subsection below.

DM and CGGI are homomorphic encryption schemes based on the *Learning with Errors* (LWE) problem and its ring variant, the *Ring Learning with Errors* (RLWE) problem. Common use-cases for these schemes are the encrypted evaluation of decision diagrams, comparisons, lookup tables and circuits.

The schemes can be used in two different modes: simple (automated) and advanced (manual). The simple mode automatically performs bootstrapping after each gate operation, providing the ability to evaluate arbitrary (typically Boolean) circuits. The simple mode is easy to configure (requires only one parameter) but can be less efficient than the advanced mode, especially when the circuit is known in advance. In the advanced mode, the user decides when to perform bootstrapping or other maintenance operation, and even has an option not to perform bootstrapping at all.

The simple mode is easy to use, it suffices to generate or compile a small Boolean circuit that corresponds to the application, and evaluate it gate by gate on encrypted inputs. The homomorphic evaluation time is proportional to the plaintext evaluation of the same circuit. If the application can be written in terms of binary decision diagrams and lookup tables, the developer will often achieve much better performance by using the advanced mode. Some speed-ups using advanced mode are illustrated in [26] (lookup table and comparison circuit).

These schemes involve the following parameters:

- Bits of security λ (main parameter in all modes);
- Ciphertext-specific computation budget measure (only in advanced mode).

The bits of security parameter λ are related to the ciphertext modulus q and ciphertext dimension n for other schemes. In the simple mode, all the parameters can be derived from λ only.

In the advanced mode, the computation budget serves as a measure for the number of homomorphic operations that can be run on a ciphertext before a bootstrapping is required. Once all computations for a given ciphertext are performed, the user has to manually call bootstrapping to reset the computation budget for further computations on the ciphertext.

4.2 Homomorphic Operations

Computations over encrypted data always produce a ciphertext as output. The simple mode supports operations on Boolean circuits. The advanced mode supports operations on Boolean circuits, integers, and fixed-precision fractional numbers.

Simple Mode Plaintext Space and Operations

In the simple mode, the plaintext is just a Boolean value, and the main operations for Boolean circuits are:

- Constants
  - ZERO/ONE
- Unary gate
  - NOT
- Binary gates
  - AND/NAND
  - OR/NOR
  - XOR/XNOR
  - ORNOT/ANDNOT
- Ternary gates
  - MUX
  - Majority/Minority

In all these gates, the inputs and outputs are ciphertexts only. There is no direct support for mixed plaintext/ciphertext inputs because a Boolean gate that takes a plaintext as an input can always be simplified: e.g. x AND 1 = x, x AND 0 = 0, x XOR 1 = NOT x.

A DM/CGGI Hello World Example (Using Simple Mode)

```
# We first must set the bits of security
lambda = 128

# Generate the keys for these parameters
myPublicKey, mySecretKey = generateKeys(lambda)

# Encrypt data, each plaintext is a boolean value
encrypted_a = encrypt(myPublicKey, 1)
encrypted_b = encrypt(myPublicKey, 1)
excrypted_c = encrypt(myPublicKey, 0)
# We have three ciphertexts now.

# Compute the AND of the first two.
encrypted_AND = EvalGate(“AND”, myPublicKey, encrypted_a,
                         encrypted_b)
# Encryption of 1 AND 1 = 1

# Maybe we want to compute OR of this with the 3rd ciphertext.
encrypted_ANDOR = EvalGate(“OR”, myPublicKey, encrypted_AND,
                           encrypted_c)
# Encryption of (1 AND 1) OR 0 = 1

# To actually be able to see the final result we have to use
       the key.
decrypted = decrypt(mySecretKey, encrypted_ANDOR)
print(decrypted)
# This should print 1
```

Advanced Mode Plaintext Space and Operations

In the advanced mode, the scheme supports different plaintext types, as well as elementary operations across these plaintext spaces, and each ciphertext carries its own computation budget. The combination of plaintext types and computation budgets determines whether an operation is allowed, and at which performance it will be executed.

The main plaintext structure is a vector of n elements, which supports vector additions and some other vector operations (see below), but vector multiplications are not supported. In this section, we explain the plaintext arithmetic and give a few examples.

The main plaintext structure is a vector of n fixed-point numbers between -0.5 and 0.5 (i.e., modulo 1), given with a precision $\pm\alpha$ (each ciphertext has its own precision). This vector is encrypted in an RLWE[5] ciphertext with noise rate α. For instance, a ciphertext that encrypts a coefficient 0.0042 with precision

[5] Here RLWE is represented mod 1, with all coefficients divided by q used in BFV, BGV, CKKS and DM.

$\alpha = 10^{-9}$ means that it can be decrypted as any number between $0.0042 - 10^{-9}$ and $0.0042 + 10^{-9}$. This inherent error is analogous to the error in floating-point arithmetic (in the case of approximate arithmetic) or can be eliminated by post-decryption rounding if the message space is discretized (in the case of exact arithmetic).

If the coefficient 0.4942 was encrypted in a ciphertext with a much higher noise rate $\alpha = 10^{-2}$, it could be decrypted as any value between 0.4842 and 0.5042, and the second one would appear as -0.4958 modulo 1. This overflow is similar to the BGV/BFV case (with *mod p* plaintext space), but with respect to real numbers *mod* 1. If such overflow is not explicitly wanted by the application, it probably means that the noise is too large, or that the inputs should be scaled down.

The supported homomorphic operations are:

- Element-wise addition and subtraction: $x + y, x - y$
  - (0.0042, 0.0011, 0.0034) + (0.0074, 0.0089, 0.0011) → (0.0116, 0.0100, 0.0045)

  The result is always reduced modulo 1 (it can either be viewed as an expected behavior, or as an overflow condition which is mitigated by downscaling the space until every coefficient is much smaller than 1 like in the above example).
- Multiplication by a small public integer constant: noted $a * x$
  - $3 \times (0.0042, 0.0011, 0, 0034) \rightarrow (0.0126, 0.0033, 0.0102)$, for $a = 3$;
  - $123 * (0.0042, 0.0011, 0, 0034) \pm (\alpha = 10^{-5}) \rightarrow (0.517, 0.135, 0.418) \pm (\alpha = 10^{-3})$, for $a = 123$.

  Here, the factor 123 is rather large, if the input noise was $\pm 10^{-5}$, only 3 decimal digits of precision remain after scaling, since the noise amplitude also increases by a factor 123.
- (Anticyclic) shift by k positions: noted $rot_k(x)$, with k a public value
  - $rot_0(\ (0.0042, 0.0011, 0, 0034, 0, 0, 0)\ ) \rightarrow (0.0042, 0.0011, 0, 0034, 0, 0, 0)$, for $k = 0$
  - $rot_2(\ (0.0042, 0.0011, 0, 0034, 0, 0, 0)\ ) \rightarrow (0, 0, 0.0042, 0.0011, 0, 0034, 0)$, for $k = 2$
  - $rot_3(\ (0.0042, 0.0011, 0, 0034, 0, 0, 0)\ ) \rightarrow (0, 0, 0, 0.0042, 0.0011, 0, 0034)$, for $k = 3$

  Any coefficient that vanishes to the right of the vector appears back on the left side with the opposite sign, so with the same example, if $n = 6$.

  - $rot_4(\ (0.0042, 0.0011, 0, 0034, 0, 0, 0)\ ) \rightarrow (-0.0034, 0, 0, 0, 0.0042, 0.0011)$, for $k = 4$
  - $rot_5((0.0042, 0.0011, 0, 0034, 0, 0, 0)) \rightarrow (-0.0011, -0.0034, 0, 0, 0, 0.0042)$, for $k = 5$
- There are also more involved operations, such as selecting only one particular position, or all odd or even positions and canceling all other positions.

Any combination of the above addition/scaling/rotations is possible: e.g. $x + 2 * rot_1(x) + 5 * rot_2(y)$ means x + twice x rotated by one position +5 times y rotated by two positions. The same expression can equivalently be factorized as $(rot_0 + 2rot_1).\ (x) + (5rot_2).\ (y)$, which isolates the linear transformations applied on each ciphertext. Applying a linear transformation on a ciphertext increases its noise (and hence the output error) by the norm of all rotation coefficients: the bigger the coefficients, the larger is the resulting noise, and the application designer must always ensure that the resulting noise remains small enough to decrypt the output.

Having all these definitions and constraints in mind, the user is free to use any plaintext vector, encrypted as an RLWE ciphertext, which can represent a real or fractional number (mod 1).[6]

If the application cannot be expressed in terms of element-wise addition, public scaling or rotation with public index and we need non-linear operations, we should use a different encryption scheme called RGSW (vector of RLWE ciphertexts). It provides the possibility to evaluate any secret linear transformation (homomorphically encrypted).

- *Homomorphic action (external product):* Given an RGSW ciphertext that encrypts a linear transformation f, and an RLWE ciphertext that encrypts a real vector x, obtain the encryption of $f(x)$.

The most useful applications of this concept are essentially:

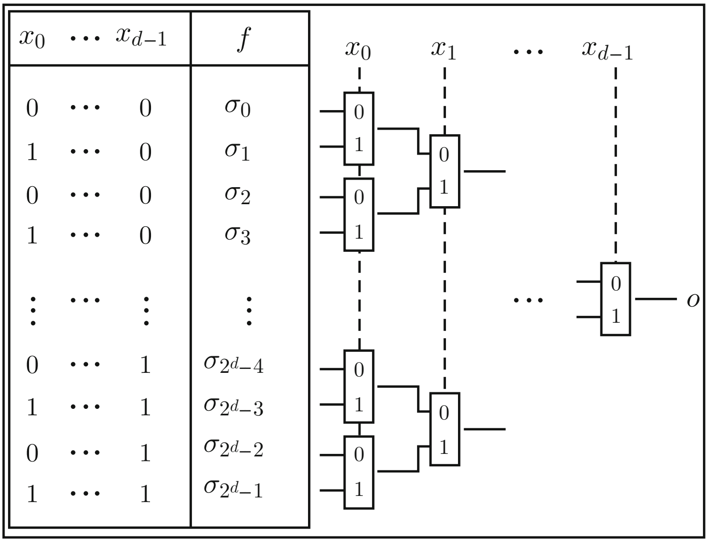
- *BlindRotation:* Given a RGSW ciphertext that encrypts rot_k, and a RLWE ciphertext that encrypts x, obtain a RLWE encryption of $rot_k(x)$, where k remains secret.
- *PrivateSelection* (CMUX): Given a RGSW ciphertext that encrypts $c = 1$ or 0,[7] and two RLWE ciphertexts that encrypt x and y, obtain an RLWE encryption of the selection $c?x{:}y$ (written like in C), which is equal to x when $c = 1$ and y when $c = 0$.

The CMUX is a building block for the evaluation of any binary decision diagrams or deterministic finite automata (DFA), like the lookup table or the automata in Figs. 2 and 3, where each selector is one CMUX gate.

These two operations above only add a constant amount of noise on top of the input RLWE ciphertexts, allowing to chain a large amount of these operations with negligible noise growth, and hence to build complex decision diagrams or deterministic automata. Many arithmetic circuits correspond to simple decision diagrams, the most famous of them is the decryption function, which is used to bootstrap ciphertexts in the simple mode.

[6] Fractional number mod 1 corresponds to an integer mod p, as in BGV/BFV, divided by p

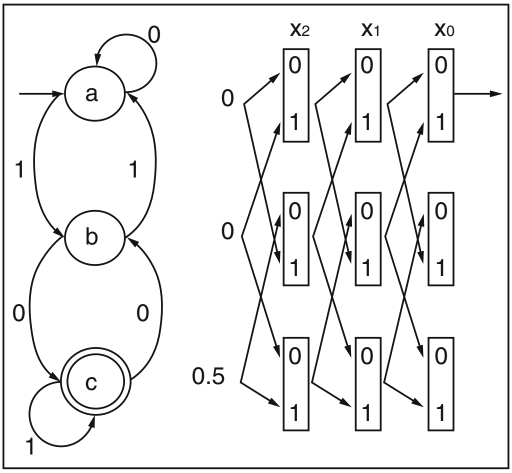
[7] 1 is the identity function



x_0	···	x_{d-1}	f
0	···	0	σ_0
1	···	0	σ_1
0	···	0	σ_2
1	···	0	σ_3
⋮	···	⋮	⋮
0	···	1	σ_{2^d-4}
1	···	1	σ_{2^d-3}
0	···	1	σ_{2^d-2}
1	···	1	σ_{2^d-1}

Fig. 2 Evaluation of lookup table.V

Fig. 3 Evaluation of DFA



Advanced-Mode CGGI Hello World Example (Corresponds to the DFA in Fig. 1)

```
# We first must set the bits of security and the noise-rate
lambda = 128
alpha = 2^-15

# Generate the keys for these parameters
myPublicKey,mySecretKey = generateKeys(lamda, alpha)
```

```
# encrypt each letter with RGSW
encrypted_x = [ encryptRGSW(myPublicKey, 0),
                encryptRGSW(myPublicKey, 1),
                encryptRGSW(myPublicKey, 0) ]

# the initial state values are trivial RLWE ciphertexts
a = RLWE(0)
b = RLWE(0)
c = RLWE(0.5)

For i = 3 to 1
    # evaluate each transition
    newA = CMux(encrypted_x[i],b,a)
    newB = CMux(encrypted_x[i],a,c)
    newC = CMux(encrypted_x[i],c,b)
    (a,b,c) = (newA,newB,newC)
EndFor
# To actually see the final result, we have to use the key.
decrypted = decrypt(mySecretKey, a)
Return decrypted
```

4.3 Further Information

Advanced Notes on Parameters

The computation budget can be equivalently expressed as the standard deviation of the ciphertext noise (noise rate α) for implementations of CGGI in the TFHE library, or to the modulus q for modular instantiations of DM and CGGI. The correspondence between the modulus q described in the security tables and the noise rate α is $q = 3.2/\sqrt{2\pi}\ \alpha$. This means, the noise grows at each operation until it reaches critical levels, or q can be rescaled down after each operation, until it reaches its minimum.

In the advanced mode, there are many sets of parameters n and α (q), each corresponding to a specific (sub-)circuit that gets bootstrapped. The details of setting the computational budget are implementation-specific.

Some More Advanced Operations Are Supported

- *Addition and composition of transformations:* Given two RGSW ciphertexts encoding f and g, we can homomorphically obtain single RGSW ciphertexts that encrypt respectively $f + g$ and $f \circ g$. This is in line with the original ring operations described in the GSW scheme [31], and it is used in DM bootstrapping [29].

- *Multiplication by the secret key s:* This allows the evaluation of polynomials in s, and by extension, unlocks a variant of BFV and CKKS schemes supporting homomorphic element-wise addition/multiplication either modulo p or on fixed point-numbers, and that can be combined with the above circuit operations (see Fig. 3 at the end of the section). This is discussed at a more advanced level in the Chimera extension of CGGI in [28].

Maintenance Operations (and More)

The DM/CGGI schemes also support some maintenance operations:

- (both DM/CGGI)
  - *Gate bootstrapping,* which "refreshes" the noise of a binary gate ciphertext. Unlike BFV, BGV and CKKS, the gate bootstrapping is fast, in the order of a few milliseconds [25, 29], and is always applied after each boolean gate in the simple mode.
  - *(Functional) Key switching* allows to switch between scalar and polynomial message spaces, and to apply any linear combination with small integer coefficients.
- (CGGI-specific)
  - *Circuit bootstrapping*, which "converts" a LWE ciphertext from the space {0,1/2} into a RGSW ciphertext of 0 or 1. The circuit bootstrapping is applied in the advanced mode to compose binary decision diagrams, and runs in 137 milliseconds [26].
  - *Functional bootstrapping*, allows to approximate homomorphically a non-linear real-valued function [23, 27].
- (DM-specific)
  - *Modulus Rescaling*, which follows almost all multiplication operations on representations modulo q. This operation simplifies the ciphertext to obtain shorter equivalent representation, with a fixed noise amplitude. This operation is implicit on representations modulo 1, where the precision of the ciphertext representation is always of the order of α.

Advanced Functionality in the CGGI Encryption Scheme

The CGGI scheme in advanced mode proposes two methods to operate on batched data to decrease the ciphertext expansion and to optimize the evaluation of look-up tables and arbitrary functions. The batching techniques provide the possibility to use the computation slots at their maximal capacity, even if the function itself is not SIMD, or has very few bits of output.

CGGI also extends the automata logic, to the leveled evaluation of deterministic weighted finite automata (WFA). These improvements speed up the evaluation of most arithmetic functions in a batched advanced mode, with a noise overhead that remains additive (more information is described in [26]).

Difference Between DM and CGGI

The CGGI scheme supports the both simple and advanced modes (with a special circuit bootstrapping proposed in [26]), DM currently supports just the simple mode, but can be generalized.

The main difference between the CGGI and DM schemes in the simple mode is in the bootstrapping procedure used for refreshing the ciphertexts [32]. CGGI uses the Gama–Izabachene–Nguyen–Xie bootstrapping procedure [30] based on CMUX gates while CGGI uses the Alperin-Sherif–Peikert bootstrapping procedure [22] via the composition of GSW ciphertexts.

Variants of DM/CGGI

The original CGGI scheme was constructed using the *Torus (Ring) Learning With Errors* (TLWE/TRLWE) problem, which is a generalization of LWE/RLWE rescaled over the *real Torus* (a set of real numbers modulo 1) [25]. The original DM scheme was constructed using LWE/RLWE [29]. The CGGI scheme was subsequently instantiated using LWE/RLWE in a unified framework including both DM and CGGI [32].

Scheme Switching Using CGGI

The Chimera framework described in [28] unifies CGGI (TFHE), BFV and CKKS under the TLWE/TRLWE/TRGSW problems and allows using the three schemes in the same computation. Figure 4 provides a schematic of scheme switching in the Chimera framework.

Reference Implementations

The following libraries have open-source CPU implementations of DM and CGGI:

- TFHE (CGGI scheme in simple and advanced modes using binary secret distribution)

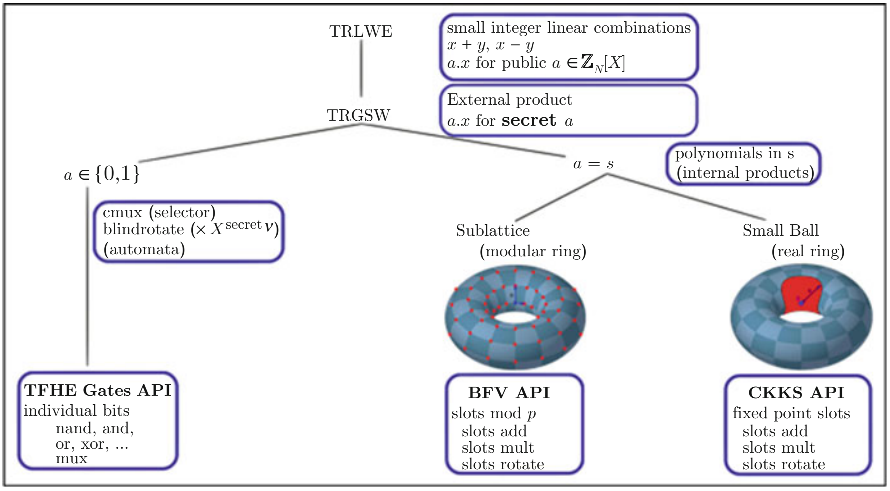


Fig. 4 The Chimera framework

- FHEW (DM scheme in the simple mode using binary distribution)
- PALISADE (DM and CGGI in simple mode using ternary secret distribution)

Please note that the parameters for binary secret distributions are not currently included in the HE Security Standard [9] but are being considered for inclusion in the standard.

The following libraries have GPU implementations of CGGI:

- cuFHE (GPU version of the CGGI scheme in the simple mode)
- nuFHE (GPU version of the CGGI scheme in the simple mode)

A proof of concept implementation of scheme switching between CGGI and CKKS (using the HEAAN library) is publicly accessible [24].

References

1. J. Alperin-Sheriff, and C. Peikert. *Faster Bootstrapping with Polynomial Error*. In CRYPTO 2014. Pages 297–314.
2. J.C. Bajard, J. Eynard, M.A. Hasan, and V. Zucca, V. *A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes*. In SAC 2016. Pages 423–442.
3. Z. Brakerski. *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP*. In CRYPTO 2012. Pages 868–886.
4. A. Al Badawi, Y. Polyakov, K.M.M. Aung, B. Veeravalli, and K. Rohloff. *Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme*. IEEE Transactions on Emerging Topics in Computing (2019). https://eprint.iacr.org/2018/589.
5. H. Chen and K. Han. *Homomorphic Lower Digits Removal and Improved FHE Bootstrapping*. In EUROCRYPT 2018. Pages 315–337.

6. J. Fan and F. Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive. Report 2012/144, 2012. http://eprint.iacr.org/2012/144.
7. C. Gentry, S. Halevi, and N. P. Smart. *Homomorphic Evaluation of the AES Circuit*. In CRYPTO 2012. Pages 850–867.
8. C. Gentry, S. Halevi, and N. P. Smart. *Better Bootstrapping in Fully Homomorphic Encryption*. In Public Key Cryptography 2012. Pages 1–16.
9. M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan. *Security of Homomorphic Encryption*. http://homomorphicencryption.org/white_papers/security_homomorphic_encryption_white_paper.pdf
10. S. Halevi, Y. Polyakov, and V. Shoup. *An Improved RNS Variant of the BFV Homomorphic Encryption Scheme*. In CT-RSA 2019. Pages 83–105.
11. S. Halevi and V. Shoup. *Algorithms in HElib*. In CRYPTO 2014. Pages 554–571.
12. S. Halevi and V. Shoup. *Bootstrapping for HElib*. In EUROCRYPT 2015. Pages 641–670.
13. T. Lepoint and M. A. Naehrig. *A Comparison of the Homomorphic Encryption Schemes FV and YASHE*. In AFRICACRYPT 2014. Pages 318–335.
14. C. Mouchet, J. Troncoso-Pastoriza and J.-P. Hubaux. *Multiparty Homomorphic Encryption: From Theory to Practice*. Cryptology ePrint Archive, Report 2020/304, 2020. http://github.com/ldsec/lattigo
15. J. H. Cheon, A. Kim, M. Kim, Y. Song, *Homomorphic Encryption for Arithmetic of Approximate Numbers*. In ASIACRYPT 2017. Pages 409–437.
16. J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song, *Bootstrapping for Approximate Homomorphic Encryption*. In EUROCRYPT 2018. Pages 360–384.
17. J. H. Cheon, K. Han, A. Kim, M. Kim, Y. Song, *A Full RNS Variant of the Approximate Homomorphic Encryption*. In SAC 2018. Pages 347–368.
18. H. Chen, I. Chillotti, Y. Song, *Improved Bootstrapping for Approximate Homomorphic Encryption*. In EUROCRYPT 2019. Pages 34–54.
19. M. Blatt, A. Gusev, Y. Polyakov, K. Rohloff, V. Vaikuntanathan, *Optimized Homomorphic Encryption Solution for Secure Genome-Wide Association Studies*, BMC Medical Genomics, 2020.
20. M. Kim, Y. Song, B. Li, D. Micciancio, *Semi-Parallel Logistic Regression for GWAS on Encrypted Data*, BMC Medical Genomics, 2020.
21. K. Han, D. Ki, *Better Bootstrapping for Approximate Homomorphic Encryption*. In CT-RSA 2020. Pages 364–390.
22. J. Alperin-Sheriff and C. Peikert. Faster Bootstrapping with Polynomial Error. CRYPTO 2014.
23. F. Bourse, M. Minelli, M. Minihold, P. Paillier: Fast Homomorphic Evaluation of Deep Discretized Neural Networks. CRYPTO (3) 2018: 483-512.
24. https://github.com/DPPH/chimera-iDash2018
25. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster Fully Homomorphic Encryption Bootstrapping in Less Than 0.1 Seconds. In Asiacrypt 2016 (Best Paper), pages 3–33.
26. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. ASIACRYPT (1) 2017: 377–408.
27. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast Fully Homomorphic Encryption over the Torus. Journal of Cryptology 2019.
28. C. Boura, N. Gama, M. Georgieva and D. Jetchev: CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes. IACR Cryptology ePrint Archive 2018: 758 (2018) (NutMic, submitted to Journal of Mathematical Cryptology 2019).
29. L. Ducas and D. Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. EUROCRYPT 2015.
30. N. Gama, M. Izabachene, P. Q. Nguyen, and X. Xie. Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions and Homomorphic Cryptosystems. EUROCRYPT 2016.
31. C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption From Learning With Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. CRYPTO 2013.

32. D. Micciancio and Y. Polyakov. Bootstrapping in FHEW-like Cryptosystems. Cryptology ePrint Archive. Report 2020/086, 2020. http://eprint.iacr.org/2020/086.
33. https://tfhe.github.io/tfhe/
34. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):1-36, 2014.
35. Z. Brakerski and V. Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. SIAM Journal on Computing, 43(2):831-871, 2014.

Part II
Homomorphic Encryption Security Standard

Homomorphic Encryption Standard

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan

We met as a group during the Homomorphic Encryption Standardization Workshop on July 13–14, 2017, hosted at Microsoft Research in Redmond, and again during the second workshop on March 15–16, 2018 in MIT. Researchers from around the world represented government, industry, and academia. There are several research

M. Albrecht
Royal Holloway University, London, UK
e-mail: Martin.Albrecht@rhul.ac.uk

M. Chase · K. Laine · S. Lokam
Cryptography and Privacy Research Group, Microsoft Research, Redmond, WA, USA
e-mail: melissac@microsoft.com; kim.laine@microsoft.com; Satya.Lokam@microsoft.com

H. Chen
Facebook, Menlo Park, CA, USA
e-mail: haoche@fb.com

J. Ding
University of Cincinnati, Cincinnati, OH, USA
e-mail: dingji@ucmail.uc.edu

S. Goldwasser
Simons Institute, Berkeley, CA, USA
e-mail: shafi@csail.mit.edu

S. Gorbunov
University of Waterloo, Waterloo, ON, Canada
e-mail: sgorbunov@uwaterloo.ca

S. Halevi
Algorand Foundation, Yorktown Heights, NY, USA
e-mail: shaih@alum.mit.edu

J. Hoffstein
Brown University, Providence, RI, USA
e-mail: jhoff@math.brown.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_2

groups around the world who have made libraries for general-purpose homomorphic encryption available for applications and general-purpose use. Some examples include [40–46, 47]. Most general-purpose libraries for homomorphic encryption implement schemes that are based on the ring learning-with-error (RLWE) problem, and many of them displayed common choices for the underlying rings, error distributions, and other parameters.

Homomorphic Encryption is a breakthrough new technology which can enable private cloud storage and computation solutions, and many applications were described in the literature in the last few years. But before Homomorphic Encryption can be adopted in medical, health, and financial sectors to protect data and patient and consumer privacy, it will have to be standardized, most likely by multiple standardization bodies and government agencies. An important part of standardization is broad agreement on security levels for varying parameter sets. Although extensive research and benchmarking has been done in the research community to establish the foundations for this effort, it is hard to find all the information in one place, along with concrete parameter recommendations for applications and deployment.

This document is an attempt to capture (at least part of) the collective knowledge regarding the currently known state of security of these schemes, to specify the schemes, and to recommend a wide selection of parameters to be used for homomorphic encryption at various security levels. We describe known attacks and their estimated running times in order to make these parameter recommendations. We also describe additional features of these encryption schemes which make them useful in different applications and scenarios.

K. Lauter
Facebook AI Research, Seattle, WA, USA
e-mail: klauter@fb.com

D. Micciancio
University of California San Diego, San Diego, CA, USA
e-mail: daniele@cs.ucsd.edu

D. Moody
Computer Security Division, National Institute of Standards and Technology, Gaithersburg, MD, USA
e-mail: dustin.moody@nist.gov

T. Morrison
Mathematics, Virginia Tech University, Blacksburg, VA, USA
e-mail: tmo@vt.edu

A. Sahai
Computer Science, UCLA, Los Angeles, CA, USA
e-mail: sahai@cs.ucla.edu

V. Vaikuntanathan
Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: vinodv@csail.mit.edu

Outline

HES Section 1 standardizes the encryption schemes to be used.

Section 1.1: introduces notation and definitions.
Section 1.2: defines the security properties for homomorphic encryption.
Section 1.3: describes the BGV and B/FV schemes.
Section 1.4: describes the GSW scheme.
Section 1.5: mentions some alternative schemes: [39], [37]/[33], and [46].
Section 1.6: describes additional features of the schemes.

HES Section 2 recommends parameter choices to achieve security.

Section 2.1: describes the hard problems: the LWE and RLWE assumptions.
Section 2.2: describes known lattice attacks and their estimated running times.
Section 2.3: mentions the Arora-Ge attack on LWE.
Section 2.4: discusses algebraic attacks on RLWE.
Section 2.5: recommends concrete parameters to achieve various security levels.

1 Homomorphic Encryption Standard Section 1: Recommended Encryption Schemes

1.1 Notation and Definitions

- ParamGen(λ, *PT, K, B*) → Params

The parameter generation algorithm is used to instantiate various parameters used in the HE algorithms outlined below. As input, it takes:

- λ denotes the desired security level of the scheme. For instance, 128-bit security ($\lambda = 128$) or 256-bit security.
- *PT* denotes the underlying plaintext space. Currently this standard specifies two types of parametrized plaintext spaces: modular integers (MI), and extension fields/rings (EX). We expect future versions of this document to introduce a third type of approximate numbers (AN).
  - (MI) Modular integers are parametrized by the modulus p of the plaintext numbers to be encrypted, namely the plaintext space is Z_p. For instance, the parameter p=*1024* means that the plaintext space is Z_{1024}, *i.e.,* each individual element of the message space is an integer from the range (0, 1023) and all operations on individual elements are performed modulo p.
  - (EX) Extension rings/fields are parameterized by a modulus p as above, and in addition by a polynomial $f(x)$ over Z_p, specifying the plaintext space as $Z[x]/(p,f(x))$. Namely, each element of the message space is an integer

polynomial of degree smaller than *f(x)* with coefficients from the range (0, p-1), and all operations over individual elements are performed modulo *f(x)*, and modulo p.

- K denotes the dimension of the vectors to be encrypted. For instance, $K = 100$, $PT = (MI, 1024)$ means the messages to be encrypted are vectors $(V_1, \ldots, V_K)$ where each V_i is chosen from the range (0, 1023) and operations are performed component-wise. That is, by definition, $(V_1, \ldots, V_K) + \left(V_1', \ldots, V_K'\right) = \Big(V_1 + V_1', \ldots, \ V_K + V_K'\Big)$. The multiplication operation over two vectors is defined similarly. The space of all possible vectors $(V_1, \ldots, V_K)$ is referred to as the message space (MS).
- B: denotes an auxiliary parameter that is used to control the complexity of the programs/circuits that one can expect to run over the encrypted messages. Lower parameters denote "smaller", or less expressive, or less complex programs/circuits. Lower parameters generally mean smaller parameters of the entire scheme. This, as a result, translates into smaller ciphertexts and more efficient evaluation procedures. Higher parameters generally increase key sizes, ciphertext sizes, and complexity of the evaluation procedures. Higher parameters are, of course, necessary to evaluate more complex programs.

- PubKeygen(Params) → SK, PK, EK

The public key-generation algorithm is used to generate a pair of secret and public keys. The public key can be shared and used by anyone to encrypt messages. The secret key should be kept private by a user and can be used to decrypt messages. The algorithm also generates an evaluation key that is needed to perform homomorphic operations over the ciphertexts. It should be given to any entity that will perform homomorphic operations over the ciphertexts. Any entity that has only the public and the evaluation keys cannot learn anything about the messages from the ciphertexts only.

- SecKeygen(Params) → SK, EK

The secret key-generation algorithm is used to generate a secret key. This secret key is needed to both encrypt and decrypt messages by the scheme. It should be kept private by the user. The algorithm also generates an evaluation key that is needed to perform homomorphic operations over the ciphertexts. The evaluation key should be given to any entity that will perform homomorphic operations over the ciphertexts. Any entity that has only the evaluation key cannot learn anything about the messages from the ciphertexts only.

- PubEncrypt(PK, M) → C

The public encryption algorithm takes as input the public key of the scheme and any message M from the message space. The algorithm outputs a ciphertext C. This algorithm generally needs to be randomized (that is, use random or pseudo-random coins) to satisfy the security properties.

- SecEncrypt(SK, M) → C

The secret encryption algorithm takes as input the secret key of the scheme and any message M from the message space. The algorithm outputs a ciphertext C. This algorithm generally needs to be randomized (that is, use random or pseudo-random coins) to satisfy the security properties.

- Decrypt(SK, C) → M

The decryption algorithm takes as input the secret key of the scheme, SK, and a ciphertext C. It outputs a message M from the message space. The algorithm may also output special symbol FAIL, if the decryption cannot successfully recover the encrypted message M.

- EvalAdd(Params, EK, C1, C2) → C3.

EvalAdd is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, two ciphertexts C1 and C2, and outputs a ciphertext C3.

The correctness property of EvalAdd is that if C1 is an encryption of plaintext element M1 and C2 is an encryption of plaintext element M2, then C3 should be an encryption of M1+M2.

- EvalAddConst(Params, EK, C1, M2) → C3.

EvalAddConst is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, a ciphertext C1, and a plaintext M2, and outputs a ciphertext C3.

The correctness property of EvalAddConst is that if C1 is an encryption of plaintext element M1, then C3 should be an encryption of M1+M2.

- EvalMult(Params, EK, C1, C2) → C3.

EvalMult is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, two ciphertexts C1 and C2, and outputs a ciphertext C3.

The correctness property of EvalMult is that if C1 is an encryption of plaintext element M1 and C2 is an encryption of plaintext element M2, then C3 should be an encryption of M1*M2.

- EvalMultConst(Params, EK, C1, M2) → C3.

EvalMultConst is a randomized algorithm that takes as input the system parameters Params, the evaluation key EK, a ciphertexts C1, and a plaintext M2, and outputs a ciphertext C3.

The correctness property of EvalMultConst is that if C1 is an encryption of plaintext element M1, then C3 should be an encryption of M1*M2.

- Refresh(Params, flag, EK, C1) → C2.

Refresh is a randomized algorithm that takes as input the system parameters Params, a multi-valued flag (which can be either one of "Relinearize", "ModSwitch" or "Bootstrap"), the evaluation key EK, and a ciphertext C1, and outputs a ciphertext C2.

The correctness property of Refresh is that if C1 is an encryption of plaintext element M1, then C2 should be an encryption of M1 as well.

The desired property of the Refresh algorithm is that it turns a "complex" ciphertext of a message into a "simple" one of the same message. Two embodiments of the Refresh algorithm are (a) the bootstrapping procedure, which takes a ciphertext with large noise and outputs a ciphertext of the same message with a fixed amount of noise; and (b) the key-switching procedure, which takes a ciphertext under one key and outputs a ciphertext of the same message under a different key.

- ValidityCheck(Params, EK, [C], COMP) → flag.

ValidityCheck is an algorithm that takes as input the system parameters Params, the evaluation key EK, an array of ciphertexts [C], and a specification of the homomorphic computation encoded as a straight-line program COMP, and outputs a Boolean flag.

The correctness property of ValidityCheck is that if ValidityCheck outputs flag = 1, then doing the homomorphic computation COMP on the vector of ciphertexts [C] produces a ciphertext that decrypts to the correct answer.

1.2 Properties

Semantic Security or IND-CPA Security At a high level, a homomorphic encryption scheme is said to be secure if no adversary has an advantage in guessing (better than ½ chance) whether a given ciphertext is an encryption of two different messages. This requires encryption to be randomized so that two different encryptions of the same message do not look the same.

Suppose a user runs the parameter and the key-generation algorithms to provide the key tuple. An adversary is assumed to have the parameters, the evaluation key EK, a public key PK (only in the public-key scheme) and can obtain encryptions of messages of its choice. The adversary is then given an encryption of one of two messages of its choice, computed by the above encryption algorithm, without knowing which message the encryption corresponds to. The security of HE then guarantees that the adversary cannot guess which message the encryption corresponds to with significant advantage better than a ½ chance. This captures the fact that no information about the messages is revealed in the ciphertext.

Compactness The compactness property of a homomorphic encryption scheme guarantees that homomorphic operations on the ciphertexts do not expand the length of the ciphertexts. That is, any evaluator can perform an arbitrary supported list of evaluation function calls and obtain a ciphertext in the ciphertext space (that does not depend on the complexity of the evaluated functions).

Efficient Decryption Efficient decryption property says that the homomorphic encryption scheme always guarantees that the decryption runtime does not depend on the functions which was evaluated on the ciphertexts.

1.3 The BGV and B/FV Homomorphic Encryption Schemes

In this section, we describe the two primary schemes for implementation of homomorphic encryption, [10] and [11]/[23], these two schemes are very similar. In Section 1.4. below we describe the GSW scheme, which is somewhat different. In Section 1.5, we also mention some alternative schemes [39], [37]/[33], and [46], but they are not described in this standard.

(a) **Brakerski-Gentry-Vaikuntanathan (BGV)**

We focus here on describing the basic version of the BGV encryption scheme. Optimizations to the basic scheme will be discussed at the end of this section.

- BGV.ParamGen(λ, PT, K, B) → Params.

Recall that λ is the security level parameter, for BGV the plaintext space PT is either of type MI or EX with integer modulus p > 1, and K ≥ 1 is an integer vector length.

In the basic BGV scheme, the auxiliary input B is an integer that determines the maximum multiplicative depth of the homomorphic computation. This is simply the maximum number of sequential multiplications required to perform the computation. For example, the function $g(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4$ has multiplicative depth 1.

In the basic BGV scheme, the parameters param include the ciphertext modulus parameter q and a ring $R = Z[x]/f(x)$ and corresponding plaintext ring R/pR and ciphertext ring R/qR. The parameters param also specify a "key distribution" D_1 and an "error distribution" D_2 over R, the latter is based on a Gaussian distribution with standard deviation σ set according to the security guidelines specified in Section 2.5.

- BGV.SecKeygen(params) → SK, EK

In the basic BGV scheme, the secret key SK is an element s in the ring R, chosen from distribution D_1.

In the basic BGV scheme, there is no evaluation key EK.

- BGV.PubKeygen(params) → SK, PK, EK.

In the basic BGV scheme, PubKeygen first runs SecKeygen and obtains (SK, EK) where SK is an element s that belongs to the ring R.

PubKeygen chooses a uniformly random element a from the ring R/qR and outputs the public key PK which is a pair of ring elements $(pk_0, pk_1) = (-a, as + pe)$ where e is chosen from the error distribution D_2.

- BGV.SecEncrypt(SK, M) → C

In the basic BGV scheme, SecEncrypt first maps the message M which comes from the plaintext space (either $Z_p{}^r$ or $(Z_p[x]/f(x))^r$) into an element $\hat{M}$ of the ring R/pR.

SecEncrypt then samples a uniformly random element a from the ring R/qR and outputs the pair of ring elements $(c_0, c_1) = \left(-a, as + pe + \hat{M}\right)$ where e is chosen from the error distribution D_2. (See Comments 1, 2 below for more general methods of encoding the message during encryption. The same comments apply also to public-key encryption with BGV.)

- BGV.PubEncrypt(PK, M) → C

In the basic BGV scheme, PubEncrypt first maps the message M which comes from the plaintext space Z_p^k into an element $\hat{M}$ of the ring R/pR. Recall that the public key PK is a pair of elements (pk_0, pk_1).

PubEncrypt then samples three elements u from distribution D_1 and e_1, e_2 from the error distribution D_2 and outputs the pair of ring elements $(c_0, c_1) = \left(pk_0u + pe_1, pk_1u + pe_2 + \hat{M}\right)$.

- BGV.Decrypt(SK, C) → M

In the basic BGV scheme, Decrypt takes as input the secret key which is an element s of the ring R, and a ciphertext $C = (c_0, c_1)$ which is a pair of elements from the ring R/qR.

We remark that a ciphertext C produced as the output of the encryption algorithm has two elements in R/qR, but upon homomorphic evaluation, ciphertexts can grow to have more ring elements. The decryption algorithm can be modified appropriately to handle such ciphertexts.

Decrypt first computes the ring element $c_0s + c_1$ over R/qR and interprets it as an element $c^{'}$ in the ring R. It then computes $c^{'} \pmod{p}$, an element of R/pR, which it outputs.

- BGV.EvalAdd(Params, EK, C1, C2) → C3.

In the basic BGV scheme, EvalAdd takes as input ciphertexts $C1 = (c_{1,0}, c_{1,1})$ and $C2 = (c_{2,0}, c_{2,1}\ )$ and outputs $C3 = (c_{1,0} + c_{2,0},\ c_{1,1} + c_{2,1})$, where the operations are done in R/qR.

- BGV.EvalMult(Params, EK, C1, C2) → C3.

In the basic BGV scheme, EvalMult takes as input ciphertexts $C1 = (c_{1,0}, c_{1,1})$ and $C2 = (c_{2,0}, c_{2,1})$ and outputs $C3 = (c_{1,0}c_{2,0},\ c_{1,0}c_{2,1} + c_{1,1}c_{2,0}, c_{1,1}c_{2,1})$, where the operations are done in R/qR.

Comment 1 The noise term $pe+\hat{M}$ in the encryption procedure can be generalized to an error term drawn from the coset $\hat{M} + pR$, according to an error-sampling procedure. All the considerations discussed below for the error distribution D_2, apply equally to the error-sampling procedure in this more general implementation.

Comment 2 There is also an equivalent "MSB encoding" of the message for BGV encryption, where the message is encoded as $W\hat{M}$+e (with $W = \lfloor q/p \rfloor$, similarly to the B/FV scheme below). There are lossless conversions between these two encoding methods, as long as the plaintext modulus p is co-prime with the ciphertext modulus q.

The Full BGV Scheme

In the basic BGV scheme, ciphertexts grow as a result of EvalMult. For example, given two ciphertexts each composed of two ring elements, EvalMult as described above results in three ring elements. This can be further repeated but has the disadvantage that upon evaluating a degree-d polynomial on the plaintexts, the resulting ciphertext has $d + 1$ ring elements.

This deficiency is mitigated in the full BGV scheme, with two additional procedures. The first is called "Key Switching" or "Relinearization" which is implemented by calling the Refresh subroutine with flag = "KeySwitch", and the second is "Modulus Switching" or "Modulus Reduction" which is implemented by calling the Refresh subroutine with flag = "ModSwitch". Support for key switching and modulus switching also necessitates augmenting the key generation algorithm.

For details on the implementation of the full BGV scheme, we refer the reader to [10].

Properties Supported The BGV scheme supports many features described in Section 6, including packed evaluations of circuits and can be extended into a threshold homomorphic encryption scheme. In terms of security, the BGV homomorphic evaluation algorithms can be augmented to provide evaluation privacy (with respect to semi-honest adversaries).

(b) **Brakerski/Fan-Vercauteren (B/FV)**

We follow the same notations as the previous section.

- BFV.ParamGen(λ, PT, K, B) → Params.

We assume the parameters are instantiated following the recommendations outlined in Section 5. Similarly to BGV, the parameters include:

- Key- and error-distributions D_1, D_2
- a ring R and its corresponding integer modulus q
- Integer modulus p for the plaintext

In addition, the B/FV parameters also include:

- Integer T, and $L = \log_T q$. T is the bit-decomposition modulus.
- Integer $W = \lfloor q/p \rfloor$

- BFV.SecKeygen(Params) -> SK, EK

The secret key SK of the encryption scheme is a random element$s$ from the distribution D_1 defined as per Section 5. The evaluation key consists of L LWE samples encoding the secret s in a specific fashion.

In particular, for $i = 1, \ldots, L$, sample a random a_i from R/qR and error e_i from D_2, compute

$$EK_i = \Big(-(a_i s + e_i) + T^i s^2, a_i\Big),$$

and set $EK = (EK_1, \ldots, EK_L)$.

- BFV.PubKeygen(params) -> SK, PK, EK.

The secret key SK of the encryption scheme is a random element s from the distribution D_1. The public key is a random LWE sample with the secret s. In particular, it is computed by sampling a random element a from R/qR and an error e from the distribution D_2 and setting:

$PK = (-(as + e), a)$, where all operations are performed over the ring R/qR.

The evaluation key is computed as in BFV.SecKeygen.

- BFV.PubEncrypt(PK, M) -> C

BFV.Pub.Encrypt first maps the message M which comes from the message space into an element in the ring R/pR .

To encrypt a message M from R/pR, parse the public key as a pair (pk_0, pk_1). Encryption consists of two LWE samples using a secret u where (pk_0, pk_1) is treated as public randomness. The first LWE sample encodes the message M, whereas the second sample is auxiliary.

In particular, $C = (pk_0 u + e_1 + WM, pk_1 u + e_2)$ where u is a sampled from D_1 and e_1, e_2 are sampled from D_2.

- BFV.SecEncrypt(PK, M) -> C

- BFV.Decrypt(SK, C) -> M

The main invariant of the BFV scheme is that when we interpret the elements of a ciphertext C as the coefficients of a polynomial then, $C(s) = W\,M + e$ for some "small" error e. The message M can be recovered by dividing the polynomial $C(s)$ by W, rounding each coefficient to the nearest integer, and reducing each coefficient modulo p.

- BFV.EvalAdd(EK, C1, C2) -> C3

Parse the ciphertexts as $Ci = (c_{i,0}, c_{i,1})$. Then, addition corresponds to component-wise addition of two ciphertext components. That is, $C3 = (c_{1,0} + c_{2,0}, c_{1,1} + c_{2,1})$.

It is easy to verify that $C3(s) = W(M_1 + M_2) + e$, where M_1, M_2 are messages encrypted in $C1$, $C2$ and e is the new error component.

- BFV.EvalMult(EK, C1, C2) -> C3

EvalMult takes as input ciphertexts $C1 = (c_{1,0}, c_{1,1})$ and $C2 = (c_{2,0}, c_{2,1})$. First, it computes

$C3' = (c_{1,0}c_{2,0},\ c_{1,0}c_{2,1} + c_{1,1}c_{2,0}, c_{1,1}c_{2,1})$ over the integers (instead of mod q as in BGV scheme above). Then set $C3 = round\left(\left(\frac{p}{q}\right)C3'\right) \bmod q$.

One can verify that $C3(s) = W(M_1 * M_2) + e$, for some error term e.

Note that the ciphertext size increases in this operation. One may apply a Relinearization algorithm as in the BGV scheme to obtain a new ciphertext of the original size encrypting the same message $M_1 * M_2$.

Properties Supported The complete BFV scheme supports many features described in Section 6, including packed evaluations of circuits and can be extended into a threshold homomorphic encryption scheme. In terms of security, the BFV homomorphic evaluation algorithms can be augmented to provide evaluation privacy.

For details on the implementation of the full BFV scheme, we refer the reader to [11], [23].

(c) **Comparison between BGV and BFV**

When implementing HE schemes, there are many choices which can be made to optimize performance for different architectures and different application scenarios. This makes a direct comparison of these schemes quite challenging. A paper by Costache and Smart [19] gives some initial comparisons between BGV, B/FV and two of the schemes described below: [39] and [33]/[37]. A paper by Kim and Lauter [27] compares the performance of the BGV and YASHE schemes in the context of applications. Since there is further ongoing work in this area, we leave this comparison as an open research question.

1.4 The GSW Scheme and Bootstrapping

Currently, the most practical homomorphic encryption schemes only allow to perform bounded depth computations. These schemes can be transformed into fully homomorphic ones (capable of arbitrary computations) using a "bootstrapping" technique introduced by Gentry [G09], which essentially consists of a homomorphic evaluation of the decryption algorithm given the encryption of the secret key. Bootstrapping is a very time-consuming operation and improving on its efficiency is still a very active research area. So, it may still not be ready for standardization, but it is the next natural step to be considered.

Bootstrapping using the BGV or BFV schemes requires assuming that lattice problems are computationally hard to approximate within factors that grow *super-polynomially* in the lattice dimension n. This is a stronger assumption than the inapproximability within *polynomial* factors required by standard (non-homomorphic) lattice-based public key encryption.

In [GSW13], Gentry, Sahai and Waters proposed a new homomorphic encryption scheme (still based on lattices) that offers a different set of trade-offs than BGV and BFV. An important feature of this scheme is that it can be used to bootstrap homomorphic encryption based on the assumption that lattice problems are hard to approximate within polynomial factors. Here we briefly describe the GSW encryption and show how both its security and applicability to bootstrapping are closely related to LWE encryption, as used by the BGV and BFV schemes. So, future standardization of bootstrapping (possibly based on the GSW scheme) could build on the current standardization effort.

For simplicity, we focus on secret key encryption, as this is typically enough for applications to bootstrapping. The GSW secret key encryption scheme (or, more specifically, its secret key, ring-based variant presented in [6, 21]) can be described as follows:

- GSW.Keygen(params):

  This is essentially the same as the key generation procedure of the BGV or BFV schemes, taking a similar set of security parameters, and producing a random ring element S which serves as a secret key.
- GSW.SecEncrypt(S,M):

  Choose an uniformly random vector A in $R^{2\log(q)}$, a small random vector E (with entries chosen independently at random from the error distribution), and output the ciphertext $C = (A, A * S + E) + M * G$ where $G = [I, 2I, \ldots, 2^{k-1}I]$ is a gadget matrix consisting of $k = \ log\ (q)$ copies of the 2×2 identity matrix I (over the ring), scaled by powers of 2.

We note that there are other possibilities for choosing the gadget matrix G above (for example the constants $2, 4, \ldots, 2^{k-1}$ can be replaced by others). Other choices may be described in future documents.

We omit the description of the decryption procedure, as it is not needed for bootstrapping. Notice that:

- The secret key generation process is the same as most other LWE-based encryption schemes, including BGV and BFV.
- The encryption procedure essentially consists of $2\ log\ (q)$ independent application of the basic LWE/BGV/BFV encryption: choose random key elements a and e, and outputs $(a, as + e + m)$, but applied to scaled copies of the message

$m = 2^i M$. (The even rows of the GSW ciphertext encrypt the message as $(a + m, as + e)$, but this is just a minor variant on LWE encryption, and equivalent to it from a security standpoint.)
- Security rests on the standard LWE assumption, as used also by BGV and BFV, which says that the distribution $(A, A * S + E)$ is pseudorandom.

So, GSW can be based on LWE security estimates similar to those used to instantiate the BGV or BFV cryptosystems.

In [GSW13] it is shown how (a public key version of) this cryptosystem supports both addition and multiplication, without the need for an evaluation key, which has applications to identity-based and attribute-based homomorphic encryption. Later, in [BV14] it was observed how the GSW multiplication operation exhibits an asymmetric noise growth that can be exploited to implement bootstrapping based on the hardness of approximating lattice problems within polynomial factors. Many subsequent papers (e.g., [6, 18, 21, 24]) improve on the efficiency of [BV14], but they all share the following features with [BV14]:

- They all use variants of the GSW encryption to implement bootstrapping.
- Security only relies on the hardness of approximating lattice problems within polynomial factors.
- They are capable of bootstrapping any LWE-based encryption scheme, i.e., any scheme which includes an LWE encryption of the message as part of the ciphertext. LWE-based schemes include BGV, BFV and GSW.

In particular, GSW can be used to implement the bootstrapping procedure for BGV and BFV and turn them into fully homomorphic encryption (FHE) schemes.

1.5 Other Schemes

Yet Another Somewhat Homomorphic Encryption ([39]) is similar to the BGV and B/FV schemes and offers the same set of features.

The scheme NTRU/Lopez-Alt-Tromer-Vaikuntanathan ([37]/[33]) relies on the NTRU assumption (also called the “small polynomial ratios assumption”). It offers all the features of BGV and BFV, and in addition, also offers an extension that supports multi-key homomorphism. However, it must be used with a much wider error distribution than the other schemes that are described in this document (or else it becomes insecure), and therefore it should only be used with a great deal of care. This standard does not cover security for these schemes.

Another scheme, called HEAAN, with plaintext type approximate numbers, was recently proposed by Cheon, Kim, Kim and Song [CKKS17]. This scheme is not described here, but we expect future version of this standard to include it.

1.6 Additional Features & Discussion

(a) **Distributed HE**

Homomorphic Encryption is especially suitable to use for multiple users who may want to run computations on an aggregate of their sensitive data. For the setting of multiple users, an additional property which we call threshold-HE is desirable. In threshold-HE the key-generation algorithms, encryption and decryption algorithms are replaced by a distributed-key-generation (DKG) algorithm, distributed-encryption (DE) and distributed-decryption (DD) algorithms. Both the distributed-key-generation algorithm and the distributed-decryption algorithm are executed via an interactive process among the participating users. The evaluation algorithms EvalAdd, EvalMult, EvalMultConst, EvalAddConst, and Refresh remain unchanged.

We will now describe the functionality of the new algorithms.

We begin with the distributed-key-generation (DKG) algorithm to be implemented by an interactive protocol among t parties $p_1, \ldots, p_t$. The DKG algorithm is a randomized algorithm. The inputs to DKG are: security parameter, number of parties t, and threshold parameter d. The output of DKG is a vector of secret keys $s = (s_1, \ldots., s_t)$ of dimension t and a public evaluation key EK where party p_i receives (EK,s_i). We remark that party p_i does not receive s_j for $i \neq j$ and that party i should maintain the secrecy of its secret key s_i.

Next, the distributed-encryption (DE) algorithm is described. The DE algorithm is a randomized algorithm which can be run by any party p_i. The inputs to DE run by party p_i are: the secret key s_i and the plaintext M. The output of DE is a ciphertext C

Finally, we describe the distributed-decryption (DD) algorithm to be implemented by an interactive protocol among a subset of the t parties $p_1, \ldots, p_t$. The DD algorithm is a randomized algorithm.

The inputs to DD are a subset of secret keys $s = (s_1, \ldots., s_t)$, the threshold parameter d, and a ciphertext C. In particular, every participating party p_i provides the inputs_i. The ciphertext C can be provided by any party. The output of DD is: plaintext M.

The correctness requirement that the above algorithms should satisfy is as follows.

If at least d of the parties correctly follow the prescribed interactive protocol that implements the DD decryption algorithm, then the output of the decryption algorithm will be correct.

The security requirement is for semantic security to hold as long as fewer than d parties collude adversarially.

An example usage application for (DKG,DE,DD) is for two hospitals, $t = 2$ and $d = 2$ with sensitive data sets M_1 and M_2(respectively) who want to compute some analytics F on the joint data set without revealing anything about M_1 and M_2 except for what is revealed by $F(M_1, M_2)$.

In such a case the two hospitals execute the interactive protocol for DKG and obtain their respective secret keys s_1 and s_2 and the evaluation key EK. They each use DE on secret key s_i and data M_i to produce ciphertext Ci. The evaluation algorithms on C1, C2 and the evaluation key EK allow the computation of a ciphertext C which is an encryption of $F(M_1, M_2)$. Now, the hospitals execute the interactive protocol DD using their secret keys and ciphertext C to obtain $F(M_1, M_2)$.

(b) **Active Attacks**

One can consider stronger security requirements beyond semantic security. For example, consider an attack on a client that holds data M and wishes to compute $F(M)$ for a specified algorithm F, and wants to outsource the computation of $F(M)$ to a cloud, while maintaining the privacy of M. The client encrypts M into ciphertext C and hands C to the cloud server. The server is supposed to use the evaluation algorithms to compute a ciphertext C' which is an encryption of $F(M)$ and return this to the client for decryption.

Suppose that instead the cloud computes some other C" which is the encryption of $G(M)$ for some other function G. This may be problematic to the client as it would introduce errors of potentially significant consequences. This is an example of an active attack which is not ruled out by semantic security.

Another, possibly even more severe attack, is the situation where the adversary somehow gains the ability to decrypt certain ciphertexts or glean some information about their content (perhaps by watching the external behavior of the client after decrypting them). This may make it possible to the attacker to mount (perhaps limited) *chosen-ciphertext attacks*, which may make it possible to compromise the security of encrypted data. Such attacks are not addressed by the semantic security guarantee, countering them requires additional measures beyond the use of homomorphic encryption.

(c) **Evaluation Privacy**

A desirable additional security property beyond semantic security would be that the ciphertext C hides which computations were performed homomorphically to obtain C. We call this security requirement *Evaluation Privacy.*

For example, suppose a cloud service offers a service in the form of computing a proprietary machine learning algorithm F on the client's sensitive data. As before, the client encrypts its data M to obtain C and sends the cloud C and the evaluation key EK. The cloud now computes C' which is an encryption of $F(M)$ to hand back to the client. Evaluation privacy will guarantee that C' does not reveal anything about the algorithm F which is not derivable from the pair $(M, F(M))$. Here we can also distinguish between semi-honest and malicious evaluation privacy depending on whether the ciphertext C is generated correctly according to the Encrypt algorithm.

A weaker requirement would be to require evaluation privacy only with respect to an adversary who does not know the secret decryption key. This may be relevant for an adversary who intercepts encrypted network traffic.

(d) Key Evolution

Say that a corpus of ciphertexts encrypted under a secret key SK is held by a server, and the client who owns SK realizes that SK may have been compromised.

It is desirable for an encryption scheme to have the following *key evolution* property. Allow the client to generate a new secret key SK' which replaces SK, a new evaluation key EK', and a transformation key TK such that: the server, given only TK and EK', may convert all ciphertexts in the corpus to new ciphertexts which (1) can be decrypted using SK' and (2) satisfy semantic security even for an adversary who holds SK.

Any sufficiently homomorphic encryption scheme satisfies the key evolution property as follows. Let TK be the encryption of SK under SK'. Namely, TK is a ciphertext which when decrypted using secret key SK' yields SK. A server given TK and EK', can convert a ciphertext C in the corpus into C' by homomorphically evaluating the decryption process. Security follows from semantic security of the original homomorphic encryption scheme.

(e) Side Channel Attacks

Side channel attacks consider adversaries who can obtain partial information about the secret key of an encryption scheme, for example by running timing attacks during the execution of the decryption algorithm. A desirable security requirement from an encryption scheme is resiliency against such attacks, often referred to as *leakage resiliency*. That is, it should be impossible to violate semantic security even in presence of side channel attacks. Naturally, leakage resilience can hold only against limited information leakage about the secret key.

An attractive feature of encryption schemes based on intractability of integer lattice problems, and in particular known HE schemes based on intractability of integer lattice problems, is that they satisfy leakage resilience to a great extent. This is in contrast to public-key cryptosystems such as RSA.

(f) Identity Based Encryption

In an identity based encryption scheme it is possible to send encrypted messages to users without knowing either a public key or a secret key, but only the identity of the recipient where the identity can be a legal name or an email address.

This is possible as long as there exists a trusted party (TP) that publishes some public parameters PP and holds a master secret key MSK. A user with identity X upon authenticating herself to the TP (e.g. by showing a government issued ID), will receive a secret key SK_X that the user can use to decrypt any ciphertext that was sent to the identity X. To encrypt message M to identity X, one needs only to know the public parameters PP and X.

Identity based homomorphic encryption is a variant of public key homomorphic encryption which may be desirable.

Remark: A modification of GSW supports identity based homomorphic encryption.

2 Homomorphic Encryption Standard Section 2: Recommended Security Parameters

2.1 Hard Problems

This section describes the computational problems whose hardness form the basis for the security of the homomorphic encryption schemes in this document. Known security reductions to other problems are not included here. **Section 2.2** below describes the best currently known attacks on these problems and their concrete running times. **Section 2.5** below recommends concrete parameter choices to achieve various security levels against currently known attacks.

(a) **The Learning with Errors (LWE) Problem**

The LWE problem is parametrized by four parameters (n, m, q, χ), where n is a positive integer referred to as the "dimension parameter", m is "the number of samples", q is a positive integer referred to as the "modulus parameter" and χ is a probability distribution over rational integers referred to as the "error distribution".

The LWE assumption requires that the following two probability distributions are computationally indistinguishable:

Distribution 1. Choose a uniformly random matrix $m \times n$ matrix A, a uniformly random vector s from the vector space Z_q^n, and a vector e from Z^m where each coordinate is chosen from the error distribution χ. Compute $c := As + e$, where all computations are carried out modulo q. Output (A, c).

Distribution 2. Choose a uniformly random $m \times n$ matrix A, and a uniformly random vector c from Z_q^m. Output (A, c).

The error distribution χ can be either a discrete Gaussian distribution over the integers, a continuous Gaussian distribution rounded to the nearest integer, or other distributions supported on small integers. We refer the reader to Section 2.5 for more details on particular error distributions, algorithms for sampling from these distributions, and the associated security implications. We also mention that the secret vector s can be chosen from the error distribution.

(b) **The Ring Learning with Errors (RLWE) Problem**

The RLWE problem can be viewed as a specific case of LWE where the matrix A is chosen to have special algebraic structure. RLWE is parametrized by parameters (m, q, χ) where m is the number of samples, as in the LWE problem above, q is a positive integer (the "modulus parameter") and χ is a probability distribution over the ring $R = Z[X]/f(X)$ (the "error distribution").

The RLWE assumption requires that the following two probability distributions are computationally indistinguishable:

Distribution 1. Choose $m + 1$ uniformly random elements $s, a_1, \ldots, a_m$ from the ring R/qR, and m more elements $e_1, \ldots, e_m$ from the ring R chosen from the error distribution χ. Compute $b_i := sa_i + e_i$, all computations carried out over the ring R/qR. Output $\{(a_i, b_i) : i = 1, \ldots m\}$.

Distribution 2. Choose $2m$ uniformly random elements $a_1, \ldots, a_m, b_1, \ldots, b_m$ from the ring R/qR. Output $\{(a_i, b_i) : i = 1, \ldots m\}$.

The error distribution χ must be supported on "small" elements in the ring R (with geometry induced by the canonical embedding). For RLWE, it is important to use an error distribution that matches the specific ring R. See Section 2.5 for more details on the error distributions, algorithms for sampling from these distributions, and the associated security implications. Here too, the secret element s can be be chosen from the error distribution.

(c) **The Module Learning with Errors (RLWE) Problem**

We mention here that there is a general formulation of the learning with errors problem that captures both LWE and RLWE, as well as many other settings. In this formulation, rather than n-vectors over Z (as in LWE) or 1-vectors over $R = Z[x]/f(X)$ (as in RLWE), we work with vectors of dimension n_1 over a ring of dimension n_2, where the security parameter is related to $n_1 \cdot n_2$. This document only deals with LWE and RLWE, but we expect future versions to be extended to deal with more settings.

2.2 Attacks on LWE and Their Complexity

We review algorithms for solving the LWE problem and use them to suggest concrete parameter choices. The schemes described above all have versions based on the LWE and the RLWE assumptions. When the schemes based on RLWE are instantiated with error distributions that match the cyclotomic rings (as described later in this document), we do not currently have attacks on RLWE that are meaningfully better than the attacks on LWE. The following estimates and attacks refer to attacks on the LWE problem with the specified parameters.

Much of this section is based on the paper by Albrecht, Player, and Scott [4], the online *Estimator* tool which accompanies that paper, and [1, 3]. Indeed, we reuse text from those works here. Estimated security levels in all the tables in this section were obtained by running the *Estimator* based on its state in March 2018. The tables in this section give the best attacks (in terms of running time expressed in log_2) among all known attacks as implemented by the *Estimator* tool. As attacks or implementations of attacks change, or as new attacks are found, these tables will need to be updated. First, we describe all the attacks which give the best running times when working on parameter sizes in the range which are interesting for Homomorphic Encryption.

The LWE problem asks to recover a secret vector $s \in Z_q^n$, given a matrix $A \in Z_q^{m\times n}$ and a vector $c \in Z_q^m$ such that $As + e = c \; mod \; q$ for a short error vector $e \in Z_q^m$ sampled coordinate-wise from an error distribution χ. The decision variant of LWE asks to distinguish between an LWE instance (A, c) and uniformly random $(A, c) \in Z_q^{m\times n} \times Z_q^m$. To assess the security provided by a given set of parameters m, χ, q, two strategies are typically considered.

The *primal* strategy finds the closest vector to c in the integral span of columns of A mod q, i.e. it solves the corresponding *Bounded Distance Decoding* problem (BDD) directly, and is explained in [31] and [28].

(a) **Primal (uSVP variant)**

Assume that $m > n$, i.e. the number of samples available is greater than the dimension of the lattice. Writing $[I_n|A^{'}]$ for the reduced row echelon form of $A^T \in Z_q^{n\times m}$ (with high probability and after appropriate permutation of columns), this task can be reformulated as solving the *unique Shortest Vector Problem* (uSVP) in the $m + 1$ dimensional q-ary lattice

$$\Lambda = Z^{m+1} \cdot \begin{pmatrix} I_n & A' & 0 \\ 0 & qI_{m-n} & 0 \\ c^T & & t \end{pmatrix}.$$

by Kannan's embedding, with embedding factor t.

The lattice Λ has volume $t \cdot q^{m-n}$ and contains a vector of norm $\sqrt{\| e\|^2 + t^2}$ which is unusually short, i.e. the gap between the first and second Minkowski minimum $\lambda_2(\Lambda)/\lambda_1(\Lambda)$ is large. If the secret vector s is also short, there is a second established embedding reducing LWE to uSVP. By inspection, it can be seen that the vector $(\nu s| e| 1)$, for some $\nu \neq 0$, is contained in the lattice Λ of dimension $d = m + n + 1$

$$\Lambda = \left\{ x \in (\nu Z)^n \times Z^{m+1} | \, x \cdot \left(\frac{1}{\nu} A \, |I_m| - c \right)^\top \equiv 0 \bmod q \right\},$$

where ν allows to balance the size of the secret and the noise. An $(n + m + 1) \times (n + m + 1)$ basis M for Λ can be constructed as

$$M = \begin{pmatrix} \nu I_n & -A^\top & 0 \\ 0 & qI_m & 0 \\ 0 & c & 1 \end{pmatrix}.$$

To find short vectors, lattice reduction can be applied. Thus, to establish the cost of solving an LWE instance, we may consider the cost of lattice reduction for solving uSVP. In [5] it is predicted that e can be found if:

$$\sqrt{\beta/d} \parallel (e|1) \parallel \approx \sqrt{\beta}\sigma \leq \delta_0^{2\beta-d} Vol(\Lambda)^{1/d},$$

where δ_0 denotes the root Hermite factor achievable by BKZ, which depends on β which is the block size of the underlying blockwise lattice reduction algorithm. This prediction was experimentally verified in [3].

(b) **Primal by BDD Enumeration (decoding).**

This attack is due to Lindner and Peikert [31]. It starts with a sufficiently reduced basis, e.g., using BKZ in block size β, and then applies a modified version of the recursive *Nearest Plane* algorithm due to Babai [8]. Given a basis B and a target vector t, the Nearest Plane algorithm finds a vector such that the error vector lies in the fundamental parallelepiped of the Gram-Schmidt orthogonalization (GSO) of B.

Lindner and Peikert note that for a BKZ-reduced basis B, the fundamental parallelepiped is long and thin, by the Geometric Series Assumption (GSA) due to Schnorr that the GSO of a BKZ-reduced basis decay geometrically and this makes the probability that the Gaussian error vector e falls in the corresponding fundamental parallelepiped very low. To improve this success probability, they "fatten" the parallelepiped by essentially scaling its principal axes. They do this by running the Nearest Plane algorithm on several distinct planes at each level of recursion. For a Gaussian error vector, the probability that it falls in this fattened parallelepiped is expressed in terms of the scaling factors and the lengths of the GSO of B. This can be seen as a form of pruned CVP enumeration [32].

The run time of the Nearest Planes algorithm mainly depends on the number of points enumerated, which is the product of the scaling factors. The run time of the basis reduction step depends on the quality of the reduced basis, expressed, for instance, by the root Hermite factor δ_0. The scaling factors and the quality of the basis together determine the success probability of the attack. Hence to maximize the success probability, the scaling factors are determined based on the (predicted) quality of the BKZ-reduced basis. There is no closed formula for the scaling factors. The *Estimator* uses a simple greedy algorithm to find these parameters due to [31], but this is known to not be optimal. The scaling factors and the quality of the basis are chosen to achieve a target success probability and to minimize the running time (by balancing the running time of BKZ reduction and the final enumeration step).

(c) **Dual**. The dual strategy finds short vectors in the lattice

$$q\Lambda^* = \left\{ x \in Z_q^m \mid x \cdot A \equiv 0 \bmod q \right\},$$

i.e. it solves the *Short Integer Solutions* problem (SIS). Given such a short vector v, we can decide if an instance is LWE by computing $(v,c) = (v,e)$ which is short whenever v and e are sufficiently short [36].

We must however ensure that (v,e) indeed is short enough, since if is too large, the (Gaussian) distribution of will be too flat to distinguish from random. Following

([31]), for an LWE instance with parameters n, α, q and a vector v of length $\|v\|$ such that $v \cdot A \equiv 0 \ mod \ q$, the advantage of distinguishing (v, e) from random is close to

$$\exp\left(-\pi(\| \ v \ \| \cdot \alpha)^2\right).$$

To produce a short enough v, we may again call a lattice-reduction algorithm. In particular, we may call the BKZ algorithm with block size β. After performing BKZ-β reduction the first vector in the transformed lattice basis will have norm $\delta_0^m \cdot Vol(q\Lambda^*)^{1/m}$. In our case, the expression above simplifies to $\| \ v \ \| \approx \delta_0^m \cdot q^{n/m}$ whp. The minimum of this expression is attained at $m = \sqrt{\frac{nlogq}{\log\delta_0}}$ [36]. The attack can be modified to take small or sparse secrets into account [1].

Lattice Reduction Algorithm: BKZ
BKZ is an iterative, block-wise algorithm for basis reduction. It requires solving the SVP problem (using sieving or enumeration, say) in a smaller dimension β, the block size. First, the input lattice Λ is LLL reduced, giving a basis $b_0, \ldots, b_{n-1}$. For $0 \leq i < n$, the vectors $b_i, \ldots, b_{min(i+\beta-1, n-1)}$ are projected onto the orthogonal complement of the span of $b_0, \ldots b_{i-1}$; this projection is called a local block. In the local block, we find a shortest vector, view it as a vector $b \in \Lambda$ of and perform LLL on the list of vectors $b_i, \ldots, b_{min(i+\beta-1, n-1)}, b$ to remove linear dependencies. We use the resulting vectors to update $b_i, \ldots, b_{min(i+\beta-1, n-1)}$. This process is repeated until a basis is not updated after a full pass.

There have been improvements to BKZ, which are collectively referred to BKZ 2.0 (see [17] for example). There are currently several different assumptions in the literature about the cost of running BKZ, distinguished by how conservative they are, the "sieve" and "ADPS16" cost models, as explained below. In our use of the *Estimator* we rely on the cost model in the "sieve" implementation, as it seems the most relevant to the parameter sizes which we use for Homomorphic Encryption.

(a) **Block Size.**

To establish the required block size β, we solve

$$\log\delta_0 = \log\left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right) \cdot \frac{1}{2(\beta - 1)}$$

for β, see the PhD Thesis of Yuanmi Chen [14] for a justification of this.

(b) **Cost of SVP.**

Several algorithms can be used to realize the SVP oracle inside BKZ. Asymptotically, the fastest known algorithms are sieving algorithms. The fastest, known classical algorithm runs in time

$$2^{0.292\beta + o(\beta)} \text{ [9]}.$$

The fastest, known quantum algorithm runs in time

$$2^{0.265\beta+o(\beta)} \text{ [29]} .$$

The "sieve" estimate approximates $o(\beta)$ by 16.4 based on some experimental evidence in [9]. The "ADPS16" from [5] suppresses the $o(\beta)$ term completely. All times are expressed in elementary bit operations.

(c) **Calls to SVP.**

The BKZ algorithm proceeds by repeatedly calling an oracle for computing a shortest vector on a smaller lattice of dimension β. In each "tour" on a d-dimensional lattice, d such calls are made, and the algorithm is typically terminated once it stops making sufficient progress in reducing the basis. Experimentally, it has been established that only the first few tours make significant progress [14], so the "sieve" cost model assumes that one BKZ call costs as much as $8d$ calls to the SVP oracle. However, it seems plausible that the cost of these calls can be amortized across different calls, which is why the "ADPS16" cost model from [5] assumes the cost of BKZ to be the same as *one* SVP oracle call, which is a strict *underestimate* of the attack cost.

(d) **BKZ Cost.** In summary:

sieve
a call to BKZ-β costs $8d \cdot 2^{0.292\beta\,+\,16.4}$ operations classically and $8d \cdot 2^{0.265\beta\,+\,16.4}$ operations quantumly.

ADPST16
a call to BKZ-β costs $2^{0.292\beta}$ operations classically and $2^{0.265\beta}$ operations quantumly.

We stress that both cost models are very conservative, and that no known implementation of lattice reduction achieves these running times. Furthermore, these estimates completely ignore memory consumption, which, too, is $2^{\Theta(\beta)}$.

(e) **Calls to BKZ.**

To pick parameters, we normalize running times to a fixed success probability. That is, all our expected costs are for an adversary winning with probability 51%. However, as mentioned above, it is often more efficient to run some algorithm many times with parameters that have a low probability of success instead of running the same algorithm under parameter choices which ensure a high probability of success.

2.3 The Arora-Ge Attack

The effectiveness of the lattice attacks above depend on the size of the error and the modulus q, in contrast Arora and Ge described in [7] an attack whose complexity depends only on the size of the error and poly-logarithmically on the modulus q. Very roughly, for dimension n and noise of magnitude bounded by some positive integer d in each coordinate, the attack uses $n^{O(d)}$ samples and takes $n^{O(d)}$ operations in the ring of integers modulo q. For the relevant range of parameters for homomorphic encryption, this attack performs worse than the above lattice attacks even when the error standard deviation is a small constant (e.g., $\sigma = 2$).

2.4 Algebraic Attacks on Instances of Ring-LWE

In practice the ring R is taken to be the ring of integers in a cyclotomic field, $R = Z[x]/\Phi_k(x)$, where Φ_k is the cyclotomic polynomial for the cyclotomic index k, and the degree of Φ_k is equal to the dimension of the lattice, $n = \phi(k)$ where ϕ is the Euler totient function.

As mentioned above, for ring-LWE the choice of the error distribution matters, and there are known examples of natural high-entropy error distributions that are insecure to use in certain rings. Such examples were first given in [22] and [15], and were subsequently improved in [12], [13], and [16]. For example, in [15] it was shown that for a prime cyclotomic index m, choosing the coefficients of the error polynomial $e \in Z[x]/\Phi_k(x)$ independently at random from a distribution of standard deviation sufficiently smaller than $\sqrt{k}$, can sometimes make this instance of RLWE easy to solve. It is therefore crucial to select an error distribution that "matches" the ring at hand.

The form of the error distribution for general cyclotomic rings was investigated, e.g., in [34, 35, 38, DD12]. We summarize these results in Section 2.5 below, but the current document only specifies concrete parameters for power-of-two cyclotomic fields, i.e. $k = 2^\ell$. We expect future versions of this document to extend the treatment also for generic cyclotomic rings. We stress that when the error is chosen from a sufficiently wide and "well spread" distributions that match the ring at hand, we do not have meaningful attacks on RLWE that are better than LWE attacks, regardless of the ring. For power-of-two cyclotomics, it is sufficient to sample the noise in the polynomial basis, namely choosing the coefficients of the error polynomial $e \in Z[x]/\Phi_k(x)$ independently at random from a very "narrow" distribution.

2.5 Secure Parameter Selection for Ring LWE

Specifying a Ring-LWE scheme for encryption requires specifying a ring, R, of a given dimension, n, along with a ciphertext modulus q, and a choice for the error distribution and a choice for a secret distribution.

Ring In practice, we take the ring R to be a cyclotomic ring $R = Z[x]/\Phi_k(x)$, where m is the cyclotomic index and $n = \phi(k)$ is the ring dimension. For example, a power of 2 cyclotomic with index $k = 2^\ell$ is $R = Z[x]/(x^{k/2} + 1)$, of degree $n = k/2 = 2^{\ell - 1}$.

Error Distribution, Power-of-Two Cyclotomics For the special case of power-of-two cyclotomics, it is safe to sample the error in the polynomial basis, namely choosing the coefficients of the error polynomial $e(x) \in Z[x]/(x^{k/2} + 1)$ independently at random from a very "narrow" distribution. Specifically, it is sufficient to choose each coefficient from a Discrete Gaussian distribution (or even rounded continuous Gaussian distribution) with a small constant standard deviation σ. Selecting the error according to a Discrete Gaussian distribution is described more often in the literature, but choosing from a rounded continuous Gaussian is easier to implement (in particular when timing attacks need to be countered).

The LWE attacks mentioned above, however, do not take advantage of the shape of the error distribution, only the standard deviation. Moreover, the security reductions do not apply to the case where the error standard deviation is a small constant and would instead require that the error standard deviation grows at least as n^{ϵ} for some constant $\epsilon > 1/2$ (or even $\epsilon > 3/4$). The analysis of the security levels given below relies on running time estimates which assume that the shape of the error distribution is Gaussian.

The standard deviation that we use below is chosen as $\sigma = 8/\sqrt{2\pi} \approx 3.2$, which is a value that is used in many libraries in practice and for which no other attacks are known. (Some proposals in the literature suggest even smaller values of σ.) Over time, if our understanding of the error standard deviation improves, or new attacks are found, the standard deviation of the error may have to change.

Error Distribution, General Cyclotomics For non-power-of-two cyclotomics, choosing a spherical error in the polynomial basis (i.e., choosing the coefficients independently) may be insecure. Instead, there are two main methods of choosing a safe error polynomial for the general case:

- The method described in [DD12] begins by choosing an "extended" error polynomial $e' \in \boldsymbol{Q}[X]/(\Theta_k(x))$, where $\Theta_k(x) = x^k - 1$ if k is odd, and $x^{k/2} + 1$ if k is even. The rational coefficients of e' are chosen independently at random from the continuous Gaussian of standard deviation $\sigma\sqrt{k}$ (for the same σ as above), and with sufficient precision, e.g., using double float numbers. Then, the error is computed as

$$e = Round\left(e' \bmod \Phi_k(x)\right)$$

- The method described in [CP16] chooses an error of the form $e = Round(e' \cdot t_k)$, where $t_k \in R$ is a fixed ring element (see below), and e' is chosen from a spherical continuous Gaussian distribution in the canonical embedding, of standard deviation σ (for the same σ as above). One way of sampling such error polynomial is to choose a spherical e' in the canonical embedding, then multiply by t_k and round, but there are much more efficient methods of sampling the error (cf. [20]).

Note that the error so generated may not be very small, since t_k is not tiny. It is possible to show that e is somewhat small, but moreover it is shown in [20] that homomorphic computations can be carried out to maintain the invariant that e/t_k is small (rather than the invariant that e itself is small).

The element t_k is a generator of the "different ideal", and it is only defined up to multiplication by a unit, so implementations have some choice for which specific element to use. One option is $t_k(x) = \Phi'_k(x)$ (i.e., the formal derivative of $\Phi_k(x)$), but other options may lead to more efficient implementations.

We stress that this document does not make recommendations on the specific parameters to use for non-power-of-two cyclotomic rings, in particular Tables 1 and 2 below only apply to power-of-two cyclotomic rings.

Secret Key For most homomorphic encryption schemes, not only the error but also the secret key must be small. The security reductions ensure that choosing the key from the same distribution as the error does not weaken the scheme. However, for many homomorphic encryption schemes (including BGV and B/FV), choosing an even smaller secret key has a significant performance advantage. For example, one may choose the secret key from the *ternary distribution* (i.e., each coefficient is chosen uniformly from $\{-1, 0, 1\}$). In the recommended parameters given below, we present tables for three choices of secret-distribution: uniform, the error distribution, and ternary.

In some extreme cases, there is a reason to choose an even smaller secret key, e.g., one with sparse coefficient vector. However, we will not present tables for sparse secrets because the security implications of using such sparse secrets is not well understood yet. We expect to specify concrete parameters for sparse secret keys in future versions of this standard.

Number of Samples For most of the attacks listed in the tables below, the adversary needs a large number of LWE samples to apply the attack with maximum efficiency. Collecting many samples may be feasible in realistic systems, since from one ring-LWE sample one can extract many "LWE-like" samples. The evaluation keys may also contain some samples.

Sampling Methods All the error distributions mentioned above require choosing the coefficients of some initial vector independently at random from either the discrete or the continuous Gaussian with some standard deviation $\sigma > 0$. Sampling from a continuous Gaussian with small parameter is quite straightforward, but sampling from a discrete Gaussian distribution is harder. There are several known methods to sample from a discrete Gaussian, including rejection sampling, inversion sampling, Discrete Zuggurat, Bernoulli-type, Knuth-Yao and Von Neumann-type. For efficiency, we recommend the Von Neumann-type sampling method introduced by Karney in [26].

Constant-Time Sampling In some of the aforementioned sampling methods, the time it takes to generate one sample could leak information about the actual sample. In many applications, it is therefore important that the entire error-sampling process is constant-time. This is easier to do when sampling from the continuous Gaussian distribution, but harder for the discrete Gaussian. One possible method is to fix some upper bound $T > 0$ such that sampling all the n coordinates e_i sequentially without interruption takes time less than T time with overwhelming probability. Then after these samples are generated, using time t, we wait for $(T - t)$ time units, so that the entire error-generating time always takes time T. In this way, the total time does not reveal information about the generated error polynomial.

Tables of Recommended Parameters

In practice, in order to implement homomorphic encryption for a particular application or task, the application will have to select a dimension n, and a ciphertext modulus q, (along with a plaintext modulus and a choice of encoding which are not discussed here). For that reason, we give pairs of (n, q) which achieve different security levels for each n. In other words, given n, the table below recommends a value of q which will achieve a given level of security (e.g. 128 bits) for the given error standard deviation $\sigma \approx 3.2$.

We have the following tables for 3 different security levels, 128-bit, 192-bit, and 256-bit security, where the secret follows the uniform, error, and ternary distributions. For applications, we give values of n from $n = 2^k$ where $k = 10, \dots, 15$. We note that we used commit (560525) of the LWE-estimator of [4], which the authors continue to develop and improve. The tables give estimated running times (in bits) for the three attacks described in Section 5.1: uSVP, dec (decoding attack), and dual.

Post-Quantum Security The BKZ.qsieve model assumes access to a quantum computer and gives lower estimates than BKZ.sieve. In what follows, we give tables of recommended ("Post-quantum") parameters which achieve the desired levels of security against a quantum computer. We also present tables computed using the "quantum" mode of the BKZ.ADPS16 model, which contain more conservative parameters.

Table 1 Cost model = BKZ.sieve

distribution	n	security level	logq	uSVP	dec	dual
uniform	1024	128	29	131.2	145.9	161.0
		192	21	192.5	225.3	247.2
		256	16	265.8	332.6	356.7
	2048	128	56	129.8	137.9	148.2
		192	39	197.6	217.5	233.7
		256	31	258.6	294.3	314.5
	4096	128	111	128.2	132.0	139.5
		192	77	194.7	205.5	216.4
		256	60	260.4	280.4	295.1
	8192	128	220	128.5	130.1	136.3
		192	154	192.2	197.5	205.3
		256	120	256.5	267.3	277.5
	16384	128	440	128.1	129.0	133.9
		192	307	192.1	194.7	201.0
		256	239	256.6	261.6	269.3
	32768	128	880	128.8	129.1	133.6
		192	612	193.0	193.9	198.2
		256	478	256.4	258.8	265.1
error	1024	128	29	131.2	145.9	141.8
		192	21	192.5	225.3	210.2
		256	16	265.8	332.6	300.5
	2048	128	s56	129.8	137.9	135.7
		192	39	197.6	217.5	209.6
		256	31	258.6	294.3	280.3
	4096	128	111	128.2	132.0	131.4
		192	77	194.7	205.5	201.5
		256	60	260.4	280.4	270.1
	8192	128	220	128.5	130.1	130.1
		192	154	192.2	197.5	196.9
		256	120	256.5	267.3	263.8
	16384	128	440	128.1	129.3	130.2
		192	307	192.1	194.7	196.2
		256	239	256.6	261.6	264.5
	32768	128	883	128.5	128.8	130.0
		192	613	192.7	193.6	193.4
		256	478	256.4	258.8	257.9

(continued)

Table 1 (continued)

distribution	n	security level	logq	uSVP	dec	dual
(-1, 1)	1024	128	27	131.6	160.2	138.7
		192	19	193.0	259.5	207.7
		256	14	265.6	406.4	293.8
	2048	128	54	129.7	144.4	134.2
		192	37	197.5	233.0	207.8
		256	29	259.1	321.7	273.5
	4096	128	109	128.1	134.9	129.9
		192	75	194.7	212.2	198.5
		256	58	260.4	292.6	270.1
	8192	128	218	128.5	131.5	129.2
		192	152	192.2	200.4	194.6
		256	118	256.7	273.0	260.6
	16384	128	438	128.1	129.9	129.0
		192	305	192.1	196.2	193.2
		256	237	256.9	264.2	259.8
	32768	128	881	128.5	129.1	128.5
		192	611	192.7	194.2	193.7
		256	476	256.4	260.2	258.2

Table 2 Cost model = BKZ.qsieve

distribution	n	security level	logq	uSVP	dec	dual
uniform	1024	128	27	132.2	149.3	164.5
		192	19	199.3	241.6	261.6
		256	15	262.9	341.1	360.8
	2048	128	53	128.1	137.6	147.6
		192	37	193.6	215.8	231.4
		256	29	257.2	297.9	316.6
	4096	128	103	129.1	134.2	141.7
		192	72	193.8	206.2	217.2
		256	56	259.2	281.9	296.5
	8192	128	206	128.2	130.7	136.6
		192	143	192.9	199.3	207.3
		256	111	258.4	270.8	280.7
	16384	128	413	128.2	129.0	132.7
		192	286	192.1	195.3	201.4
		256	222	257.2	263.1	270.6
	32768	128	829	128.1	128.4	130.8
		192	573	192.0	193.3	197.5
		256	445	256.1	259.0	265.2

(continued)

Table 2 (continued)

distribution	n	security level	logq	uSVP	dec	dual
error	1024	128	27	132.2	149.3	144.5
		192	19	199.3	241.6	224.0
		256	15	262.9	341.1	302.3
	2048	128	53	128.1	137.6	134.8
		192	37	193.6	215.8	206.7
		256	29	257.2	297.9	281.4
	4096	128	103	129.1	134.2	133.1
		192	72	193.8	206.2	201.8
		256	56	259.2	281.9	270.4
	8192	128	206	128.2	130.7	130.1
		192	143	192.9	199.3	198.5
		256	111	258.4	270.8	266.6
	16384	128	413	128.2	129.0	130.1
		192	286	192.1	195.3	196.6
		256	222	257.2	263.1	265.8
	32768	128	829	128.1	128.4	129.8
		192	573	192.0	193.3	192.8
		256	445	256.1	259.0	260.4
$(-1, 1)$	1024	128	25	132.6	165.5	142.3
		192	17	199.9	284.1	222.2
		256	13	262.6	423.1	296.6
	2048	128	51	128.6	144.3	133.4
		192	35	193.5	231.9	205.2
		256	27	257.1	327.8	274.4
	4096	128	101	129.6	137.4	131.5
		192	70	193.7	213.6	198.8
		256	54	259.7	295.2	270.6
	8192	128	202	129.8	130.7	128.0
		192	141	192.9	202.5	196.1
		256	109	258.3	276.6	263.1
	16384	128	411	128.2	129.5	129.0
		192	284	192.0	196.8	193.7
		256	220	257.2	265.8	260.7
	32768	128	827	128.1	128.7	128.4
		192	571	192.0	194.1	193.1
		256	443	256.1	260.4	260.4

Organizers

Kristin Lauter	klauter@fb.com
Vinod Vaikuntanathan	vinod.nathan@gmail.com

Contributors

Martin Albrecht	martinralbrecht@googlemail.com
Melissa Chase	melissac@microsoft.com
Hao Chen	haoche@fb.com
Jintai Ding	jintai.ding@gmail.com
Shafi Goldwasser	shafi@theory.csail.mit.edu
Sergey Gorbunov	sgorbunov100@gmail.com
Shai Halevi	shaih@alum.mit.edu
Jeffrey Hoffstein	hoffsteinjeffrey@gmail.com
Satya Lokam	Satya.Lokam@microsoft.com
Kim Laine	kim.laine@microsoft.com
Daniele Micciancio	daniele@cs.ucsd.edu
Dustin Moody	dustin.moody@nist.gov
Travis Morrison	tmo@vt.edu
Amit Sahai	amitsahai@gmail.com

References

1. Albrecht, M. R. (2017). On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In J. Coron & J. B. Nielsen (Eds.), *EUROCRYPT 2017, part ii* (Vol. 10211, pp. 103–129). Springer, Heidelberg.
2. Martin R. Albrecht, Robert Fitzpatrick, and Florian Gopfert: *On the Efficacy of Solving LWE by Reduction to Unique-SVP*. In Hyang-Sook Lee and Dong-Guk Han, editors, ICISC 13, volume 8565 of LNCS, pages 293–310. Springer, November 2014.
3. Albrecht, M. R., Göpfert, F., Virdia, F., & Wunderer, T. (2017). Revisiting the expected cost of solving uSVP and applications to LWE. In T. Takagi & T. Peyrin (Eds.), *ASIACRYPT 2017, part i* (Vol. 10624, pp. 297–322). Springer, Heidelberg.
4. Martin R. Albrecht, Rachel Player and Sam Scott. *On the concrete hardness of Learning with Errors*. Journal of Mathematical Cryptology. Volume 9, Issue 3, Pages 169–203, ISSN (Online) 1862–2984, October 2015.

5. Alkim, E., Ducas, L., Pöppelmann, T., & Schwabe, P. (2016). Post-quantum key exchange - A new hope. In T. Holz & S. Savage (Eds.), *25th USENIX security symposium, USENIX security 16* (pp. 327–343). USENIX Association. Retrieved from https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim
6. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 297–314.
7. Sanjeev Arora and Rong Ge. *New algorithms for learning in the presence of errors.* In ICALP, volume 6755 of Lecture Notes in Computer Science, pages 403–415. Springer, 2011.
8. László Babai: *On Lovász' lattice reduction and the nearest lattice point problem,* Combinatorica, 6(1):1–3, 1986.
9. Becker, A., Ducas, L., Gama, N., & Laarhoven, T. (2016). New directions in nearest neighbor searching with applications to lattice sieving. In R. Krauthgamer (Ed.), *27th soda* (pp. 10–24). ACM-SIAM. https://doi.org/10.1137/1.9781611974331.ch2
10. Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan. *(Leveled) fully homomorphic encryption without bootstrapping.* In ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. Pages 309–325.
11. Zvika Brakerski. *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP,* In CRYPTO 2012. Pages 868–886.
12. W. Castryck, I. Iliashenko, F. Vercauteren, *Provably weak instances of ring-lwe revisited.* In: Eurocrypt 2016. vol. 9665, pp. 147–167. Springer (2016a)
13. W. Castryck, I. Iliashenko, F. Vercauteren, *On error distributions in ring-based LWE.* LMS Journal of Computation and Mathematics 19(A), 130–145 (2016b) 7.
14. Chen, Y. (2013). *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe* (PhD thesis). Paris 7.
15. Hao Chen, Kristin Lauter, Katherine E. Stange, *Attacks on the Search RLWE Problem with Small Errors*, SIAM J. Appl. Algebra Geometry, Society for Industrial and Applied Mathematics, Vol. 1, pp. 665–682. (2017) https://eprint.iacr.org/2015/971
16. Hao Chen, Kristin Lauter, Katherine E. Stange. *Security Considerations for Galois Non-dual RLWE Families, SAC 2016:*Selected Areas in Cryptography – SAC 2016 Lecture Notes in Computer Science, Vol. 10532. Springer pp 443–462.
17. Y. Chen, P.Q. Nguyen. *BKZ 2.0: Better Lattice Security Estimates*. In: Lee D.H., Wang X. (eds) Advances in Cryptology – ASIACRYPT 2011. ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073. Springer, Berlin, Heidelberg.
18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33.
19. Ana Costache, Nigel P. Smart, *Which Ring Based Somewhat Homomorphic Encryption Scheme is Best?* Topics in Cryptology - CT-RSA 2016, LNCS, volume 9610, Pages 325–340.
20. Eric Crockett and Chris Peikert. $\Lambda \circ \lambda$: Functional Lattice Cryptography. In ACM-CCS 2016.
21. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640.
22. Yara Elias, Kristin Lauter, Ekin Ozman, Katherine E. Stange, *Provably weak instances of Ring-LWE,* CRYPTO 2015
23. J. Fan and F. Vercauteren. *Somewhat practical fully homomorphic encryption*. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.pdf
24. Gama, N., Izabachène, M., Nguyen, P.Q., Xie, X.: Structural lattice reduction: generalized worst-case to average-case reductions. In: EUROCRYPT 2016, https://eprint.iacr.org/2014/283.pdf
25. C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In CRYPTO 2013 (Springer).
26. C.F.F. Karney, *Sampling Exactly from the Normal Distribution*. ACM Transactions on Mathematical Software, 42, Article No. 3.

27. Miran Kim and Kristin Lauter, Private Genome Analysis through Homomorphic Encryption, *BioMedCentral Journal of Medical Informatics and Decision Making* 2015 15 (Suppl 5): S3.
28. Kim Laine and Kristin Lauter, Key Recovery for LWE in Polynomial Time. https://eprint.iacr.org/2015/176
29. Laarhoven, T. (2015). *Search problems in cryptography: From fingerprinting to lattice sieving* (PhD thesis). Eindhoven University of Technology.
30. Laarhoven T., Mosca M., van de Pol J. (2013) *Solving the Shortest Vector Problem in Lattices Faster Using Quantum Search.* In: Gaborit P. (eds) Post-Quantum Cryptography. PQCrypto 2013. Lecture Notes in Computer Science, vol 7932. Springer, Berlin, Heidelberg.
31. Richard Lindner and Chris Peikert: *Better key sizes (and attacks) for LWE-based encryption.* In Topics in Cryptology – CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, Aggelos Kiayias, Editor, volume 6558 of LNCS, pages 319–339.
32. Liu, M., & Nguyen, P. Q. (2013). Solving BDD by enumeration: An update. In E. Dawson (Ed.), *CT-rsa 2013* (Vol. 7779, pp. 293–309). Springer, Heidelberg. https://doi.org/10.1007/978-3-642-36095-4_19
33. A. Lopez-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In STOC, pages 1219–1234, 2012.
34. Vadim Lyubashevsky, Chris Peikert, and Oded Regev : *On Ideal Lattices and Learning with Errors over Rings.* Journal of the ACM (JACM), Volume 60, Issue 6, November 2013a, Article No. 43.
35. Vadim Lyubashevsky, Chris Peikert, and Oded Regev : *A toolkit for ring-LWE cryptography.* Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2013b.
36. Micciancio, D., & Regev, O. (2009). Lattice-based cryptography. In D. J. Bernstein, J. Buchmann, & E. Dahmen (Eds.), *Post-quantum cryptography* (pp. 147–191). Berlin, Heidelberg, New York: Springer, Heidelberg.
37. J. Hoffstein, J. Pipher, and J. H. Silverman. *NTRU: A ring-based public key cryptosystem.* In J. Buhler, editor, ANTS, volume 1423 of Lecture Notes in Computer Science, pages 267–288. Springer, 1998.
38. C. Peikert, *How Not to Instantiate Ring-LWE*, in SCN'16, volume 9841 of LNCS, Springer, 2016.
39. Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. *Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme*, in IMA CC 2013. http://eprint.iacr.org/2013/075.pdf

Software References for 7 Homomorphic Encryption Libraries

40. https://github.com/Microsoft/SEAL
41. https://github.com/shaih/HElib
42. https://github.com/CryptoExperts/FV-NFLlib
43. https://git.njit.edu/groups/palisade
44. https://github.com/vernamlab/cuHE
45. https://github.com/vernamlab/cuFHE
46. https://github.com/kimandrik/HEAAN
47. https://tfhe.github.io/tfhe/

Part III
Applications of Homomorphic Encryption

Privacy-Preserving Data Sharing and Computation Across Multiple Data Providers with Homomorphic Encryption

Juan Troncoso-Pastoriza, David Froelicher, Peizhao Hu, Asma Aloufi, and Jean-Pierre Hubaux

1 Motivation

Many data processing scenarios have to deal with sensitive data split in multiple silos. Supporting joint computation over distributed datasets can enable more meaningful, representative, and statistically significant results that would otherwise be hidden within the isolated datasets. For example, multiple hospitals, biobanks, and university labs want to contribute their patient data for a joint study in which computations are performed in distributed (Fig. 1) or centralized (Fig. 2) manner. However, they want to preserve the confidentiality of their data and avoid any leakage to the other parties or to external attackers.

This is especially relevant in health and financial environments, where the access and availability of extremely sensitive data for training more accurate prediction and analysis models is limited. In this whitepaper, we exemplify our solutions by focusing on the health scenario.

2 System Models and Use Cases

Homomorphic encryption enables computations to be performed directly on encrypted data without decrypting it first. This capability enables joint computations to be performed without accessing the sensitive data in clear-text from the participating data providers.

J. Troncoso-Pastoriza (✉) · D. Froelicher · J.-P. Hubaux
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
e-mail: juan.troncoso-pastoriza@epfl.ch; david.froelicher@epfl.ch; jean-pierre.hubaux@epfl.ch

P. Hu · A. Aloufi
Rochester Institute of Technology, Rochester, NY, USA
e-mail: Peizhao.Hu@rit.edu; ama9000@rit.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_3

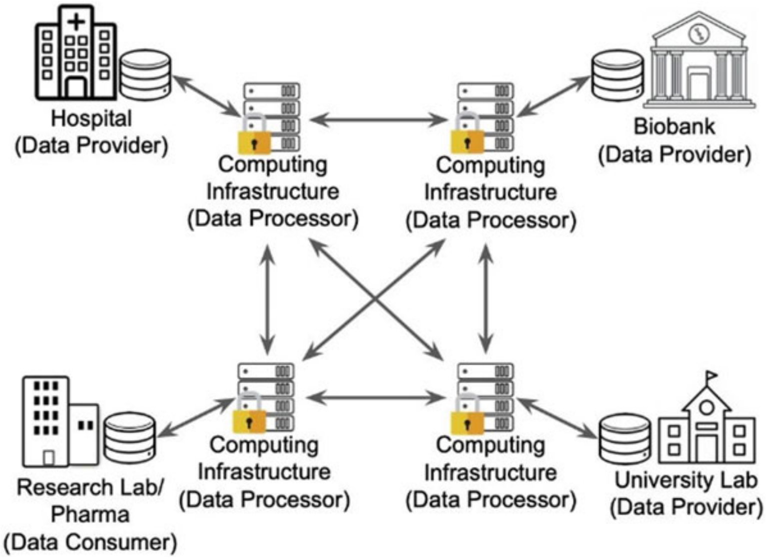


Fig. 1 Distributed homomorphic computing

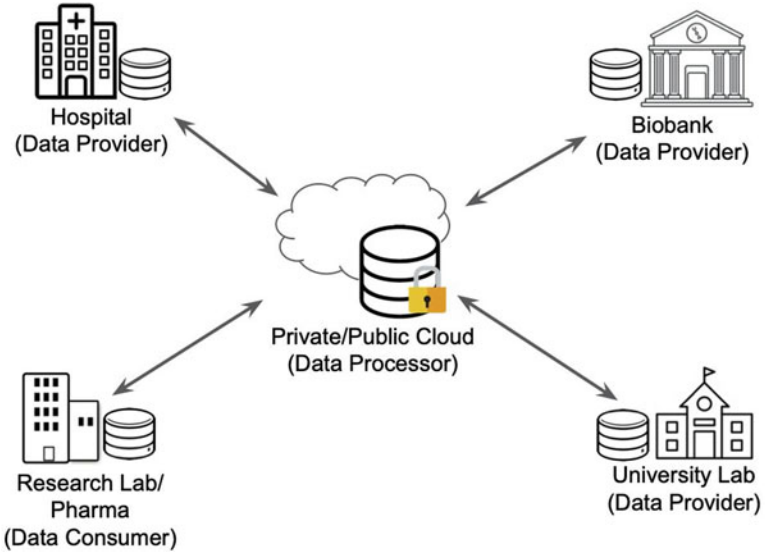


Fig. 2 Centralized homomorphic computing

As illustrated in Figs. 1 and 2, the data providers encrypt their data before making it available to the computing infrastructure (be it distributed or centralized) for the desired homomorphic computation.

We consider two computation models:

- Distributed scenario, depicted in Fig. 1, where the computation is "sent to the data", and each of the data providers makes their respective computing infrastructure available for performing the joint computation on-site.
- Centralized scenario, depicted in Fig. 2, where the encrypted data and computations are delegated to a private or public cloud computing infrastructure. This computation model leverages the economic efficiency and ubiquitous accessibility of the cloud infrastructures to support joint computations.

Both models support encrypted queries sent by a client (the data consumer), who wants to extract insights from the datasets contributed by the participating data providers.

Let us consider a use case in collaborative medical research. Medical data is susceptible to being used for research purposes and, eventually, for precision medicine [1–4]. This sensitive data is usually siloed in independent clinical sites due to the inherent sensitivity of the data itself, the data sharing and transfer restrictions imposed by regulations (e.g., GDPR [5] in Europe, HIPAA [6] in US), and/or policies at the clinical sites.

Managing this data in a privacy-conscious way would accelerate and automate IRB (Institutional Review Board) review processes for sharing sensitive (and personally identifiable) medical data with external researchers. Review processes can take several weeks, if not months, to permit researchers to access the data, and these processes are often denied because the necessary privacy and security guarantees cannot be provided. Furthermore, the ability to collectively analyze the shared data across a multitude of institutions can improve the results of the analysis, producing more accurate and precise results. This can help advance the understanding of rare diseases and the personalized effect of determined drugs and treatments on diverse patient populations. Hence, secure medical-data sharing holds an immense potential to increase the effectiveness and reduce the costs of healthcare.

According to forecasts by the Organisation for Economic Co-operation and Development (OECD), the Health expenditure will outpace GDP growth over the next 15 years in almost every OECD country [7]. Health spending per capita will grow at an average annual rate of 2.7% across the OECD and will reach 10.2% of GDP by 2030, up from 8.8% in 2018 [7]. Additionally, the R&D expenditure on health and medical sciences in OECD countries ranges from an average of 4% of the GDP in Western Pacific countries to a 17.4% in Eastern Mediterranean countries [8]. With this data, there is no question that the health sector is one of the biggest drivers of the world economy, and any improvements in cost-effectiveness of treatments and diagnoses can yield a significant impact in economical and societal terms. In a recent white paper [9], the World Economic Forum (WEF) analyzes the trade-offs between data protection and innovation in health research. The report acknowledges the benefits of federated data systems in dealing with health-related data, and their potential as a solution for the current "search of new models of data access that enable them to capture the value of such data and provide better patient care and drive innovation". Among the benefits, it is worth highlighting the access to richer insights, the reduction of operational and financial costs, and the facilitation of cross-border data sharing. The report calls for the development of open technical standards that can help build and deploy federated data systems. The Value in Healthcare project launched by the WEF has also identified [10] the need to "move from fragmented healthcare systems to unprecedented cooperation among all stakeholders", calling for new standards to address the "protection of data security and patient privacy". Homomorphic encryption (HE), and in particular, the Multiparty Homomorphic Encryption (MHE) variant [11], is one of the key enablers for data sharing and federated data analytics, and as such, it can respond to the aforementioned challenges in the health sector. In the centralized model, HE has the potential to give data subjects and data controllers confidence in data privacy when they delegate data and computations to the cloud. The distributed model provides high confidence in data ownership because computations take place in-situ, whereas the centralized model is more suitable for secure data computations at-rest after data being offloaded and stored on the cloud.

Within this landscape, this document describes protocols that provide the ability for (i) clinicians to find patients with similar (possibly identifying) characteristics to those of the patient under examination in order to take more informed decisions in terms of diagnosis and treatment, (ii) researchers to securely query massive amounts of distributed clinical and genetic data to obtain descriptive statistics indispensable for generating new hypotheses in clinical research studies, and (iii) researchers to efficiently and privately train and execute machine learning models on encrypted data partitioned across a network of clinical sites, therefore guaranteeing privacy and regulatory compliance [12].

3 Stakeholders and Functionalities

There are four different roles of each participating entity in the described scenario:

- **Data providers** (data controllers): They hold personal data from individuals (patients, research subjects), that they want to contribute to a joint computation together with data from other providers. In the medical scenario, data providers are represented by clinics, hospitals, sequencing facilities, research labs, cancer registries, etc.
- **Data processors**: Data processors (or Storage and Processing Units, SPU) provide computing resources and infrastructure to process the input data. They can be either in-house facilities under the control of one of the data providers, or pay-per-use services. In the medical scenario, processors can be the data providers (e.g., hospitals) themselves, private/public clouds, or HPC (High-Performance Computing) infrastructures.

- **Data consumers** (queriers/clients): They are the authorized entities that want to query the system to obtain some insights or results on a predefined analysis, performed on the totality of the database or in a subset thereof. In the medical scenario, consumers are researchers, data analysts, insurance companies, and pharma companies, among others.
- **External attackers**: They are unauthorized entities that are interested in learning information about the private data records. In the medical scenario, external attackers are represented by eavesdroppers and hackers.

4 Functionality Goals

The purpose of the system is to enable data consumers to securely explore and process the input data hosted by the various data providers in the network, with the following functionalities:

- *Exploration and selection* [13]: An authorized researcher should be able to obtain the number of records (and/or their pseudonyms) across data providers who satisfy a set of inclusion/exclusion criteria, optionally grouped by parameters such as age, gender and ethnicity. More formally, the system must support SQL-like queries such as

```
SELECT COUNT(records) / SELECT records
           FROM distributed_dataset
           WHERE criteria_i AND/OR criteria_j
           AND/OR ...
           GROUP BY criteria_k;
```

- *Distributed* [14–16] *or Centralized* [17] *analysis*: An authorized client should be able to get the result of a function (e.g., aggregation, statistics, training of a statistical or machine learning model) on a set of data (cohort), previously identified by the *Exploration and selection* process, and that can be either centralized or horizontally or vertically partitioned across the different providers.

  Potential functions comprise, but are not limited to: sum, count, frequency count, average, variance, standard deviation, cosine similarity, min/max, and/or, set intersection/union, training and evaluation of generalized linear models (e.g., linear, logistic, multinomial logistic regressions [15], feed-forward and convolutional neural networks [16]), and classification with other machine learning models (e.g., random forests [17], deep neural networks [18, 19]).

5 Threat Models and Security Requirements

Data providers and data processors are assumed to be semi-honest. They will follow the protocol specification, but might try to infer data from their view of the encrypted inputs/outputs and system execution, such as launching possible side-channel attacks. Data consumers and external attackers might be malicious. In particular, the data consumers might try to infer further data from the evaluated results or to send multiple queries to probe the structure of the input data or computation model. The external attackers might try to eavesdrop on the secure communication channel.

We also assume that collusion is possible between all players, as long as there is at least one non-colluding data processor and one non-colluding data provider. All parties are computationally-bounded adversaries, but can have quantum-computing capabilities (i.e., the solution must be quantum-safe).

The system must always provide the following security and privacy requirements:

- **Trust Decentralization**: There should be no single point of failure in the system.
- **End-to-end Data Protection**: The confidentiality of the input data must be protected at rest, in transit and during computation. The data must be encrypted by the data provider before being sent to a data processor, and the result of the computed operation can be decrypted only by the data consumer issuing the query.

Depending on the access privileges of the querier, the system should be able to also provide the following optional features (either or both of them):

- **Unlinkability**: The client must not be able to trace a query response back to its original data provider.
- **Result Obfuscation**: The query result must be obfuscated in order to achieve formal privacy guarantees (e.g., differential privacy) and prevent re-identification.

6 High-Level Workflow

The core query processing protocol comprises the steps sketched below:

1. **Setup**: Selection of cryptographic parameters.
2. **Key management**: Individual and collective key generation and distribution, definition of usage and revocation policies.
3. **Data preparation**:
   - We assume that data is structured and interoperable (coded under the same structure, format and ontology).
   - ETL (Extract, Transform, Load) process: The data providers extract the data from their respective databases, pre-process and encrypt them with the collective key.
   - The data providers load the encrypted data in the corresponding data processor (centralized or distributed).

4. **Query generation**: An authorized client generates the query that contains the task description, i.e., either the filtering criteria for exploration or the computation definition and the involved attributes for analysis.
5. **Joint computation**: This is the main step where the data processor(s) compute the results of the query (the matching records for cohort exploration, and the features/predictions of the trained/evaluated model, for data analysis).
6. **Results preparation**:
   - Result obfuscation: If required by the access level of the researcher, the data processors homomorphically obfuscate the results (both for cohort exploration and data analysis).
   - Distributed results shuffling: If required by the access level of the researcher, the data processors execute an interactive protocol to randomly shuffle the obtained encrypted results.
   - Distributed results re-encryption: The data processors collectively re-encrypt the obtained results from the collective key to the key of the authorized researcher, and send the resulting encryptions to the latter.
7. **Results decryption**: The authorized client obtains the encrypted results under his/her individual key and decrypts them.

7 Example Protocol Instantiations

We introduce now three protocol instantiations, that serve as representative examples of how the high-level workflow can be implemented in a variety of scenarios. Namely, we present:

- Distributed Data Discovery (MedCo). See [13] for further details
- Centralized Data Analysis. See [17] for details.
- Distributed Data Analysis. See [14–16] for details.

For a comprehensive review of HE schemes and techniques that support homomorphic evaluation on ciphertext encrypted under multiple keys, readers can refer to the survey paper [20]. The distributed protocols in this whitepaper are based on the paradigm of multiparty homomorphic encryption, introduced by Mouchet et al. [11].

7.1 Distributed Data Discovery (MedCo)

The protocol used in MedCo for distributed data discovery is sketched in Fig. 3. Each step of the protocol is described in detail below.

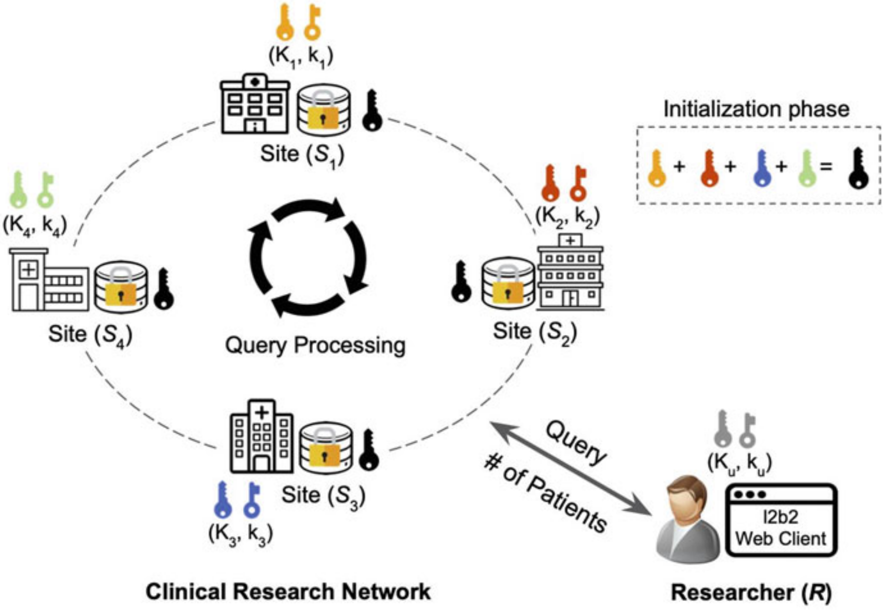


Fig. 3 Medco system model and query workflow

Setup

The data providers and data processors agree on a set of cryptographic parameters that guarantee the desired level of security (HomomorphicEncryption.org [21] security standard in this volume Part 2) for further details on recommended parameter sets for homomorphic cryptosystems.

Initialization

During the initialization of MedCo, each data processor (SPU_i) generates a pair of cryptographic keys (k_i, K_i) of a distributed homomorphic cryptosystem (at least additively homomorphic), along with a secret s_i. Then, all SPUs additively combine their public keys in order to generate a single collective public key K that will be used by the different clinical sites to encrypt the data to be outsourced.

ETL Process

During the data-ingestion phase, i.e., extraction transformation and loading (ETL) phase, each clinical site extracts patient-level data from its private EHR system or clinical research data warehouse, and transforms the data in order to fit the data model used in MedCo. This star-schema data model is based on the Entity-Attribute-Value (EAV) concept also used by widespread clinical research systems such as i2b2 [22, 23], where clinical and genetic observations (or "facts") about patients (e.g., diagnosis, medications, procedures, laboratory values and genetic variants) are stored in a narrow table called "fact" table. Observations are encoded by ontology concepts from an extensible set of medical terminologies, e.g., the International Classification of Disease (ICD) or the US National Drug Code (NDC). In this data model, four other "dimension" tables further describe the patients' data and meta-data. For example, the "patient dimension" table contains pseudonymized demographic information of the patients, and the "visit dimension" table stores information about the visit, such as its date and time and the type of provider.

In such a data model, the information that clinical sites want to protect from potential semi-honest adversaries at the storage and processing units is represented by the mapping between the patients in the database and the set of their clinical and genomic observations stored in the "fact" table that are considered to be sensitive or identifying. In order to protect such mapping, each site separately performs the following three steps:

1. **Generation of Dummy Patients**: Each site generates a set of dummy patients with plausible clinical observations specifically chosen so that the distribution of observations across patients in the "fact" table is as close as possible to the uniform distribution. This is required to avoid frequency attacks on the deterministically encrypted data used when matching query terms. To distinguish the real patients from the dummies, each site also generates a binary flag to be appended to the demographic information in the "patient dimension" table. This flag is set to 1 for real patients and to 0 for dummy patients.
2. **Data Encryption**: In order to break the link between the patients and their sensitive observations in the "fact" table, each site encrypts with the collective public key K the set of ontology concepts that encode these observations along with the patients' binary flags. As the used homomorphic cryptosystem is a probabilistic encryption scheme, each clinical site obtains a set of probabilistic ciphertexts that are totally indistinguishable from each other.
3. **Data Loading and Re-Encryption**: After encryption, each site uploads the encrypted data to the selected storage and processing unit that immediately starts a *Distributed Deterministic Re-Encryption (DDR)* protocol (explained below) in which the encrypted concepts are sent across the network of SPUs so that their encryption is switched from probabilistic to deterministic. This re-encryption is necessary for enabling the secure processing of equality-matching queries (as those defined in the *Security and Privacy Goals* section) that otherwise would be extremely inefficient with probabilistic ciphertexts. Due to the presence of dummy patients, even if the deterministic nature of the ciphertexts leaks the equality of the underlying plaintexts, a semi-honest adversary is not able to perform a frequency attack to distinguish ontology concepts based on their frequency distribution. Dummy patients are computationally indistinguishable

from real patients, as long as the patients' binary flags are probabilistically encrypted.

Query Generation

The secure query protocol starts with an authenticated and authorized researcher who wants to obtain either the number of patients or the pseudonyms of the patients who match a set of inclusion/exclusion clinical and genetic criteria across the different clinical sites. In clinical research, this procedure is called "cohort selection." For this purpose, the researcher builds a query by logically combining (i.e., through AND and OR operators) a set of "sensitive" and "non-sensitive" concepts from a common (i.e., shared across the different sites) ontology. The "sensitive" concepts in the query are encrypted with the collective public key K and the query is sent along with the researcher's public key K_r to one of the storage and processing units.

Query Re-encryption

The SPU that receives the query starts a *Distributed Deterministic Re-Encryption (DDR)* protocol in order to switch the encryption of the sensitive concepts in the query from probabilistic to deterministic. We refer the reader to [13, 24] for the detailed explanations of the distributed protocols. Once the DDR protocol is over, the initial SPU broadcasts the deterministic version of the query to the other SPUs in the network.

Local Query Processing

Each SPU locally processes the query by filtering the patients (both dummy and real) in the "patient dimension" table whose observations in the "fact" table (both the unencrypted and the deterministically encrypted ones) match the concepts in the query. If the query requests the list of matching patients' pseudonyms, each SPU returns the list of matching patients' pseudonyms along with the probabilistically encrypted binary flags. If the query requests the number of matching patients, each SPU homomorphically adds the matching-patients' dummy flags and returns the encrypted result $E_K(R_i) = E_K(\sum_{j \in \phi} f_i^j) = \sum_{j \in \phi} E_K(f_i^j)$, where $E_K(f_i^j)$ is the encrypted flag of the j-th patient in site S_i and ϕ is the set of patients matching the query. In the homomorphic summation, the binary flags of the dummy patients have a null contribution (i.e., $E_K(0)$), hence the encrypted final result corresponds to the actual number of real matching patients.

Result Obfuscation

This step is optional and depends on (i) the type of query and (ii) the researcher's privileges. In order to guarantee differential privacy (we refer the reader to [25] for details about the appropriateness of this technology for dynamic query systems), each SPU can obfuscate the encrypted patient counts computed during the previous step by homomorphically adding noise sampled from a Laplacian distribution. More specifically, let ϵ_q be the privacy budget allocated for a given query q and μ be the noise value drawn from a Laplacian distribution with mean 0 and scale $\Delta f/\epsilon_q$, where the sensitivity Δf is equal to 1, due to the query being a count. Then, the encrypted obfuscated query result is obtained as $E_K\left(\widehat{R_i}\right) = E_K\left(R_i + \mu\right) = E_K\left(R_i\right) + E_K\left(\mu\right)$. We note that the query result is released to the researcher only if the researcher's differential privacy budget is enough for such a query, i.e., if $\epsilon_r - \epsilon_q > 0$.

Result Shuffling

This step is also optional and depends, as the previous step, on (i) the type of query and (ii) the researcher's privileges. In order to break the link between the encrypted (potentially obfuscated) query results generated at the different SPUs and the corresponding clinical sites, the SPUs jointly run a *Distributed Shuffling (DS)* protocol [13, 24, 26] on the set of encrypted patient counts. As a result, each SPU receives encrypted counts that might have been generated by another SPU.

Proxy Re-encryption of the Result

The query results securely computed by each SPU are encrypted with the collective key K; to be decrypted by the researcher, each SPU runs a *Distributed Key Switching (DKS)* protocol [13, 24] that involves the other SPUs and switches the encryption of the query results from an encryption with K to an encryption with K_r, the researcher's public key. After this, the newly encrypted query results are sent back to the SPU that initiated the protocol and then on to the researcher.

Decryption

As the query results are encrypted with K_r, the researcher can use the corresponding secret key k_r to decrypt them and obtain the corresponding plaintext values. If the query results are the list of patients' pseudonyms along with the patients' binary flag, the researcher can simply rule out the dummy patients by discarding those who have the flag set to zero.

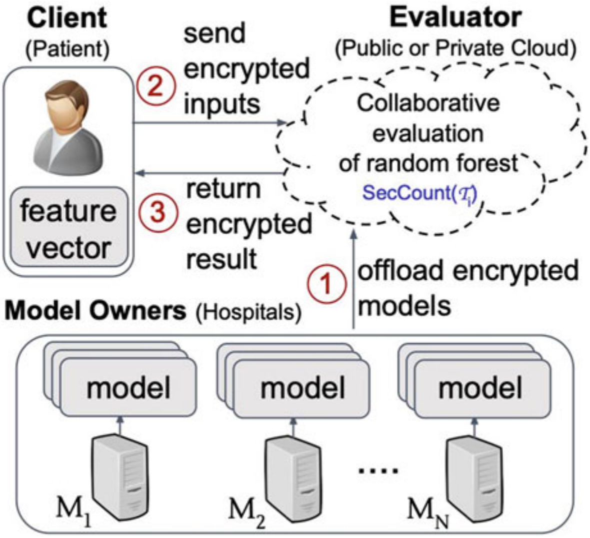
7.2 Centralized Data Analysis (Private Evaluation of Random Forests)

Decision tree is one of the most widely used nonparametric machine learning techniques for classification and regression. The evaluation process is a series of comparisons at each decision node of the tree, which compares the input from a client with the threshold of the node as specified in the model. The Boolean results decide which descendant node to traverse and eventually leads to a leaf node representing a result. Similar to some other machine learning frameworks, relying on a single such tree may incur the model-overfitting problem. A random forest which aggregates the results from individual decision trees can provide more accurate results. The final result is either a list of classification labels together with counts associated with each label, or a classification label that most of the trees agreed on.

Let us assume data providers possess private data on which they train decision tree models for secure classification. In order to obtain results from these privately held models, the client has to send separate requests to each model owner and aggregate the results from these models. Leaving aside the communication overhead caused by the exchange of intermediate results, this naive method reveals the individual decision made by each model owner to the client.

Alternatively, model owners can outsource their models (decision trees) to a third-party evaluator (a public or private cloud infrastructure), as illustrated in Fig. 4. In this case, the client sends the encrypted query to the evaluator who will perform the joint computation.

Fig. 4 Collaborative evaluation of random forests



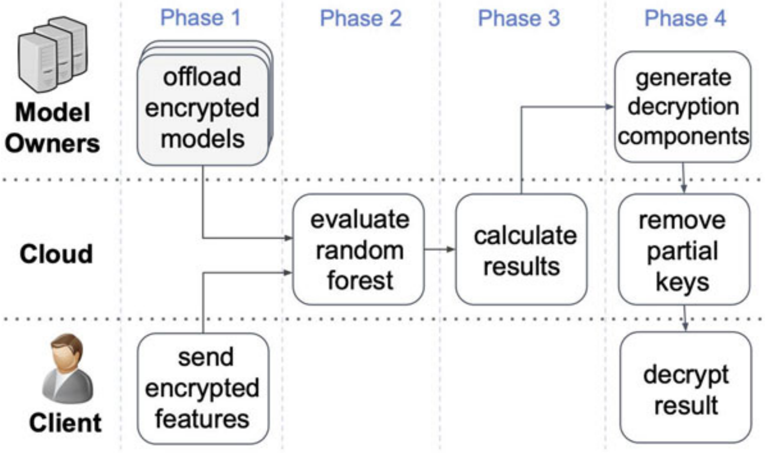


Fig. 5 Phases in the protocol execution

Figure 5 illustrates the four phases of interactions between different parties. In the first phase, each model owner encrypts a set of decision trees, including all the threshold values in their binary format. Delegating the encrypted models to the evaluator can be performed as a one-time setup before servicing the clients.

In the second phase, upon receiving a feature vector encrypted under the key of the client, the evaluator evaluates every decision tree in the entire random forest. Once it is done, each decision tree outputs a class label. The evaluator will perform a secure counting protocol to obliviously aggregate the number of occurrences for each unique class label. The evaluator then sends the class labels with their associated counts to the client.

In the final phase, each model owner will participate in the partial decryption, which sends a decryption component to the evaluator to convert the encrypted result to a ciphertext which is decryptable by the secret key of the client. More details on these steps are described in [17].

7.3 Distributed Data Analysis (Statistical Computation and Training of Machine Learning Models)

The distributed data analysis workflow is similar to the distributed data discovery one and also follows the *high-level workflow* presented before. We highlight here the main differences with respect to the distributed data discovery and we refer the reader to [14–16] for the in-depth description of the protocols, including the training and evaluation of generalized linear models [15] and feed-forward neural networks [16].

For analysis, in the *query generation*, the querier specifies the attributes involved and the intended computation as a *target function* which can be represented as an encodable operation [14]:

$$f(r) \equiv \pi\left(\{\rho\left(r_i\right)\}_{i=1}^{N}\right),$$

where r is the set of all distributed dataset records, r_i is a set of records belonging to the i-th data provider, and π is a polynomial combination (aggregation) of the outputs of the encoding $\rho(.)$. The encodings are defined as locally computed functions on the partitions (r_i) of every clinical site. It is also possible to express an encodable operation as a recursive function:

$$f_k(r) \equiv \pi\left(\{\rho\left(r_i, f_{k-1}(r)\right)\}_{i=1}^{N}\right).$$

During the *Joint computation,* all the data providers locally compute on their data and their corresponding results are then collectively aggregated. In the case of a recursive or iterative computation, this process is repeated for a pre-defined number of times. Afterwards, the *Results Preparation* and *Results Decryption* are performed similarly as in the data discovery workflow.

8 Concluding Remarks

We refer the reader to the publications below for the detailed protocol definitions and for complete performance evaluations of the proposed solutions. For completeness, we summarize some of these results here:

- **Distributed Data Discovery**: In the MedCo system [13], the query response time grows linearly with the number of elements that are retrieved and the number of filtering criteria included in the query, whereas the overhead introduced by the encryption with respect to the a cleartext solution is negligible (a few seconds). MedCo can scale up to billions of data points, hundreds of data providers and queries with hundreds of terms. Even large queries that consist of hundreds of filtering criteria are executed in just a few minutes with an average overhead of only 1% to 3%, compared to an unprotected solution operating on plaintext data.
- **Centralized Data Analysis with Random Forests** [17]: When data providers outsource the evaluation of a random forest to the cloud, the private evaluation has to be performed non-interactively. Because of this requirement, we design a non-interactive secure comparison protocol which outputs a single encrypted bit to support consecutive homomorphic operations. The performance of this secure comparison protocol grows linearly with the bit-length of inputs, number of decision nodes, and maximum depth of trees. The running time grows linearly with the bit-length because more homomorphic multiplications are required. The

increase in tree depth increases the number of decision nodes, which dictates the number of invocations of the secure comparison protocol. Our solution includes parallel algorithms to lower multiplicative depth and achieve significant performance gain by a factor of x7 compared to sequential evaluation.

- **Distributed Data Analysis**: For the statistical computations, the proposed system's execution time [14–16] scales linearly with the number of data providers and the number of attributes involved in the computation. Its execution time is almost independent of the data providers' local datasets sizes. For the training and execution of machine learning models [15, 16], the proposed solution relies on the workload distribution among multiple data providers to efficiently cope with a large number of them, and its execution time is practically independent of this number. Moreover, it accommodates models with a large number of features, as it scales better than linearly with this number.

All the proposed solutions enable, streamline, and facilitate data discovery and analysis in environments in which the data is particularly sensitive and aim at breaking the existing barriers that slow down collaborative medical research and data sharing in general, bringing significant benefits for data sharing, new business opportunities, and the possibility of achieving the promise of personalized and precision medicine.

References

1. J. V. Selby, A. C. Beal, and L. Frank. "The patient-centered outcomes research institute (PCORI) national priorities for research and initial research agenda," JAMA, vol. 307, no. 15, pp. 1583–1584, 2012.
2. Swiss Academies of Arts and Sciences. "Swiss Personalized Health Network," http://www.samw.ch/en/Projects/SPHN.html, last Accessed: July 23, 2019.
3. The Global Alliance for Genomics and Health. "A federated ecosystem for sharing genomic, clinical data," Science, vol. 352, no. 6291, pp. 1278–1280, 2016.
4. "All of us research program," https://allofus.nih.gov/, last accessed: July 23, 2019.
5. EU Parliament. "The EU General Data Protection Regulation (GDPR)," http://www.eugdpr.org/, last Accessed: July 23, 2019.
6. U.S. Department of Health & Human Services. "The health insurance portability and accountability act (HIPAA)," https://www.hhs.gov/hipaa/index.html, last Accessed: July 23, 2019.
7. OECD (2019), Health at a Glance 2019: OECD Indicators, OECD Publishing, Paris, https://doi.org/10.1787/4dd50c09-en
8. Gross domestic R&D expenditure on health (health GERD) as a % of gross domestic product (GDP). World Health Organization. Global Observatory on Health R&D. January 2020. Available online: https://www.who.int/research-observatory/indicators/gerd_gdp/
9. Federated Data Systems: Balancing Innovation and Trust in the Use of Sensitive Data, World Economic Forum, July 2019. Available online: https://www.weforum.org/whitepapers/federated-data-systems-balancing-innovation-and-trust-in-the-use-of-sensitive-data/
10. Value in Healthcare: Mobilizing cooperation for health system transformation, World Economic Forum, February 2018. Available online: https://www.weforum.org/reports/value-in-healthcare-mobilizing-cooperation-for-health-system-transformation/

11. Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, Jean-Pierre Hubaux. “Multiparty Homomorhic Encryption from Ring-Learning-with-Errors,” in Proceedings on Privacy Enhancing Technologies, vol. 4, pp. 291–311, 2021.
12. James Scheibner, Jean Louis Raisaro, Juan Ramón Troncoso-Pastoriza, Marcello Ienca, Jacques Fellay, Effy Vayena, Jean-Pierre Hubaux. “Revolutionizing Medical Data Sharing Using Advanced Privacy Enhancing Technologies: Technical, Legal and Ethical Synthesis”. Journal of Medical Internet Research, vol. 23, No. 2. February 2021, https://doi.org/10.2196/25120
13. J. L. Raisaro, J. R. Troncoso-Pastoriza, M. Misbach, J. S. Sousa, S. Pradervand, Edoardo Missiaglia, Olivier Michielin, Bryan Ford and Jean-Pierre Hubaux, “MedCo: Enabling Secure and Privacy-Preserving Exploration of Distributed Clinical and Genomic Data,” IEEE/ACM Transactions on computational biology and bioinformatics. vol. 16, no. 4, pp. 1328–1341, 1 July-Aug. 2019. https://doi.org/10.1109/TCBB.2018.2854776
14. D. Froelicher, J.R. Troncoso-Pastoriza, J.S. Sousa, and J.P. Hubaux. “Drynx: Decentralized, Secure, Verifiable System for Statistical Queries and Machine Learning on Distributed Datasets,” IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3035–3050, 2020. https://doi.org/10.1109/TIFS.2020.2976612.
15. David Froelicher, Juan Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, Jean-Pierre Hubaux. “Scalable Privacy-Preserving Distributed Learning,” in Proceedings on Privacy Enhancing Technologies, vol. 2, pp. 323–347, 2021.
16. Sav, Sinem, Apostolos Pyrgelis, Juan R. Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. “POSEIDON: Privacy-Preserving Federated Neural Network Learning.” NDSS 2021.
17. Asma Aloufi, Peizhao Hu, Harry W.H. Wong, and Sherman S.M. Chow. “Blindfolded Evaluation of Random Forests with Multi-Key Homomorphic Encryption,” in IEEE Transactions on Dependable and Secure Computing (TDSC). Sept 2019.
18. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in International Conference on Machine Learning, pp. 201–210. 2016.
19. Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. “CryptoDL: Towards Deep Learning over Encrypted Data.” In Annual Computer Security Applications Conference (ACSAC 2016), Los Angeles, California, USA, vol. 11. 2016.
20. Asma Aloufi, Peizhao Hu, Yongsoo Song, and Kristin Lauter. “Computing Blindfolded on Data Homomorphically Encrypted under Multiple Keys: An Extended Survey.” https://arxiv.org/abs/2007.09270
21. Homomorphic Encryption Standardization Group. https://homomorphicEncryption.org
22. S.N. Murphy, G. Weber, M. Mendis, V. Gainer, H.C. Chueh, S. Churchill, and I. Kohane. “Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2),” Journal of the American Medical Informatics Association, vol.17, no.2, pp.124–130, 2010
23. B. D. Athey, M. Braxenthaler, M. Haas, and Y. Guo. “tranSMART: an open source and community-driven informatics and data sharing platform for clinical and translational research,” AMIA Summits on Translational Science Proceedings, vol. 2013, p. 6, 2013.
24. D. Froelicher, P. Egger, J. S. Sousa, J. L. Raisaro, Z. Huang, C. Mouchet, B. Ford, and J.-P. Hubaux. “UnLynx: A decentralized system for privacy-conscious data sharing,” in Proceedings on Privacy Enhancing Technologies, vol. 4, pp. 152–170, 2017.
25. MedCo – Legal perspective. Available online at https://medco.epfl.ch
26. C. A. Neff. “Verifiable mixing (shuffling) of ElGamal pairs.” VHTi Technical Document, VoteHere, Inc, 2003.

Secure and Confidential Rule Matching for Network Traffic Analysis

Dimitar Jetchev and Alistair Muir

1 Introduction

The homomorphic encryption standardization group HomomorphicEncryption.org [1] is an open consortium of industry participants and academics working together to standardize an encryption technology called homomorphic encryption.

Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. This essentially means that analysis on encrypted data can be performed as if that data was in plaintext.

Homomorphic encryption holds tremendous promise in a number of applications in government, health, intelligence and in private sector contexts.

The purpose of this document is to illustrate a specific problem that is applicable in the private sector and intelligence communities alike and how this problem can be addressed through the application of Homomorphic encryption.

1.1 Motivation and Business Problem

It is increasingly important for governments, private sector organizations and the intelligence community to detect potential cyber threats on their networks. One of

D. Jetchev (✉)
Inpher Inc., Lausanne, Switzerland
e-mail: dimitar@inpher.io

A. Muir
Vanteum, Sydney, NSW, Australia
e-mail: alistair@vanteum.io

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_4

the key growth areas in this market is the area of Advanced Network Traffic Analysis (NTA) which is where patterns of behavior, methods and techniques used by cyber bad actors are learned over time and applied to network traffic to detect and identify these potential threats.

The global market for advanced NTA is expected to reach $3.7 billion by 2025, rising at a market growth of 18.3% CAGR between now and then. One of the challenges that faces this market is that cyber security specialists and intelligence agencies have built proprietary and, in some cases, classified sets of rules to detect the signature of cyber threat activity. In a cybersecurity context, this proprietary or classified information could describe the behaviors, methods and techniques used by actors whose identity is sensitive.

It is possible for this information, or a part of this is information to be encoded with enough precision to detect and monitor threat actors' presence in network traffic and system telemetry, and thereby identify them via their cyber patterns.

The goal of this application of homomorphic encryption is to evaluate those rules in untrusted environments without revealing either the signatures themselves or the network traffic matching those signatures.

Such a system would allow the provisioning of classified and/or proprietary cyber-security signatures in appliances that could be deployed in insecure and unclassified networks such as in private sector organizations, government or in critical infrastructure networks.

2 Threat Model

A threat model is essentially a structured representation of all the information that affects the security of an application. In essence, it is a view of the application and its environment through security glasses. Threat modeling is a process for capturing, organizing, and analyzing all of this information.

We assume, similarly to the case of standard intrusion-detection algorithms on plaintext, that the computing environment in which the traffic analysis is to be run (client side **C**) is **honest-but-curious** in the following sense:

- Even if the network is compromised, it does not prevent the accurate execution of the intrusion-detection analytics.
- It will run the exact algorithms that the solution provider **(P)** requires without malicious tampering of the computations.
- It will run the analysis on an untampered version of the client's network packet data.
- Any entity on the network (including adversaries) can store and analyze all input and output of the analysis along with any intermediate data generated during the computation.

3 Protocol

The protocol evaluates rules against network traffic to detect and monitor threat actors in a secure and confidential manner such that neither the rules nor the traffic that matches them is revealed to a non-authorized party.

This protocol will use homomorphic encryption to accomplish the goal within the above honest-but-curious threat model. Depending on the sensitivity of the rule set and the trust model of the provider it may also be possible to utilize a trusted environment for testing the rule set and standard encryption to secure the results.

3.1 Client

The client **C** has its network traffic analyzed via a rule-based checker developed by the provider **P**. The **C** will receive feedback and recommendations from **P** based on the results of the rule application to its network traffic.

3.2 Solution Provider

The solution provider **P** is the party that develops the set of rules to be applied to the network traffic (packets) on the side of the client **C**. It owns the set of rules (updates the occasionally), analyzes the results of the rules applied to client's network packets data, and provides (non-real time) feedback to the client based on the analysis.

3.3 Rule Sets

The basis of the threat detection and analysis being provided to **C** is **P**'s rule set. For general network traffic analysis, a rule typically consists of the following:

- **Action:** determines the outcome when a packet matches a rule.
- **Header:** rule information pertaining to the packet header (i.e., the protocol, IP addresses, ports, and direction of communication for the rule).
- **Rule options**: defines the specifics of the rule.

In order to keep **P**'s proprietary rule set confidential, for this protocol the action of the rules is restricted to being an alert (**P** will provide feedback to **C** following an analysis of rule matches). For the purposes of this paper, we have selected an example rule set from SURICATA which is a widely used Open source network threat detection engine [2] (see [3] for details of the rule syntax).

Examples of Rules

The SURICATA rule format consists of an action, protocol, source IP, source port, direction, destination IP, rule options. An example of a rule is the following [3]:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (
       msg: ″ET TROJAN Likely Bot Nick in IRC (USA +..)″;
       flow: established, to_server;
       flowbits: isset, is_proto_irc;
       content: ″NICK‶;
       pcre: ″/NICK.⋆USA.⋆[0-9]{3,}/i″;
       reference: url,doc.emergingthreats.net/2008124;
       classtype: trojan-activity;
       sid: 2008124;
       rev: 2;
)
```

For details of the full syntax of Suricata rules see [3]. A simple example of a rule for intrusion-detection system would be:

```
alert tcp 1.2.3.4 [80:184,!89] -> any [80,100] (
       content: ‶Trojan″
)
```

This rule will cause an alert on TCP traffic packets with source IP of 1.2.3.4, a source port in the range 80 to 184 except for 89, any destination IP, a destination port in the range 80 to 100, and contains “Trojan” in its payload.

3.4 Prerequisites of the Protocol

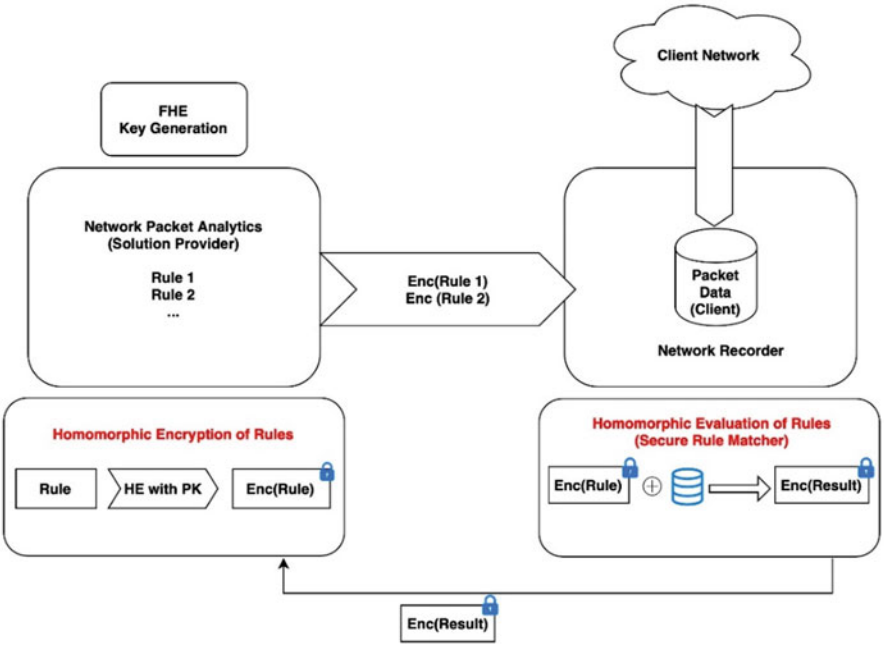
1. A client has network traffic that they want analyzed for cyber threats.
2. The provider has a proprietary set of rules that can be applied to the client’s network packet traffic to detect and monitor threat actors.

3.5 Protocol Steps

1. The provider generates a cryptographic key and corresponding public parameters.
2. The provider uses their key to create a homomorphically encrypted version of their rule set.
3. The provider packages the encrypted version of their rule set along with the public parameters into a secure rule checker. This can take the form of a hardware device that will be hosted by the client or software that will be run in the client’s environment.
4. The client agrees to host the provider’s secure rule checker and to allow the secure rule checker to access the client’s network packet traffic.

5. Upon receipt of network packets from the client, the secure rule checker will homomorphically evaluate the encrypted rule set against the unencrypted network traffic to generate an encryption of the results of the rules checked against the network traffic.
6. The client sends the encrypted results of the secure rule checker to the provider.
7. The provider uses their secret key to decrypt and analyzes the results.
8. The provider gives feedback and advice (at an agreed upon level of detail) to the client.
9. The client acts on the feedback and advice that it receives from the provider.

The following diagram demonstrates the flow:



Note that the rule matching and analysis of the results do not need to be done in real time (although the rule matching will need to be able to keep up with the network traffic volume).

Additionally, the client and threat actors can attempt to learn information about the provider's rule set from the feedback that is provided to the client. This can be mitigated by the format of the feedback and recommendations ("Install the following software updates/patches", not "block network traffic with the following rule") as well as providing periodic feedback instead of real time feedback. It will be the responsibility of the provider to balance the detail of their feedback with their requirement to keep their rule set private.

4 Performance, Usability, and Scalability

The performance of the protocol will depend on the volume of **C**'s network traffic, the number of rules in **P**'s rule set, the level of complication in the rule set, the hardware hosting the secure rule checker, and the security level on the encryption. In order to obtain a desired network packet throughput, **P** may need to limit the level of complexity in the rule set and/or the number of rules. Limiting the rule set in this way may have an effect on the ability of **P** to detect and identify threat actors.

It will be the responsibility of **P** to produce a rule set that can perform the desired task within the performance envelope. If required, increased performance may be possible by using an HE scheme, like that of Genise et al. [1], designed more specifically to the types of operations required in rule checking. Standardized FHE schemes [4] enable some efficient operations (such as the comparisons of encrypted integers) but may be less efficient than other HE schemes on other operations (such as substring matchings).

4.1 Security Agencies

This is a common issue for intelligence agencies around the world. One such example is the Canadian Communications Security Establishment which has explicitly asked for solutions to the problem [5].

4.2 Fraud Detection

Applying the same model of secure and confidential rule matching as in the above example, it could be possible to detect patterns of fraudulent behavior in banking transactions within a bank environment without disclosing those rules.

References

1. N. Genise et al., Homomorphic Encryption for Finite Automata, (2019)
2. Suricata, https://suricata-ids.orgSuricata
3. Suricata User's Guide, Available at https://readthedocs.org/projects/suricata/downloads/pdf/latest/
4. Homomorphic Encryption Standard – https://homomorphicencryption.org/standard/
5. CSE Innovation challenge: https://www.ic.gc.ca/eic/site/101.nsf/eng/00082.html

Trusted Monitoring Service (TMS)

Xiaoqian Jiang, Miran Kim, Kristin Lauter, Tim Scott, and Shayan Shams

1 Privacy-Preserving Health Monitoring[1]

In healthcare, timely monitoring of patients is a big problem, especially those who live alone in their own home or nursing home. There are many situations where people fall but no help is available, which leads to severe conditions and even mortality. Falls are the second leading cause of accidental or unintentional injury deaths worldwide, each year an estimated 646,000 individuals die from falls globally of which over 80%, are in low- and middle- income countries [1]. In the US, the age-adjusted rate of fall deaths is 62 deaths per 100,000 older adults and this rate is increasing [2]. Fall death rates among adults aged 65 and older have increased more than 30% from 2007 to 2016 [3]. Among older people in the U.S. (age 65+)

[1] https://www.youtube.com/watch?v=myfGdhZtKa4

X. Jiang · S. Shams
School of Biomedical Informatics, University of Texas Health Science Center at Houston, Houston, TX, USA
e-mail: xiaoqian.jiang@uth.tmc.edu; shayan.shams@uth.tmc.edu

M. Kim
Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea
e-mail: mirankim@unist.ac.kr

K. Lauter
Facebook AI Research, Seattle, WA, USA
e-mail: klauter@fb.com

T. Scott (✉)
Deloitte, Sydney, NSW, Australia
e-mail: timscott@deloitte.com.au

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_5

there are approximately 750,000 falls per year requiring hospitalization due to either bone fracturing (approx. 480,000 cases) or hip fracturing (approx. 270,000 cases) [4].

The disconnect between healthcare providers and the elderly population is a concerning problem. A simple camera and monitoring system do not resolve the problem, as healthcare providers cannot reliably monitor dozens of screens (e.g., fatigue, failure to recognize, etc.). Advanced artificial intelligence technology can detect or predict high risk patterns (e.g., falling or seizure), which can help to address this important challenge. This approach has already had success in healthcare applications (i.e., hip fracture prediction [5], skin cancer classification [6], lymph node metastases [7], etc.). Uploading real-time video/vital signals to a cloud service provider (e.g., Microsoft Azure) for real-time motion/gait recognition could be a promising way to close the technology gap.

However, the non-trivial risk of invasion of privacy must be addressed to ensure the viability of providing virtual care to patients [8]. Many people would find it uncomfortable to deploy surveillance cameras in bathrooms, even though the benefits are clear since a large proportion of falls happen in the bathroom. Based on a survey of senior residents' perceived need of and preferences for "smart home" sensor technologies, 10 out of 14 participants stated that they would not want to have a video sensor installed in their residence [9]. It is emotionally unacceptable for many patients to have video streams of their personal life constantly uploaded to a virtual machine at a commercial cloud service provider. The high-profile information breaches reported in the news in recent years [10] heightens public concern that inappropriate handling of their personal data by an untrusted 3rd party will put their personal privacy at risk.

To solve this problem, we need a cloud service which can securely process video streams using Artificial Intelligence (AI) solutions to detect falls, without sharing the video stream of the patient with the cloud in the clear. *Homomorphic Encryption provides a solution to this problem:* we can build a service which uploads only homomorphically encrypted data to the cloud, and the cloud can then process the AI algorithms on the encrypted video stream and provide encrypted alerts which detect falls to the Nursing station of the Trusted Monitoring Service.

This is an example of secure outsourcing of computation to monitor patient motion features, where the patient's interaction with their provider is enabled by AI models hosted by an untrusted 3rd party. We believe homomorphic encryption (HE) is the right tool to reconcile this dilemma by delivering utility while protecting privacy. Using carefully designed HE protocols, we can send the minimum necessary information for motion recognition (encrypted) to the cloud service provider, where advanced machine learning algorithms (i.e., from different developers) are deployed. When serious events are detected, an immediate alert will be sent to the care provider and helpers can be dispatched to save lives. Such a cloud service could help to reconcile the goals of privacy and utility, providing timely support to the elderly population.

2 Business Motivation

Based on information provided by the CDC, each year about $50 billion is spent on non-fatal fall injuries and $754 million is spent on fatal falls [11]. For non-fatal falls:

- $ 29 billion is paid by Medicare,
- $ 12 billion is paid by Private or Out of Pocket payers,
- $ 9 billion is paid by Medicaid.

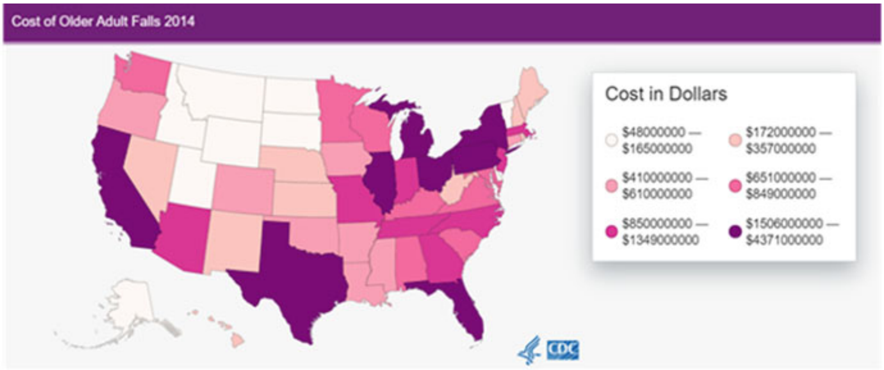
As the number of Americans, age 65 and older, grows we can expect the number of fall injuries and the cost to treat these injuries to soar. The global fall detection systems market was valued at USD 365 million in 2018 and is expected to reach a market valuation of approximately USD 544 billion by 2026 growing at a CAGR of 4.2% during the forecast period [12]. Note that these figures only account for detection; fall prediction potentially has a much higher value, currently not quantified by the market.

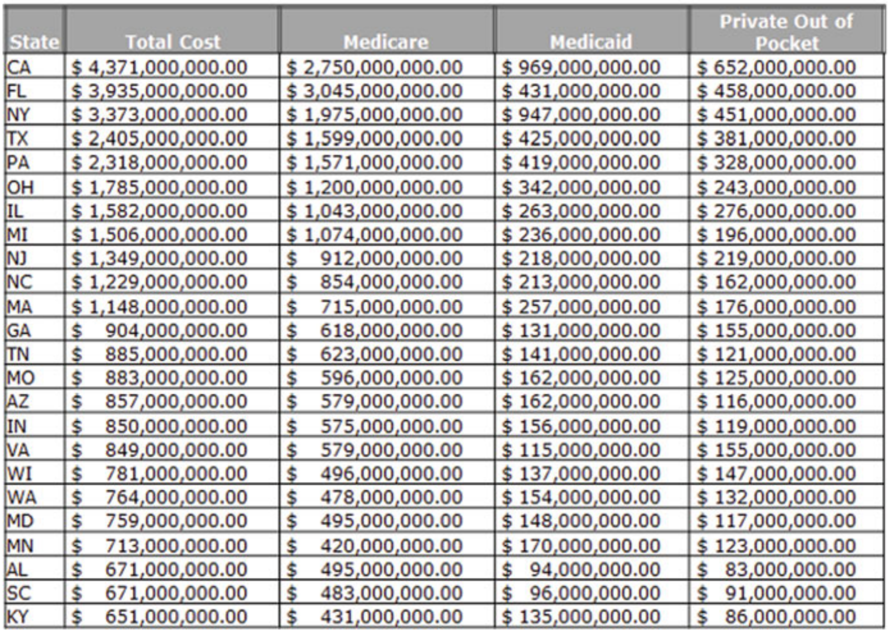
There is a significant cost associated with injuries incurred during a fall for the elderly. By 2020, the cost to the US Healthcare system is expected to be $ 54.9 billion [13]. This can come in the form of emergency services, hospital admission, medical intervention, recovery, and ongoing medical and family support. We are constantly bombarded with news stories of loved ones who have fallen and become injured, and through either inappropriate or untimely intervention, the fall has resulted in death or serious long-term injuries (Fig. 1).

Existing capabilities only provide support after the fact, such as an emergency call button [14] which is worn and pressed once the fall has occurred. If for any reason the device is not with the person, or during the incident the device cannot be operated, the service is ineffective. Wearables like the Apple iWatch with its fall detection feature have tried to address this, but once again only address the fall itself. Our proposed solution will monitor, notify of fall events, and look at predictive factors that can prevent a fall. This monitoring addresses other critical health and environmental factors. The use of AI to support predictive monitoring will allow patients with pre-existing medical conditions to structure support mechanisms with their caregivers to prevent events through the monitoring of known triggers.

For elderly care providers, having proactive monitoring in high risk/personally sensitive areas such as bathrooms offers a significant advantage in providing the appropriate care in a less labor-intensive way. Timely intervention can significantly impact the outcome from a fall, reducing the ongoing cost of care and facility reputation. It also provides the residents with a greater level of autonomy and independence which can greatly affect mindset.

Medical professionals such as geriatricians and GP's with access to a proactive monitoring capability can better provide specific medical care in the most appropriate environment (home, hospital, or care facility). Monitoring of major events (such as a fall) can help them better inform hospitals and emergency services of any





State	Total Cost	Medicare	Medicaid	Private Out of Pocket
CA	$ 4,371,000,000.00	$ 2,750,000,000.00	$ 969,000,000.00	$ 652,000,000.00
FL	$ 3,935,000,000.00	$ 3,045,000,000.00	$ 431,000,000.00	$ 458,000,000.00
NY	$ 3,373,000,000.00	$ 1,975,000,000.00	$ 947,000,000.00	$ 451,000,000.00
TX	$ 2,405,000,000.00	$ 1,599,000,000.00	$ 425,000,000.00	$ 381,000,000.00
PA	$ 2,318,000,000.00	$ 1,571,000,000.00	$ 419,000,000.00	$ 328,000,000.00
OH	$ 1,785,000,000.00	$ 1,200,000,000.00	$ 342,000,000.00	$ 243,000,000.00
IL	$ 1,582,000,000.00	$ 1,043,000,000.00	$ 263,000,000.00	$ 276,000,000.00
MI	$ 1,506,000,000.00	$ 1,074,000,000.00	$ 236,000,000.00	$ 196,000,000.00
NJ	$ 1,349,000,000.00	$ 912,000,000.00	$ 218,000,000.00	$ 219,000,000.00
NC	$ 1,229,000,000.00	$ 854,000,000.00	$ 213,000,000.00	$ 162,000,000.00
MA	$ 1,148,000,000.00	$ 715,000,000.00	$ 257,000,000.00	$ 176,000,000.00
GA	$ 904,000,000.00	$ 618,000,000.00	$ 131,000,000.00	$ 155,000,000.00
TN	$ 885,000,000.00	$ 623,000,000.00	$ 141,000,000.00	$ 121,000,000.00
MO	$ 883,000,000.00	$ 596,000,000.00	$ 162,000,000.00	$ 125,000,000.00
AZ	$ 857,000,000.00	$ 579,000,000.00	$ 162,000,000.00	$ 116,000,000.00
IN	$ 850,000,000.00	$ 575,000,000.00	$ 156,000,000.00	$ 119,000,000.00
VA	$ 849,000,000.00	$ 579,000,000.00	$ 115,000,000.00	$ 155,000,000.00
WI	$ 781,000,000.00	$ 496,000,000.00	$ 137,000,000.00	$ 147,000,000.00
WA	$ 764,000,000.00	$ 478,000,000.00	$ 154,000,000.00	$ 132,000,000.00
MD	$ 759,000,000.00	$ 495,000,000.00	$ 148,000,000.00	$ 117,000,000.00
MN	$ 713,000,000.00	$ 420,000,000.00	$ 170,000,000.00	$ 123,000,000.00
AL	$ 671,000,000.00	$ 495,000,000.00	$ 94,000,000.00	$ 83,000,000.00
SC	$ 671,000,000.00	$ 483,000,000.00	$ 96,000,000.00	$ 91,000,000.00
KY	$ 651,000,000.00	$ 431,000,000.00	$ 135,000,000.00	$ 86,000,000.00

Fig. 1 Cost of CDC older adult fall statistics [12]

preexisting medical complications that may impact the intervention provided by the front-line medical staff.

For emergency services, accurate, qualified, and timely notification of an event means the appropriate response personnel can be dispatched. This has a follow-on effect for governments that can reduce the number of presentations to emergency departments, reducing the burden on the healthcare system. Early intervention my also reduce the severity of injuries presented, leading to better outcomes.

The key stakeholders are family members that support their loved ones. The opportunity for their loved ones to maintain a high level of independence and autonomy while still providing an appropriate level of care is a critical factor. The cost to this group must be counted in both dollars saved and the emotional support.

3 Protocol (Workflow)

The following solution for analyzing an encrypted video feed is described in a recent technical paper, HEAR: Human Action Recognition via Neural Networks on Homomorphically Encrypted Data [15].

There are four parties in the protocol that interact with one another, see Fig. 2 for a high-level diagram of the architecture.

- **Parties/devices (4):**
    1. Streaming video recording device
    2. Processor (may include GPU accelerator)
    3. Cloud provider
    4. Nurse station or Trusted Monitoring Service (TMS) provider
- **Set-up:**
    - Step 1: Processor, Cloud, and TMS agree on the required functionality and the parameters for the HE-based system.
    - Step 2: All parties are provisioned with software to perform HE operations. The Cloud is provisioned with (could be encrypted or unencrypted) custom CNN prediction model to use as a classifier. Processor encodes and encrypts streaming skeletal data (frequency and frame rate to be determined). Cloud processes predictions given CNN model. Nurse decrypts alerts or classifications.
    - Step 3: Key provisioning. A public/secret key pair is generated by the patient. The public key is transmitted securely to the Processor and the secret key is transmitted securely to the Nursing station.
- **Work/data flow:**
    - Step 1. Video recording by (stationary video camera) of patient is transferred to the processor (physical cord)
    - Step 2. Processor creates skeletal video
    - Step 3. Processor encrypts skeletal video stream (frame rate to be determined) using public key
    - Step 4. Processor upload encrypted feed to the cloud
    - Step 5. Cloud service processes predictions/classifications on encrypted data (every second, 5–10 frames)
    - Step 6. Cloud sends encrypted classifications to the Nurse Station (TMS) (alert flag or ~ 20-fold classification)
    - Step 7. Nurse decrypts classification results and responds to any alerts.

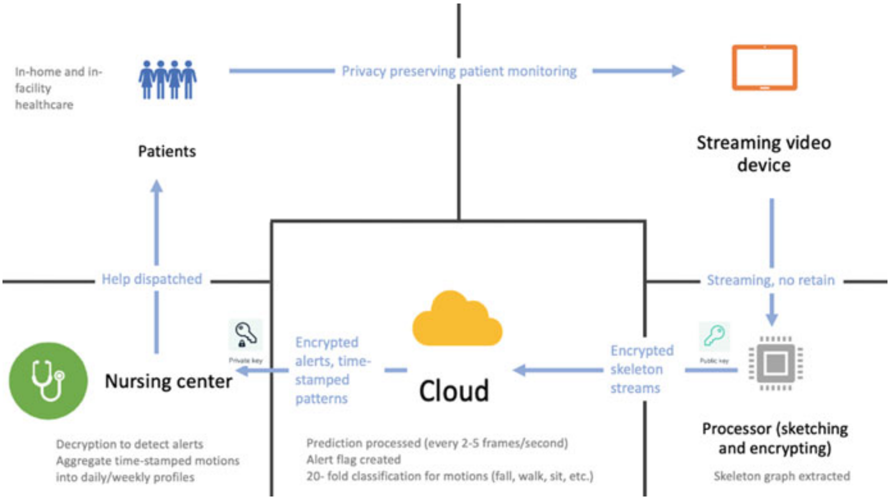

In-home and in-facility healthcare
Patients
Privacy preserving patient monitoring
Streaming video device
Help dispatched
Streaming, no retain
Private key
Nursing center
Encrypted alerts, time-stamped patterns
Cloud
Encrypted skeleton streams
Public key
Processor (sketching and encrypting)
Decryption to detect alerts
Aggregate time-stamped motions into daily/weekly profiles
Prediction processed (every 2-5 frames/second)
Alert flag created
20- fold classification for motions (fall, walk, sit, etc.)
Skeleton graph extracted

Fig. 2 Architecture for a privacy-preserving healthcare monitoring protocol

- **Assumptions (Threat model)**

1. Honest-but-curious security model: all parties follow the protocol & execute all steps correctly.
2. Secure authenticated channels

   – Physical cords connect the video recorder (1) to the processor (2)
   – TCP-IP/TLS channels between processor (2) and cloud (3) and between cloud (3) and nurse station (4)
   – Secure authenticated channels are required to ensure safety and integrity: to prevent an attacker from faking the video stream or impersonating the Trusted provider or the semi-honest cloud.

3. Non-collusion assumption: cloud (3) and nurse station (4) do not collude. The cloud stores the encrypted skeleton stream but does not have the decryption key. The nurse station has the decryption scheme but only receives encrypted alerts or classifications. If they share data, then both parties can access the decrypted skeletal video stream. This skeletal stream may uniquely identify the individual if used in other databases of skeletal video streams.
4. Streaming device & processor do not retain video stream: enforced deletion after encryption.

5. Our solution does not prevent:
    - Denial of Service attacks (DOS),
    - Network availability failure or physical disconnection of the video recording device,
    - Negligence in the TMS (nurse is absent or does not respond to alert).

4 Performance, Usability, Scalability

A privacy preserving healthcare monitoring system is very time sensitive. For example, most seizures stop spontaneously within 2 min and patients usually receive treatments 30 min after seizures begin [16]. The system must ensure both accuracy and efficiency to be useful. The state-of-the-art Radar system [17] can detect a fall per 0.25 s at 100% accuracy but the cost is prohibitive. We aim to achieve similar performance while supporting privacy-preserving monitoring (which the Radar system does not). But there is a tradeoff between these metrics. Deep learning models usually perform better by using a larger time window of video frames, but this introduces more complexity to model evaluation on encrypted data, delaying critical event detection. Outsourcing computation on encrypted data to the cloud would alleviate the problem by using virtual machines of different capacities to support different needs (e.g., lifestyle monitoring, fall detection, etc.). This may allow to strike the right balance between cost and usability and provide virtual care to a large population. Because the sketching phase costs are constant, performance is essentially determined by the cloud service provider, which makes the system highly scalable. It is important to develop machine learning models with a good understanding of the fundamentals of HE [18]. There are several choices for HE schemes which have different advantages in terms of operations (i.e., logical or arithmetic) but none of them are well-suited to the evaluation of deep circuits (HE is not efficient for high degree polynomials). Designing customized low-depth learning models is thus an area ripe for innovation.

5 Applications of Trusted Monitoring Systems

We have described a Trusted Monitoring System for fall detection for privacy-preserving healthcare monitoring. Our framework design can be used in many other scenarios, as the need for privacy-preserving monitoring is quite ubiquitous. Here, we describe some additional examples where our framework would apply.

Examples	Features
In-home and nursing home fall detection	Monitoring of sensitive areas within the home/nursing home to detect when a fall occurs and trigger a notification to nursing or emergency services. Notifications can also be sent to families and care providers.
Hospital wards	Monitoring of patients in a hospital ward for falls or adverse events with a timely notification issued to the closest nursing station.
Psych-wards	Use of AI predictive models for the detection of potential fights or adverse events with a timely notification issued to the closest nursing station. With the predictive capability nursing intervention can be converted from reactive to proactive.
Child protective services	In-home monitoring of at-risk children and families who are currently under investigation and require constant monitoring.
Schools	Monitoring of high-risk areas for the detection of bullying, fighting, or other antisocial activities which then notify school authorities.
Telemedicine	Tracking and monitoring of patients in rural or remote areas where health services are not available. Predictive modeling could support front-line medical staff that might not typically be exposed to specific cases.
24-hour, unsupervised gyms	Monitoring of people at gyms during unsupervised times for the detection of falls and the notification to emergency services.
Medically at-risk individuals living or recovering at home	Monitoring individuals who suffer from lifelong medical conditions affecting their movement or stability. Providing better quality of life and greater independence.

Possible extensions of the service model

Extension	Descriptions
Marketplace for predictive models	A marketplace could be developed for uploading specific predictive models based on known triggers for conditions such as epilepsy.
Richer data collection	Motion and vitals could be collected via additional devices such as wearables and video technology to provide a holistic image of the patient’s medical condition. Data to be provided to medical care providers (doctors) to track a patient’s health over time.
Data marketplace	Collected data could be accessed for research and AI/ML model training.

References

1. Falls. https://www.who.int/news-room/fact-sheets/detail/falls (accessed 6 Feb 2020).
2. Deaths from Falls | Home and Recreational Safety | CDC Injury Center. 2019. https://www.cdc.gov/homeandrecreationalsafety/falls/fallcost/deaths-from-falls.html (accessed 6 Feb 2020).
3. E. Burns, R. Kakara, Deaths from Falls Among Persons Aged ≥ 65 Years-United States, 2007-2016. MMWR Morb Mortal Wkly Rep 2018; 67:509–14.

4. TJ Petelenz, SC Peterson, SC Jacobsen, Elderly fall monitoring method and device, US Patent, 2002. https://patentimages.storage.googleapis.com/c5/7a/c0/8431d535a77e29/US6433690.pdf (accessed 7 Feb 2020).
5. MA Badgeley, JR Zech, L Oakden-Rayner, et al. Deep learning predicts hip fracture using confounding patient and healthcare variables. NPJ Digit Med 2019; 2:31.
6. A. Esteva, Skin cancer classification with deep learning. https://cs.stanford.edu/people/esteva/nature/ (accessed 6 Feb 2020).
7. Ehteshami Bejnordi B, Veta M, Johannes van Diest P, et al. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. JAMA 2017; 318:2199–210.
8. SR Steinhubl, K-I Kim, T Ajayi, et al. Virtual care for improved global health. Lancet. 2018; 391:419.
9. G Demiris, BK Hensel, M Skubic, et al. Senior residents' perceived need of and preferences for 'smart home' sensor technologies. 2008; 24:120–4.
10. C Bradford, 7 Most Infamous Cloud Security Breaches – StorageCraft. StorageCraft Technology Corporation 2017. https://blog.storagecraft.com/7-infamous-cloud-security-breaches/ (accessed 7 Feb 2020).
11. Falls Data | Home and Recreational Safety | CDC Injury Center. 2019. https://www.cdc.gov/HomeandRecreationalSafety/Falls/fallcost.html (accessed 7 Feb 2020).
12. Fall Detection Systems Market Analysis – Global Industry Size, Share, Growth Opportunity, Trends and Forecast 2026. MarketWatch. https://www.marketwatch.com/press-release/fall-detection-systems-market-analysis-global-industry-size-share-growth-opportunity-trends-and-forecast-2026-2019-04-05 (accessed 7 Feb 2020).
13. A Lee, K-W Lee, P Khang, Preventing falls in the geriatric population. Perm J 2013; 17:37–9.
14. C Roberts, How to choose a medical alert system. https://www.consumerreports.org/medical-alert-systems/how-to-choose-a-medical-alert-system/ Published Online First: 2018.
15. M Kim, X Jiang, K Lauter, S Shams, HEAR: Human Action Recognition via Neural Networks on Homomorphically Encrypted Data, in submission, 2020.
16. DH Lowenstein, T Bleck, RL Macdonald, It's time to revise the definition of status epilepticus. Epilepsia 1999; 40:120–2.
17. BY Su, KC Ho, M Rantz, et al. Radar placement for fall detection: Signature and performance. AIS 2018; 10:21–34.
18. Homomorphic Encryption Standard, https://homomorphicencryption.org/standard/HomomorphicEncryption.org, 2017, in this volume, Part 2.

Private Set Intersection and Compute

Flavio Bergamaschi, Tancrède Lepoint, Peter Leihn, and Sreekanth Kannepalli

1 Motivation

We consider the scenario where two or more data owners would like to join their data and compute some functions over the intersection of their data in a privacy-preserving way and without disclosing their dataset to each other nor the intersection of their datasets. This scenario is primarily motivated by the following aspects:

- Enabling compliance with emerging privacy regulation in multiple jurisdictions. Privacy regulations such as Europe's General Data Protection Regulations (GDPR) and California's Consumer Privacy Act (CCPA) significantly restrict an organization's ability to share consumer data without the explicit permission of the individual consumer.
- Unlocking valuable insights from external datasets which would otherwise be inaccessible due to the sensitivity or commercial value of that data.

1.1 Privacy Compliance

Privacy compliance risks are real. Gartner estimates that by 2022, the personal information of half of the planet's population will be covered by local privacy regulations in line with GDPR, up from one-tenth in 2019 (Gartner, 2019) [1]. The

F. Bergamaschi · T. Lepoint · P. Leihn
SRI International, Menlo Park, CA, USA
e-mail: crypto@tancre.de; peter.leihn@ixup.com

S. Kannepalli (✉)
Microsoft, Redmond, WA, USA
e-mail: vkanne@microsoft.com

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_6

International Association of Privacy Professionals and EY (2017) estimated that Fortune's Global 500 companies will spend roughly $7.8 billion to ensure GDPR compliance [1]. Similarly, the State of California has estimated CCPA compliance could cost companies a total of $55 billion (Attorney General's Office California Department of Justice, 2019) [2].

Fully homomorphic encryption (FHE) is one of the most promising technologies that allows each party to provide their data in encrypted form, so that the data can be analyzed while it remains encrypted throughout the computation. That is, data is not actually shared among the parties, instead it is only made available for analytics in encrypted form. So, it can be processed without disclosing the underlying personally identifiable information (PII) and other sensitive fields, which is the primary concern for privacy regulators. Importantly, data custodians remain in control of how their consumers data is used.

1.2 Co-marketing as a Use Case

Today the consequences of privacy breaches are more serious than ever, not only in terms of penalties, but impact on reputation and customer loyalty. Marketers are now concerned with ensuring the security and privacy of their customer's data but continue to desire to gain valuable new insights to tailor new offers and drive revenue growth.

For example, an airline, a hotel chain, and a rental car company may wish to identify customers that they have in common with an average consolidated spend on all their products above a certain threshold, for the purpose of preparing a joint promotion to those customers. Similarly, a train company and point of sale providers may run a Private Set Intersection and Compute protocol to identify how many train riders went to these points of sale providers, and how much money they spent in total,[1] to make business decisions on the joint data.

The privacy challenge is that no party wants to (nor can) share their customer data with any of the others. The following is an approach to achieving this using proven private set intersection and homomorphic computation techniques.

2 Application Functionality

2.1 Database Statistics on PSI Selected Entries

We consider a setting where two or more parties (data owners) possess databases of identifiers and associated data. Together they would like to compute functions

[1] https://www.youtube.com/watch?v=mPMLY6UzvsI

on the associated data of the identifiers that they have in common to draw useful information from the aggregates, without sharing the data. A main security goal is that the data remains fully encrypted once it leaves the client's nodes. These parties will use client nodes to encrypt and/or prepare their data and will compute the intersection and compute over the joint data either among themselves (in a privacy-preserving way) or with the help of a third party (for which the data remains unreadable throughout the process).

We consider the following parties:

- Data owners that hold locally a database of keys and associated values.
- Client nodes that are responsible for preparing and encrypting the database prior to uploading the encrypted database to each other or to a third-party server and optionally perform the intersection and computation over the joint data.
- Central server that performs the intersection and computation over the joint data.

We consider the following security/threat models:

- honest clients that do not corrupt or falsify input data;
- semi-honest client compute nodes, and semi-honest central compute node.

3 Protocol

In the Private Set Intersection and Compute framework, four main steps will be performed on the encrypted data.

1. The preparation step enables the parties to agree on the computation they would like to perform, to preprocess their data, and to exchange key material information that will be needed in the following steps.
2. The intersection step enables the parties to privately find the intersection of their data keys, without learning which keys they have in common.
3. The compute step enables the parties to privately compute the agreed-upon function on the keys and values associated with the keys in the intersection, without learning anything about the intermediate values in the computation.
4. The reveal step enables the parties to reveal to one or more parties of their choosing the result of the computation.

Each of these steps may or may not include a third party. In the rest of this section, we will describe the input/output and functionality of each of these steps, and in the rest of the section we will describe in detail the first of three instantiations of the protocol:

1. A protocol with a central compute node for $N \geq 2$ parties, in which the central node will learn the size of the intersection and will perform both the intersection and computation.
2. A protocol without a central compute node for $N = 2$ parties, in which one party will learn the size of the intersection and perform the computation step.

3. A protocol with a central compute node for $N \geq 2$ parties, in which the central node will perform both the intersection and computation and learn no information whatsoever about the intersection. This protocol will perform all operations with homomorphic encryption. This scheme can adapt to either scenarios the space of keys is bounded (e.g., an integer between 1 and 20,000) or not.

3.1 Workflow

This section describes the general workflow of the protocol. We consider N users $U_1, U_2, \ldots, U_N$ and potentially a central node C. For all i, user U_i possesses a database with m records $(k_{i,1}, v_{i,1}), (k_{i,2}, v_{i,2}), \ldots, (k_{i,m}, v_{i,m})$.

- During this preparation step, all users agree (with or without the help of a central compute node) and over regular authenticated and secure communication channels on the following elements:
  - the input/output of the protocol – what is the key space and who should recover the output,
  - the precise computation to be performed on the joint data (such as sums, differentially private aggregations, linear regression, filtering, etc.),
  - who participates in the computation and in particular whether an untrusted central server can help with the computation.
- During the intersection step, all users will jointly compute a database that corresponds to the temporary intersection of all the databases:

  $$\left(Enc\left(sk, k_j\right), HE\left(sk', v_{1,j}\right), \ldots, HE\left(sk', v_{N,j}\right)\right)$$

  so that, for every user U_i, the element $(k_j, v_{i,j})$ is present in her database. In this step, we consider two types of encryption schemes:
  - $Enc(sk, \ldots)$ is a deterministic encryption scheme and enables the party performing the intersection to intersect over the values of the encryptions of keys.
  - $HE(sk', \ldots)$ is an homomorphic encryption scheme that supports the function agreed-upon during the preparation step to be computed.

  During the compute step, one party will compute homomorphically the function agreed-upon. More precisely, from the temporary database, the party will compute the agreed-upon function on all the $(HE(sk', v_{1,j}), \ldots, HE(sk', v_{N,j}))$ values using the homomorphic encryption scheme.
- During the reveal step, the party who computed the result using homomorphic encryption sends the result to the party (or parties) that is supposed to receive it and know the decryption key of the homomorphic encryption scheme.

3.2 First Protocol: N Parties with One Central Compute Node

This protocol considers N > = 2 data owners that collaborate with a central compute node to perform the private set intersection and computation. This protocol is particularly adapted to the co-marketing use case where the central compute node is a cloud provider independent of the clients.

- Setup: key value databases $DB := (DB_{key}, DB_{value})$
- Steps:
  - Preparation: Data owners use their client nodes to generate and exchange encryption keys: symmetric keys (e.g., AES) for encrypting DB_{key} ‘s and HE keys for encrypting DB_{value} ‘s.
  - Intersection: After encrypting, clients upload $Enc(DB) := (Enc(DB_{key}), HE(DB_{value}))$ to central node. Central node performs set intersection across $Enc(DB_{key})$ ‘s, producing the intersection of $HE(DB_{value})$ ‘s. The count of intersections and their indexes in DB are leaked to central node.
  - Computing: Central node performs data aggregation/computation across each intersection and get encrypted results the results $HE(result)$.
  - Revealing: Central node return results to each entitled client node. Client node decrypts $HE(result)$ for consumption.

4 Examples

The protocols described in this document gave rise to many prototypes by academia and industry, and even deployments in products (IXUP, Google, etc.). We mention below two examples that correspond to the first two protocols described.

4.1 IXUP

IXUP[2] is a commercially available platform for parties to undertake privacy preserving analytics. The platform never sees or stores encrypted data and provides a modelling environment for data custodians to agree on exactly what insights will be extracted in a data collaboration. While the platform knows the number of matches, the platform’s governance prohibits this being disclosed to any party without the permission of all data contributors. The platform sees no values used in the homomorphic computation which is conducted using Microsoft’s SEAL HE library.

[2] https://ixup.com/Platform/

More precisely, IXUP uses the first protocol described in the previous section, which includes a central compute node and $N \geq 2$ parties.

- During the preparation phase, the data custodians assemble and use the platform to decide on the intersection parameters and function to be computed. Once all parties agree, a shared key for a deterministic encryption scheme is generated and shared by all the data custodians but remains unknown to the compute node.
- During the intersection phase, all data custodians encrypt their data: they encrypt the keys on which the intersection is made using the shared key, and encrypt the values associated with this key with the homomorphic encryption scheme key. They then upload the encrypted data to the central compute node. The central compute node then performs the intersection over the deterministic encryption of the keys. In that setting, the central node learns the size of the intersection, but not the values of the keys or associated data.
- During the compute phase, the central compute node uses the homomorphic encryption scheme to compute homomorphically over the data.
- During the reveal phase, the central compute node will send the result of the computation to the party (parties) that are supposed to receive the result and need to know the decryption key of the homomorphic encryption scheme.

4.2 Private Join and Compute

In a blog post [3] published in June 2019, Google described a solution called Private Join and Compute in which $N = 2$ parties can encrypt their identifiers and associated data, join them, and perform computations on the overlapping set of data. Contrary to the IXUP solution, there is no need for a central server in the Private Join and Compute solution. Google's solution uses a deterministic double encryption scheme. In that scheme, a value encrypted with key k_1 and then with key k_2 is equal to a value encrypted with key k_2 and then with key k_1.

More precisely, Private Join and Compute can expand to the first protocol described in the previous section, which includes a central compute node and $N \geq 2$ parties. In Private Join and Compute, only the second party has data associated with each key.

- During the preparation phase, the data custodians agree on the computation to be performed.
- During the intersection phase, the first party samples a secret key k_1 and encrypts its own data identifiers with that key. It then sends its encrypted data to the second party. The second party samples a secret key k_2, a homomorphic encryption key K, encrypts its own data identifiers with k_2 and the associated values with K, and then doubly encrypt with k_2 and shuffle the encrypted data received from the first party. Then it sends all the encrypted values from previously to the first party. The first party can doubly encrypt with k_1 the encrypted identifiers received from the other party and compare the doubly encrypted values to identify the intersection,

without knowing the values of the identifiers in this intersection. Note that in that setting, the first party will learn the number of elements in the intersection.

- During the computation phase, the first party uses the homomorphic encryption scheme to compute homomorphically over the encrypted associated data.
- During the reveal phase, the value computed by the first party is sent to the second party for decryption with the key K.

5 Performance, Usability, and Scalability

A paper by Google [4] reported on the deployment of a Private Set Intersection and Sum application, under the Private Join and Compute framework. A detailed comparison of Private Intersection-Sum protocol and variants is demonstrated in Table 4 of the paper.

We report compute timings (in seconds) by IXUP for computations that are similar – from a comparative standpoint not the same. The protocol timings listed have a different set of parameters and behavior, including (but not limited to):

- IXUP is loading all the data into a database and moving it in and out to the serverless functions, whereas the paper appears to be using in memory datasets as they are quite small giving a significant performance boost.
- The protocols tested in the paper use a combined PSI and homomorphic sum assuming distributed data sets. IXUP uses ASE and Hashing for PSI and then sums the results using SEAL in a non-distributed manner.
- The Azure VM IXUP used for encryption was similar in spec to the machine quoted in the paper.
- The IXUP approach involves an encryption step on the client side as it never allows unencrypted data on the platform. Timings for this step are also included.

Taking the IXUP testing results for total compute, excluding the encryption step and file transfer time, the approach taken by IXUP using the first protocol is returning significantly faster results as the size of the calculation increases.

The IXUP testing results were measured in the following settings:

- All files are encrypted locally on the client's computer.
- After encryption, the files are uploaded into the IXUP environment for processing.
- The private intersection and homomorphic actions are done as separate processes, with seal homomorphic encryption only used in the homomorphic computation phase.
- All times have been recorded in seconds.
- The 100,000 (Groups) has calculated 263 homomorphic sums. The file was grouped by country, and a sum for each of 263 groups was calculated.

	Encryption	Private Pintersection	Seal homomorphic sum	otal compute (Excl encrypt Tand file import)	Total encrypt and compute (excl file import)
1000	11.99	3.143	3.563	6.706	18.696
2000	12.58	3.35	3.064	6.414	18.994
3000	15.5	3.853	3.317	7.17	22.67
4000	11.94	4.03	3.327	7.357	19.297
5000	18.18	3.846	4.283	8.129	26.309
10,000	13.27	5.087	3.48	8.567	21.837
20,000	14.28	6.143	4.347	10.49	24.77
30,000	13.8	6.32	5.227	11.547	25.347
40,000	12.95	6.123	5.117	11.24	24.19
50,000	8.77	1.786	6.29	8.076	16.846
100,000	17.71	6.374	14.286	20.66	38.37
100,000 (Groups)	15.76	11.047	43.323	54.37	70.13
100000 (Fuzzy)	50.8	140.474	16.78	157.254	208.054

References

1. Global 500 companies to spend $7.8B on GDPR compliance. https://iapp.org/news/a/survey-fortune-500-companies-to-spend-7-8b-on-gdpr-compliance/.
2. Roland-Holst, David, Samuel Evans, Drew Behnke, Samuel Neal, Liam Frölund, and Yao Xiao. Standardized Regulatory Impact Assessment: California Consumer Privacy Act of 2018 Regulations. http://www.dof.ca.gov/Forecasting/Economics/Major_Regulations/Major_Regulations_Table/documents/CCPA_Regulations-SRIA-DOF.pdf.
3. Walker, Amanda, Sarvar Patel, and Moti Yung. Helping organizations do more without collecting more data. https://security.googleblog.com/2019/06/helping-organizations-do-more-without-collecting-more-data.html.
4. Ion, Mihaela, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. "On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality." In 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 370–389. IEEE, 2020.

Part IV
Applications of Homomorphic Encryption

Private Outsourced Translation for Medical Data

Travis Morrison, Sarah Scheffler, Bijeeta Pal, and Alexander Viand

1 Introduction

Overcoming language barriers remains a key challenge for aid organisations working globally. In the case of medical aid, effective communication is required not just for efficient logistics and administration, but is of vital importance to the main mission. Doctors volunteering abroad need to understand patients' records in order to make correct diagnoses and select appropriate treatments. After many in-clinic treatments, patients must follow specific drug or care regimens. These instructions must be communicated effectively to ensure a positive outcome.

Overcoming these language barriers can place considerable strain on the resources of these organisations. While bilingual doctors can facilitate the translation of medical records, prescriptions, and instructions with high accuracy, not all doctors volunteering abroad are proficient in their patients' languages. Meanwhile, professional translators are a costly resource and volunteer clinics are often under-staffed and under-funded to begin with. Therefore, efficient and

T. Morrison
Mathematics, Virginia Tech University, Blacksburg, VA, USA
e-mail: tmo@vt.edu

S. Scheffler
Department of Computer Science, Boston University, Boston, MA, USA
e-mail: sscheff@bu.edu

B. Pal
Department of Computer Science, Cornell University, Ithaca, NY, USA
e-mail: bp397@cornell.edu

A. Viand (✉)
Department of Computer Science, ETH Zurich, Zurich, Switzerland
e-mail: alexander.viand@inf.ethz.ch

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_7

effective automated solutions for medical translation are needed to lighten the translation workload.

Automated translation has become increasingly accurate due to the development of neural-network-based machine learning approaches. Cloud-based services like Google Translate offer fast and accurate translations of documents and speech. However, uploading medical data to the cloud can violate patient privacy. Meanwhile, offline translation systems are often less accurate than state-of-the-art cloud based solutions. In the case of medical translation services, there could also be concerns about releasing models trained on private medical data.

Homomorphic encryption (HE) could allow users to outsource translation to a server without revealing the underlying information. A service provider would train and maintains a machine-learning model, likely derived from private medical data, and make the translation service based on it available to the clients. The client then sends an encrypted query to the server, which homomorphically evaluates the model on the query and returns the encrypted result.

A Machine-Learning-as-a-Service (MLaaS) provider might donate the required server resources to charities, or public institutions like hospitals or universities could run such a platform, which would be many orders of magnitude cheaper than human translators. We imagine that in this setting, client devices belong to the doctors, pharmacist, or clinics rather than individual patients. Since patients already trust their doctors with their private data, this offers significant deployment benefits with little practical impact on the privacy guarantees.

While much prior work considers homomorphically encrypted machine learning (e.g., [8, 11]), natural language processing tasks like translation generally make use of Recurrent Neural Networks (RNNs) which offer unique challenges. Due to their recurrent nature, they inherently have a high depth of computation. Convolutional Neural Networks (CNNs), in contrast, are generally 'wide' rather than deep. The high-cost of evaluating deep circuits in levelled FHE is usually compensated by fully utilizing *batching* to amortize costs across many parallel computations. However, the 'narrow-but-deep' layout of an RNN computation makes it challenging to fully exploit this common technique. In addition, RNNs rely heavily on complex non-linear activation functions. While all neural network architectures tend to feature non-linear-activation functions, it has been shown [8] that low-degree polynomial functions can achieve the desired effects in lower-depth CNNs. RNNs, however, require activation functions that 'clamp' values to a fixed interval in order to avoid numerical issues. Such functions are inherently nearly impossible to emulate with low-degree polynomials.

Therefore, implementing RNN-based machine translation using homomorphic encryption is a highly non-trivial task. In order to set a feasible goal, we consider prescriptions, specifically the directions on when and how to take the drug, as a first step. These texts are suitable for an initial design since they are generally a single short phrase (e.g., "one capsule every eight hours"), frequently contain vocabulary that is otherwise rarely used (e.g., "Apply one inhalation…") yet feature a small overall vocabulary. In addition, relaying the information in these texts is both vitally important and surprisingly challenging. A large number of studies have shown how complex language or unclear instructions lead to patient misunderstandings [15].

The need for clear effective translations of prescription instructions is also evidenced by the development of standardised translation tables [1] that provide a mapping from common phrases (e.g., “Take one pill at bedtime”) into several languages. While these represent an important first step, they cover only a very limited number of prescription instructions, cannot accommodate additional explanations and only work uni-directionally.

We therefore propose a solution for private outsourced translation for medical texts, specifically prescription instructions, using homomorphic encryption. We focus on the concise texts found in prescription instructions, allowing us to showcase a feasible proof-of-concept implementation using existing tools. We evaluate our prototype and explore how the remaining challenges might be overcome. Finally we propose avenues for future work in this area.

2 Machine Translation

Modern approaches to machine translation are frequently based on neural networks, specifically Recurrent Neural Networks (RNNs). In contrast to feed-forward networks, RNNs can process sequences of inputs, incorporating information from previous parts of the input in the decision process. This makes them especially suited to natural language processing tasks.

For text-based tasks like translation, words are represented by their index in a fixed-size dictionary, i.e. a list of the k most relevant words for the task, with typical sizes for k starting around 5000 words. Chunks of the one-hot encoded phrase are then converted into *embeddings* in, e.g., $\mathbb{R}^w$ using a simple, non-task-specific, model. These chunks, rather than the individual words, make up the input sequence for the RNN.

Generating a translation with an RNN consists of two stages. First the input sequence of chunks is run through an encoding phase, which generates a fixed-length *hidden representation* of the sequence. This allows easy training of the model with sentences of varying lengths. Second, this representation is used as the input for the decoding phase, which generates the translation output.

The encoding and decoding phase each consist of a single unit, that could be either a simple “fully connected” layer or a more complex architecture such as GRUs [7] or LSTMs [10]. Each unit features at least one source of *non-linearity*. This is most commonly the `tanh` activation function which has bounded outputs, preventing numerical issues that arise due to the recurrent nature of RNNs. In each phase, the current input from the sequence and the output from the previous step are fed into the unit, as shown in Fig. 1. This allows the model to incorporate information from previous parts of the sequence, but also leads to a very deep computation graph, which makes a homomorphic encryption implementation challenging.

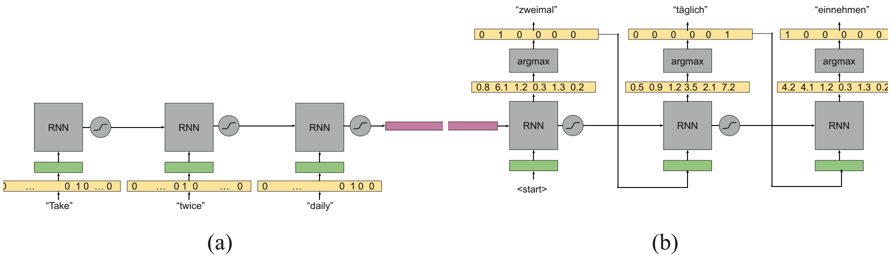


Fig. 1 Encoding and decoding stages of the model. (**a**) Encoding. (**b**) Decoding

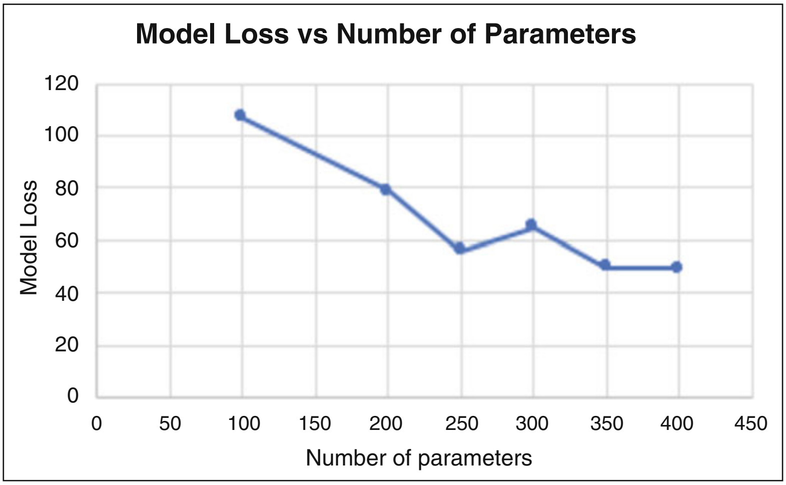


Fig. 2 ML model parameters to determine acceptable value of w

While plaintext-based solutions often err on the side of larger hidden representations, we experimented with a simple RNN model in the clear to determine what an acceptable value of w would be; the results are shown in Fig. 2.

3 Design

We assume that the client, i.e. the doctor or clinic, posses a symmetric secret key for the CKKS scheme [4]. The server, meanwhile, has access to the corresponding public relinearization and rotation keys required to evaluate the computation. For convenience of deployment, these might be set-up before a doctor departs his home country or at a major city in the destination country, where high-speed internet is available.

Given a phrase to translate, the client software first tokenizes the phrase, representing each word as a one-hot encoding vector referring to a fixed-length dictionary. Should a word not be present in the dictionary, it is usually assigned a special “unknown” token when evaluating translation models. However, in a practical deployment setup it might be more suitable to notify the user and give them a chance to provide an alternative word. In our design, we choose a dictionary size of 5000 words, which is on the smaller side for general language tasks but still common. Since the language of prescription instructions is fairly standardized, we would expect unknown words to occur with very small frequency.

Chunks of the phrase are then embedded into the hidden space, in our case $\mathbb{R}^{256}$ as this provides a good balance between accuracy and model size (See Fig. 2). This embedding is performed on-device, using a simple pre-trained lookup table. The list of embedded chunks, $x_0, x_1, \dots, x_n \in \mathbb{R}^{256}$ is the input to the FHE computation. The client encrypts the chunks using the symmetric secret key, batching them into a single ciphertext which is then sent to the server.

The server then evaluates the RNN on the input. For simplicity, we consider a fully recurrent neural network, i.e. for each element x_i in the input sequence, we have

$$h_i = g(W_x x_i + W_h h_{i-1} + b)$$

where $W_x, W_h \in \mathbb{R}^{256\times256}$ are weight matrices, b is a bias vector, and $g : \mathbb{R}^{256} \to \mathbb{R}^{256}$ is a non-linear activation function. In the decoding phase, the weight matrices are of size $\mathbb{R}^{512\times256}$ and the resulting vector is split into h_t and y_t. To derive the actual output from $y_t \in \mathbb{R}^{256}$, we find the dictionary entry that has the embedding vector that is closest to y_t (`argmax`). This model architecture can be seen in Fig. 1.

In an ideal setting, the server would evaluate the whole model under FHE and return the encrypted result to the client, once again batched into a single ciphertext for communication efficiency. However, there are several challenges that make a straightforward evaluation of the network infeasible.

3.1 Challenges

One challenge of a homomorphic encryption implementation of an RNN is the sequential nature of the architecture. A long input sentence will result in a high multiplicative depth of the computation, requiring larger parameters and therefore slower computations. Another challenge are the required non-polynomial functions. These include the non-linear activation functions, which are also present in traditional feed-forward networks. However, in RNNs they are considerably more important since the deep nature of the computation can quickly lead to numerical issues if unsuitable activation functions are chosen. More importantly, however, the decoder requires the repeated evaluation of the `argmax` function. While many neural networks use functions to like `max`, `softmax` or even `argmax`, these are generally applied at the very end of the computation and can therefore be left to the client to perform after decryption. In the RNN decoder architecture, however, the output of the argmax needs to be fed into the next stage of the computation.

We explored a variety of possible solutions to these issues during the course of this project:

1. *Polynomial Approximation:* In previous work on neural networks in FHE, non-linear activation functions like RELu have been approximated with varying low-degree polynomials [8]. In more general purpose computations, using Chebyshev polynomials to approximate continuous functions on an interval is a standard techniques in HE. However, due to the deep nature of RNNs, using standard approximations can lead to significant accuracy issues.

   More importantly, existing techniques do not admit a straightforward method for approximating `argmax` which is a multivariate function. Using the method of [5] for computing a maximum, it might be possible to compute the `argmax` by performing a linear number of `max` operations over the output vector. While this `max` operation is computationally costly, all the costs would be incurred at the server, not the computationally limited client device.
2. *Binary Representation:* Changing to a binary representation would allow us to compute the non-linear functions directly. Here, it would be best to use a scheme and implementation optimized for this setting (e.g. TFHE [6]). The activation functions can be implemented via lookup tables at the desired accuracy, while the `argmax` could be implemented directly. However, in this setting matrix-vector multiplications become prohibitively expensive. In fact, the only existing FHE implementation of a RNN that we are aware of [14] actually uses the binary setting, but uses an extremely quantized network with 4 bit weights to avoid this multiplication overhead. At such high quantization levels, more complex machine learning tasks like machine translation are likely to lose too much accuracy to be of practical use. In essence, we would only be trading the challenge of polynomial approximation with the challenge of extreme quantization.
3. *Scheme Switching:* Some other frameworks have proposed switching between different representations or schemes in order to improve performance. Glyph [13] and Chimera [2] both switch to Fully Homomorphic Encryption on the Torus (TFHE) [6] for non-linear computations such as `softmax`. While Chimera seems to have been implemented for the i-DASH 2019 competition, the implementation is not yet publicly available.

   On a conceptual level, it seems straight-forward to switch from CKKS to TFHE to e.g. evaluate a non-linearity activation function after using CKKS for matrix-vector multiplications. However, while Chimera also introduces transitions from TFHE to CKKS, these transitions produce very specific CKKS ciphertexts that contain not the original message but an exponentiated form. It is not obvious to us at this point whether or not it would be possible to "complete the circle" and convert a TFHE ciphertext back into a CKKS ciphertext that would be suitable for continuing the evaluation of the network. Even if it is not possible to switch back-and-forth in a straight-forward way, it might be possible to rephrase the entire computation from scratch in a way that can take advantage of the power of both schemes.

4. *Client Interaction:* Finally, the most simple solution is to send each individual unit's output back to the client, who will decrypt it, compute the required non-linear functions including activation functions and `argmax`, and send it back to the server. While this option introduces several additional rounds of communication and significantly increases the communication overhead, it is simple to implement and makes each individual unit a very low-depth and efficient circuit.

For the encoding phase, we use low-degree polynomial approximations. For the input sizes common for prescription instructions, the circuit depth—while high—is not entirely prohibitive. While this requires more complex training procedures, it has been shown that e.g. replacing `tanh` with `ReLU` in RNNs leads to only slightly lower performance [12]. Hopefully, with significant adjustments to the training, polynomial activation functions could be shown to have acceptable performance, too. For the decoding phase, where `argmax` is required, we consider client interaction to be most feasible for an initial design. However, the authors want to continue exploring the feasibility of scheme switching for a fully-outsourced RNN implementation.

4 Implementation and Evaluation

We developed a proof-of-concept implementation[1] using the Microsoft SEAL library [16], showing the feasibility of evaluating a sequence of RNN units. The core of the computation is made up of matrix-vector products between the plaintext weight matrices of the model and the (encrypted) input or hidden-representation vectors. We approximate the non-linear activation functions during the encoding phase with $g(x) = x^2$.

4.1 Encoding

Choosing the right *batching* layout is essential for an efficient implementation. CKKS ciphertexts of lattice dimension n can hold up to $n/2$ independent values in (virtua) *slots*. Arithmetic operations apply component-wise, i.e. in a SIMD fashion. Special automorphisms can be used to *rotate* the elements between slots cyclically.

[1] Available at https://github.com/PrivateAI-Group1/SEAL.

In order to optimize the matrix-vector products, we use the "diagonal method" [9], where we encode the matrices not row- or column-wise but instead encode the diagonals. This allows us to compute the matrix-vector-product between a matrix of dimension $k \times k$ and a vector of length k with only $k - 1$ rotations, k component-wise multiplications and $k - 1$ component wise additions.

However, since rotations on the slots are cyclical, naively encoding the values produces correct results only if $k = n/2$. In addition, we want to encode all inputs x_i (of length $k = 256$) into a single ciphertext (with $n \geq 16{,}348$) in order to minimize the communication overhead. Since the diagonal method only requires rotations in a single direction, and by at most $k - 1$, we choose to simply encode each vector twice. While this does require more slots, we already need to choose n very large to accommodate the depth of the computation, therefore we can still easily fit many duplicated input vectors into the same vector.

4.2 *Optimizations*

While the diagonal method already requires a relatively small number of rotations, this can be improved further by using a baby-step–giant-step approach [3]. This relies upon the fact that we can split each rotation into two separate rotations and only perform the second rotation *after* aggregating vectors that require the same rotation. For a rotation of l steps, we can decompose l into $k * n_1 + j$ for $n = n_1 * n_2$ and $0 \leq k \leq n_1$, $0 \leq j \leq n_2$. We first store copies of the vector rotated by each possible j. For each k, we compute the component-wise multiplication between each of the pre-rotated vectors and the corresponding matrix diagonal. Then, we add all n_2 of these products together and rotate the resulting vector by $k * n_1$ steps. Note that this requires pre-rotating each matrix diagonal $k * n_2 + j$ steps in the *opposite* direction prior to multiplication, to account for the second rotation. However, since the matrices are available as plaintext, this cost is negligible.

The client must provide the server with the necessary Galois keys to perform the ciphertext rotations. Even though this is a one-time setup, we nevertheless want to minimize the size of these keys. During the computation, we need to rotate the vector by up to $k - 1$ steps, however choosing all $k - 1$ keys would be very space-inefficient. By default, SEAL already picks rotation keys corresponding to steps by 2^i and -2^i for $0 \leq i \leq \log n/2$, and rotations are assembled from the Non-Adjacent-Form decomposition of the number of steps required, which minimizes the number of rotations required. However, since we never need to rotate all the way to $n/2 - 1$, we choose only the powers required to reach k. Considering the decomposition of the rotations in the baby-step–giant-step approach might allow us to select even fewer rotation keys, but even with the current approach we can already reduce the key size from 247 MB to 152 MB for $n = 16{,}348$.

4.3 Results

We evaluated the performance of our implementation on a standard desktop computer with an Intel i7-8700 CPU (6 cores, up to 4.60 GHz) and 32 GB of RAM. Using CKKS as implemented in Microsoft Seal v 3.4.5, we set $n = 32{,}768$, a coefficient modulus with 880 bits and a scale of 2^{40}. We evaluated the encoding phase of the network, using x^2 as the activation function. While this naturally led to a significant blow-up in values (even when rescaling appropriately), we considered this a good test case to demonstrate that even reasonably deep circuits can be practical. The ciphertext transmitted to the server was 3.8 MB in size, and for five RNN units, the computation took a total of 90 s. In the context of a clinic appointment, latencies in the order of minutes seem more than acceptable, making this initial result a promising start.

5 Discussion

We have introduced our design and prototype for privacy-preserving outsourced translation of medical data. By focusing on short, formulaic prescription instructions we have picked an application area where FHE could feasibly be deployed in the near future. At the same time, existing makeshift solutions show the need for translation in this domain and medical surveys confirm the importance of understandable prescription instructions for positive treatment outcomes. Our initial implementation shows the feasibility of such a system and features a variety of optimizations to implement the underlying computations efficiently. The major challenge going forward is the inability to efficiently evaluate more complex activation functions and the `argmax` function in CKKS using standard FHE techniques. We consider future work in this area, especially investigating how to apply scheme switching techniques, to be of independent interest and will continue to explore in this direction.

References

1. Agency for Healthcare Research and Quality. Explicit and standardized prescription medicine instructions, December 2014.
2. Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Chimera: Combining ring-LWE-based fully homomorphic encryption schemes. Cryptology ePrint Archive, Report 2018/758, 2018. https://eprint.iacr.org/2018/758.
3. Hao Chen. Techniques in privacy-preserving machine learning. https://github.com/WeiDaiWD/Private-AI-Bootcamp-Materials/blob/master/4_Hao_Techniques_in_PPML.pdf, 2019.

4. 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.
5. Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. Cryptology ePrint Archive, Report 2019/1234, 2019. https://eprint.iacr.org/2019/1234.
6. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2016.
7. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar*, pages 1724–1734, 2014.
8. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In Maria Florina Balcan and Kilian Q Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, New York, USA, 2016. PMLR.
9. Shai Halevi and Victor Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO 2014*, pages 554–571. Springer Berlin Heidelberg, 2014.
10. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
11. 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.
12. Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. Apr 2015.
13. Qian Lou, Bo Feng, Geoffrey C Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *arXiv preprint arXiv:1911.07101*, 2019.
14. Qian Lou and Lei Jiang. SHE: A fast and accurate deep neural network for encrypted data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10035–10043. Curran Associates, Inc., 2019.
15. NR Samaranayake, Wasana Bandara, and Chinthana Manchanayake. A narrative review on do’s and don’ts in prescription label writing – lessons for pharmacists. *Integrated Pharmacy Research and Practice*, Volume 7:53–66, June 2018.
16. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL, October 2019. Microsoft Research, Redmond, WA.

HappyKidz: Privacy Preserving Phone Usage Tracking

Benjamin M. Case, Marcella Hastings, Siam Hussain, and Monika Trimoska

1 Introduction

Smartphones are indispensable parts of our daily lives. Along with adults, children are also using them for both education and entertainment. However, the adverse effects of excessive phone usage have created concerns among parents and social scientists. While these effects are observed in children and adults alike, children are considered to be more susceptible [2, 6, 9, 17, 19, 20].

To tackle this issue, several apps are designed to allow parents to oversee the phone usage of their children. A study on popular parenting apps in the Google Play Store identified two key features of these apps—remote monitoring and remote locking [12]. Another study on the acceptance of these apps among children has reported that the ratings given by the children to these apps are significantly lower than those given by the parents [8]. According to this study, children felt that the apps were overly restrictive and invasive of their privacy, which negatively impacts their relationship with parents.

B. M. Case
Facebook Inc. (work done while at Clemson University), Menlo Park, CA, USA
e-mail: bmcase@g.clemson.edu

M. Hastings (✉)
University of Pennsylvania, Philadelphia, PA, USA
e-mail: mhast@seas.upenn.edu

S. Hussain
University of California San Diego, San Diego, CA, USA
e-mail: s2hussai@eng.ucsd.edu

M. Trimoska
University of Picardie Jules Verne, Amiens, France
e-mail: monika.trimoska@u-picardie.fr

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_8

The existing apps have two limitations. First, it annoys the children, especially teenagers, who want more control over their lives, thus leading to more complex problems and worsening the situation in many cases. “Hover parenting”, has been associated with increased levels of child anxiety and depression [8, 13, 16]. Second, it is often difficult to determine when a phone usage pattern becomes unhealthy. Most mental health apps approved by the Anxiety and Depression Association of America [1] are targeted toward individual, self-guided management of existing disorders or are designed to be used in tandem with a licensed therapist. Moreover, the signs of depression in the children often go unnoticed by the parents. A poll by the University of Michigan [3] suggests that two-thirds of parents face barriers in recognizing depression in their own children.

In this work, we aim to help the parents effectively monitor the well-being of their children in a non-invasive way. We propose an app, called *HappyKidz*, that automatically collects usage data from a child’s phone and sends it to a server. The server holds a Machine Learning (ML) model that is trained collaboratively by a large number of parents as well as child psychologists and social scientists to calculate a well-being score of the child. The parents receive a periodic update of the score on their phones. In this way, instead of constantly monitoring the child’s usage, parents only need to intervene if there is a drop in the well-being score.

While this approach solves the above-mentioned limitations of the existing apps, it brings a more crucial issue—protecting the privacy of the child’s data. Allowing the server to view the raw data creates the possibility of corporate misuse, e.g., using knowledge of depressive behaviors to tailor predatory advertisements or selling health data to insurance companies or other partners. In the proposed app, to ensure the privacy of the child the collected data is encrypted locally with Homomorphic Encryption (HE) at the child’s phone before being sent to the server. The server computes the well-being score by performing ML on the encrypted data and sends the encrypted score to the parents’ phone that can decrypt it locally. This allows the parents to benefit from a well-trained ML model that is enriched by the knowledge of other parents and experts without compromising the privacy of their children.

We present a proof-of-concept implementation of the app in this paper. The proposed ML model takes as input the app usage data with the granularity of different app categories and hours of usage. It also takes the sleep pattern of the child since this is considered a strong indicator of the mental health condition [19]. This data is encrypted with HE using the Microsoft SEAL library [14]. While designing the ML model, we take into account both the precise calculation of the well-being score as well as its efficient execution through the SEAL library. In our evaluation, one inference requires $\sim$100 ms and deviates by $\sim$0.0002 from the inference result without encryption. In the current implementation, the training is performed on unencrypted data. Efficient training of any generic ML model on encrypted data is still an open problem. However, we outline a concrete methodology to train on encrypted data.

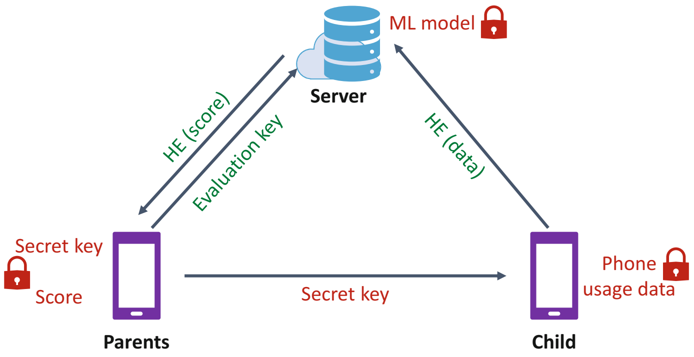


Fig. 1 Privacy model

1.1 Privacy Model

The privacy model of HappyKidz is illustrated in Fig. 1. It involves three parties: the parent, the child, and the server. The parent generates the secret key and evaluation key for HE and sends the secret key to the child and the evaluation key to the server. The phone usage data is collected at the child's phone, encrypted with HE using the parent's secret key and sent to the server. This guarantees that the server cannot access the child's data. The server who holds the ML model uses the evaluation key to compute the well-being score. The result generated by the server is an encryption of the well-being score under the parent's secret key. This result is sent to the parent who uses the secret key to decrypt it and learn the well-being score of the child. We assume that the server does not collude with parents to release additional data about the child.

2 Proof of Concept Implementation

We implemented a proof-of-concept version of this app during the 2019 Microsoft Private AI Bootcamp.[1] This section describes the details of the implementation.

2.1 Data Selection and Features

Overall, the HappyKidz app aims to evaluate well-being by measuring quantitative behavioral indicators associated with mental health issues. The proof of concept

[1] https://www.microsoft.com/en-us/research/event/private-ai-bootcamp/.

uses two commonly cited indicators: total time spent on phone apps and sleep patterns. Data from these behaviors are collected on the child's phone and consolidated into *features* that are used in the machine learning model.

Various studies have found a correlation between overall social media use and depressive symptoms [10, 18]. We define three categories of phone apps (social media, education, and games) and divide each day into three time-blocks (school hours, evening hours, and sleep hours). We aggregate the total time a child spends using apps in each category. This breakdown provides insight into appropriate phone usage. For example, a child is welcome to use social media during their evening free time, but excessive use while at school or during sleeping hours is less appropriate.

Sleep also plays a role in adolescent well-being. A variety of studies indicate links between sleep deprivation and behavioral problems in youth. Clarke and Harvey [4] suggest that improved sleep quality in adolescents with insomnia correlates with improved moods. We record the time that the child falls asleep and the total duration of sleep each night. This pair is stored locally for 3 days. Each day, we send the past three nights of sleep data to the model. This accommodates natural fluctuations in bedtimes (e.g. a child may stay up late one night to finish their homework) while still identifying longer-term patterns (e.g. a child goes to bed late every night).

These data provide 15 features each day: 9 from app usage and 6 from sleep data. The data are encrypted and uploaded to the cloud. For discussion of other potential data sources, see Sect. 3.1.

2.2 Learning Model

The app implements a model consisting of two fully connected (FC) layers. The output of the model is a wellness score between 0 and 1, where a higher score indicates positive behavioral indicators and thus good mental health.[2] In the proof of concept, we trained the model on a simulated feature vector (described in Sect. 2.1) with hand-labeled wellness scores.

Formally, our model is described as the following function, which takes the input feature vector x of length n:

$$f(x) = b_2 + W_2(s(b_1 + W_1 x)). \tag{1}$$

In this function, $b_1 \in \mathbb{R}^n$, $b_2 \in \mathbb{R}$ are bias vectors, $W_1 \in \mathbb{R}^{n \times n}$ and $W_2 \in \mathbb{R}^{1 \times n}$ are weight matrices, and $s : \mathbb{R}^n \to \mathbb{R}^n$ is the activation function (which operates element-wise on a vector). The bias vectors and weight matrices are generated during training.

[2] In the parent's app, we will color-code the wellness score for easy interpretation. High scores will be green, low scores will be red.

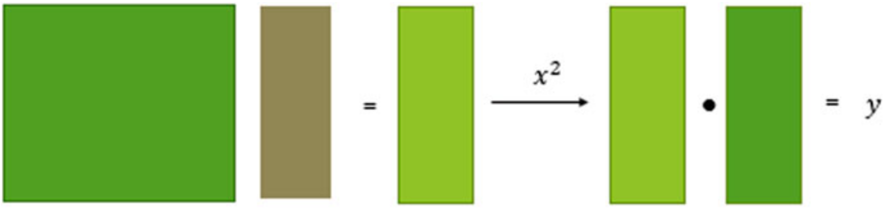


Fig. 2 Schema of the inference SEAL implementation

We define the activation function s as the square function. It provides high inference accuracy for low-depth ML models [7] and is efficient to calculate under homomorphic encryption. Given our two-layer model, this is the most suitable option.

2.3 *Microsoft SEAL Implementation*

We implemented the neural net described in Sect. 2.2 using the Microsoft SEAL [14] homomorphic encryption library. The library supports several protocols and data representations; we used the CKKS scheme [5] with a multiplicative depth of three.

As described in Eq. 1 and Fig. 2, the three main operations are a matrix-vector multiplication (W_1x), a square activation function (s), and an inner product ($W_2s(\cdot)$). The matrix-vector multiplication uses the diagonal method introduced by Halevi and Shoup [11]. The square activation function can be computed using a square-in-place homomorphic multiplication. The inner product operation is optimized to use only $O(\log N)$ rotations.

Our model is stored in plaintext as a weight matrix W_1 and a weight vector W_2. When the server receives a batched encoding CKKS ciphertext x, it computes the matrix-vector product $W_1 \cdot x$. To make this more efficient, two preprocessing steps are done. One, on the server-side the diagonals of W_1 are encoded as plaintext vectors. Second, on the child's device, the ciphertext x has the features repeated to fill all the slots. The diagonal method for the matrix-vector multiplication requires us to be able to rotate the slots of a ciphertext. In our implementation, we introduce some temporary ciphertext so that we can get to all the necessary rotations by only rotating one position each time. We can then request just this rotation in the Galois key.

```
// perform the multiplication
Ciphertext temp, temp2;
Ciphertext enc_result;
temp2 = ct; // ct = x

for (int i =0; i < dimension ; i++){
   temp = temp2;
   // multiply
   evaluator.multiply_plain_inplace(temp, ptxt_diag[i]);
   if (i == 0){
      enc_result = temp;
   } else{
      evaluator.add_inplace(enc_result, temp);
   }
   evaluator.rotate_vector(temp2, 1, galk, temp2);
}
evaluator.rescale_to_next_inplace(enc_result);
enc_result.scale() = pow(2.0, my_scale);
```

Next, we add the bias vector b_1. The activation function $s(x) = x^2$ can be applied by squaring the ciphertext in place followed by relinearization and rescaling.

```
//add bias vector b1
encoder.encode(b1, enc_result.parms_id(),scale, b1_plaintext);
evaluator.add_plain_inplace(enc_result,b1_plaintext);

//square in place
evaluator.square(enc_result, enc_result);
evaluator.relinearize_inplace(enc_result, relin_keys);
evaluator.rescale_to_next_inplace(enc_result);
enc_result.scale() = pow(2.0, my_scale);
```

Next, the inner product with the weight vector W_2 can be done by first performing a component wise multiplication and then summing the slots. To do this we need to rotate the ciphertext by powers of 2 rotations; we request these specific Galois keys be created. We follow this up with adding in the final bias correction value b_2.

```
//multiply in place
evaluator.multiply_plain_inplace(enc_result, W2);

// Sum the slots
Ciphertext temp_ct;
for (size_t i = 1; i <= encoder.slot_count() / 2; i <<= 1) {
   evaluator.rotate_vector(enc_result, i, galk, temp_ct);
   evaluator.add_inplace(enc_result, temp_ct);
}

// add bias value b2
encoder.encode(b2,enc_result.parms_id(), enc_result.scale(),
    b2_plaintext);
evaluator.add_plain_inplace(enc_result,b2_plaintext);
```

In the interest of performance, we tried to minimize the size of the CKKS parameters. We chose a polynomial of degree 8192 and a ciphertext modulus with prime factors of sizes {60, 30, 30, 60}. The communication sizes of the ciphertexts are in Table 1. There are two ways to encrypt the data on the child's device, either using a secret key that is shared with the parent's device or with a public key that

Table 1 Ciphertext sizes

Client to server (encrypting with secret key) (feature vector)	144 KB
Client to server (encrypting with public key) (feature vector)	288 KB
Server to client (wellness score)	130 KB

corresponds to the secret key on the parent's device. Encrypting with the secret key saves about a factor of 2 in ciphertext size.

Since the server needs to compute rotations, it will need a set of Galois keys. We generated the smallest set of Galois keys necessary, which includes rotations $\{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. The total size of these keys is 7.5 MB. Since the server will also be performing relinearization as part of the homomorphic computation, it will need relinearization keys, which have size 627 KB. The *evaluation key* is the name given to all the key material (relinearization keys and Galois keys) that is needed for the server to run the homomorphic computation. In total the evaluation key has size 8.1 MB but is only generated by the parent's device and sent to the server in the initial setup.

The total execution time of the homomorphic circuit is around 100 ms. With these parameters, the floating-point approximation of CKKS gives us about 4 decimal digits of precision. The full code for our implementation can be found at https://github.com/bmcase/bootcamp.

3 Soundness and Future Work

In this section, we address the feasibility of the application design, the threat model and failure cases of the app, and its practical usability.

Why Use HE?

When designing a HE application, one must compare against performing this computation locally without homomorphic encryption. We think there are a couple of reasons why using homomorphic encryption is the better option for our application. First, all the data storage can happen on the server and we can periodically send the parent long term statistical summaries of the data and also train other models to look for unhealthy trends in the long term data. Second, it may be the case that the app wants to keep the model from being easily taken by a competitor and charge a monthly subscription for the app. If this is the desire, then it is better to have the model computation done on the server. Keeping the model on the server also gives the app designers more flexibility in updating the model periodically.

How Can We Prevent Malicious Parties from Corrupting the ML?

One issue any ML application faces is collecting and maintaining accurate training data. There may be adversarial parties who wish to influence the outcome of the model: for example, a game developer may try to reclassify its products as education apps or train the model to associate higher wellness to children with increased game times.

In other use cases for machine learning with homomorphic encryption such as phishing or spam detection, the adversary has control over the target that the model is trying to identify (e.g. the phishing and spam emails), and in such a setting, it is necessary to continually retrain the model to stay up-to-date against modern threats. In our use case, healthy usage of social media, games, and educational apps does not change significantly on a short-term basis. We can train a single model via a large-scale study and use it for an extended time period without compromising its accuracy. Such a study should be done in collaboration with psychologists in settings where children already have devices (some schools have programs to provide students with devices).

The app also depends on secondary data sources, such as app classifications. This data is not used in the machine learning model but is necessary to featurize data or interpret the results. Since this is stored in the clear, we can issue updates to the child and parent apps (e.g. with new classification lists) to combat malicious behavior from app developers.

How Can We Detect If the App Is Not Functioning Correctly?
Another concern is how to incentivize correct usage by children. This app fails to be useful if, for example, the child has a secondary phone that they use for certain types of behavior. One mitigating factor is to provide high-level data for the parent, such as total time spent in each of three app categories (these statistics could also be computed using homomorphic encryption). If the parent is roughly aware of their child's typical phone usage, they should be able to identify cases where the app data doesn't correlate with the child's behavior patterns.

How Can We Customize the App to Irregular Schedules?
This app is designed to be useful for the average child, but many families have schedules that fall outside the norm. For example, home-schooled children may not have typical 9–3 school hours and varsity athletes may wake up early for team practice. One potential mitigating approach is to allow parents to define custom schedules. They can locally set expected hours for sleep, school, and evening/playtime. These are sent to the child's device and used to define the app usage features.

What Is a *Good* Well-Being Score?
Since the perception of a good well-being score may vary among parents, we do not define a concrete threshold between good and bad. Instead, we divide the scores into four ranges and color code the ranges as red, orange, yellow, and green where red indicates the worst and green indicate the best. Along with the absolute value, the changes in the well-being score is also an important indicator of the mental health of the child.

3.1 Future Work

The proof-of-concept app described in this paper is fairly limited. Future work includes producing higher-quality training data and expanding the machine learning models to provide more useful data.

Training Data and Features
We need to collect and accurately label real-world, representative data. In a commercial setting, we would collect data via a larger scientific study. Some telecom companies, including Sprint and TracFone, have partnered with public schools to provide free cell phones to students. We could work with such programs to install a preliminary app on the free phones that would collect training data. This study would have to partner with child and education psychologists or other trained professionals who could evaluate the students individually to assess their mental health and assign labels to the data. This type of study would also provide intuition to whether our two-layer model is appropriate for this setting.

Another option for collecting labeled training data is to work in partnership with parents and continually retrain the model. In this setting, we could have parents answer questions on the app regarding their child's well-being. The answers would provide new labels for their child's collected data. This approach has several issues.

- Training a model on encrypted data is prohibitively inefficient, so it would have to be done in the clear on the client-side. This might require the parent to use a more powerful device (e.g. a desktop computer) to answer questions and update the model.
- The server may be able to infer information about the client's data by comparing the model before and after an update. A typical mitigation is to send the model along a chain of parents (each of whom provides an update) before returning it to the server. However, this requires extensive communication and synchronization between individual users of the app.
- There are a variety of approaches for training a model in parallel, but these are not compatible with the privacy requirements of our application. The server would only be able to request updates from one parent (or chain of parents) at a time.
- This provides an avenue for parents to provide arbitrary or incorrect answers. We would have to compute server-side cross-validation after each retraining session to protect the model.

In the future, we may also wish to add more features to the app. For example, the SleepCycle [15] app computes a "quality" score that correlates with the measured amount of deep or REM sleep. We also wish to incorporate more granular app categories or time blocks, or data beyond sleep and phone usage. The maximum ciphertext size in the CKKS implementation is much larger than our current input vector, so it is technically simple to add more features.

Expanded Models

In the future, we would like to make longer-term evaluations about overall mental health. There are two potential approaches: We can store daily scores on the parent's phone and report monthly averages and trends. Alternately, we can store encrypted features on the cloud and train new models on the aggregates to produce long-range wellness scores. These would likely be more informative and less reactive than day-to-day snapshots.

The current app architecture provides flexibility to evaluate more complex models on the cloud. The simple two-layer network may not be appropriate for use on real data, but we can train and evaluate larger and more useful models that incorporate more data and advanced ML techniques.

4 Conclusion

We have presented the HappyKidz app that allows parents to monitor the well-being of their children through phone usage and sleep patterns. Contrary to the existing parental apps, which researchers have found to be unpopular among children, this app does not control or report phone usage of the children directly to the parents. Instead, it calculates a well-being score of the child using an ML model that is trained collaboratively by a large group of parents and experts and is deployed on a server. The app protects the privacy of the child's data by encrypting it with HE. Our proof-of-concept implementation shows that computation of the well-being score on encrypted data is practical in terms of computation time and memory usage.

References

1. ADAA reviewed mental health apps. https://adaa.org/finding-help/mobile-apps. Accessed: 3 December 2019.
2. Adriana Bianchi and James G Phillips. Psychological predictors of problem mobile phone use. *CyberPsychology & Behavior*, 8(1):39–51, 2005.
3. Sarah J. Clark, Gary L. Freed, Sekhar Deepa, Dianne C. Singer, Acham Gebremariam, and Sara L. Schultz. Recognizing youth depression at home and school. *Mott Poll Report*, 35(2), November 2019.
4. Greg Clarke and Allison G Harvey. The complex role of sleep in adolescent depression. *Child and Adolescent Psychiatric Clinics*, 21(2):385–400, 2012.
5. 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.
6. Yolanda Linda Reid Chassiakos, Jenny Radesky, Dimitri Christakis, Megan A Moreno, Corinn Cross, et al. Children and adolescents and digital media. *Pediatrics*, 138(5):e20162593, 2016.
7. Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical Report MSR-TR-2016-3, February 2016.

8. Arup Kumar Ghosh, Karla Badillo-Urquiola, Shion Guha, Joseph J LaViola Jr, and Pamela J Wisniewski. Safety vs. surveillance: what children have to say about mobile apps for parental control. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.
9. Elizabeth Hoge, David Bickham, and Joanne Cantor. Digital media, anxiety, and depression in children. *Pediatrics*, 140(Supplement 2):S76–S80, 2017.
10. Taylor Heffer, Marie Good, Owen Daly, Elliott MacDonell, and Teena Willoughby. The longitudinal association between social-media use and depressive symptoms among adolescents and young adults: An empirical reply to Twenge et al.(2018). *Clinical Psychological Science*, 7(3):462–470, 2019.
11. Shai Halevi and Victor Shoup. Algorithms in HElib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.
12. Minsam Ko, Seungwoo Choi, Subin Yang, Joonwon Lee, and Uichin Lee. Familync: facilitating participatory parental mediation of adolescents' smartphone use. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 867–878, 2015.
13. Eun Jee Lee and Yolanda Ogbolu. Does parental control work with smartphone addiction?: A cross-sectional study of children in South Korea. *Journal of addictions nursing*, 29(2):128–138, 2018.
14. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL, October 2019. Microsoft Research, Redmond, WA.
15. Sleep Cycle AB. Sleep cycle: Sleep analysis & smart alarm clock. https://www.sleepcycle.com/, 2020.
16. Holly H Schiffrin, Miriam Liss, Haley Miles-McLean, Katherine A Geary, Mindy J Erchull, and Taryn Tashner. Helping or hovering? the effects of helicopter parenting on college students' well-being. *Journal of Child and Family Studies*, 23(3):548–557, 2014.
17. Mercedes Sánchez-Martínez and Angel Otero. Factors associated with cell phone use in adolescents in the community of Madrid (Spain). *CyberPsychology & Behavior*, 12(2):131–137, 2009.
18. Jean M Twenge, Thomas E Joiner, Megan L Rogers, and Gabrielle N Martin. Increases in depressive symptoms, suicide-related outcomes, and suicide rates among us adolescents after 2010 and links to increased new media screen time. *Clinical Psychological Science*, 6(1):3–17, 2018.
19. Fangbiao Tao, Liwei Zou, Xiaoyan Wu, Shuman Tao, Honglv Xu, Yang Xie, and Yajuan Yang. Mediating effect of sleep quality on the relationship between problematic mobile phone use and depressive symptoms in college students. *Frontiers in Psychiatry*, 10:822, 2019.
20. Kimberly S Young and Robert C Rogers. The relationship between depression and internet addiction. *Cyberpsychology & behavior*, 1(1):25–28, 1998.

i-SEAL^2: Identifying Spam EmAiL with SEAL

I. Demertzis, D. Froelicher, N. Luo, and M. Norberg Hovd

1 Introduction

End-to-end encrypted emails are desirable with regards to privacy, as it prevents your email provider from storing and reading your emails in plaintext. However, with the perk of privacy from the end-to-end encryption, you lose the spam filter, as the filtering process requires an analysis on the email's content, or its metadata. The classification of whether an email is spam typically relies on machine learning algorithms that have been trained on large amounts of emails. A naive approach to combine end-to-end encryption of emails and a spam filter would be for every user to simply build their own model using only their own emails to train the machine learning model. However, one user typically only has a limited number of emails and this local approach is going to result in a model which is less accurate than the one provided by an email provider, simply due to the size of the dataset used to train the machine learning model. In order to obtain an accurate model, large amounts of diverse data are required.

I. Demertzis
CSE, University of California, Santa Cruz, CA, USA
e-mail: idemertz@ucsc.edu

D. Froelicher (✉)
Laboratory for Data Security, EPFL, Lausanne, Switzerland
e-mail: david.froelicher@epfl.ch

N. Luo
Computer Science, Yale University, New Haven, CT, USA
e-mail: ning.luo@yale.edu

M. N. Hovd
Institute of Informatics, University of Bergen, Bergen, Norway
e-mail: martha.hovd@uib.no

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_9

Another approach would consist in sharing only spam emails, which usually do not contain sensitive or private information, with the email provider. This approach solves the previous problem of having a too small training set. However, it would result in a one-class classification scenario, as the machine learning model would be trained on distinguishing spam and ham emails by being trained on a set containing mostly spam emails. This will result in a less accurate model than one trained on a dataset with an even (or less biased) distribution of both types of emails.

In this short paper, we suggest to rely on homomorphic encryption to circumvent this problem, and propose a solution that enables privacy-preserving classification of emails as spam or ham. We also describe a solution for an oblivious training of spam detection machine learning model that enables the training of accurate detection model without hindering the privacy of the users.

2 Private Classification

In Fig. 1, email users want to obtain a classification of their emails as spam or ham. Email providers, e.g., Google or Microsoft, already have (cleartext) classification models, which are trained on millions of users' emails and can be used to perform this classification. However, in today's solution, this requires them to access the private content of the emails.

In order to avoid this, we propose a solution in which the user sends encrypted information, generated from the received emails, i.e., a vector of encrypted features, to the service provider and receives in return a (encrypted) classification of the

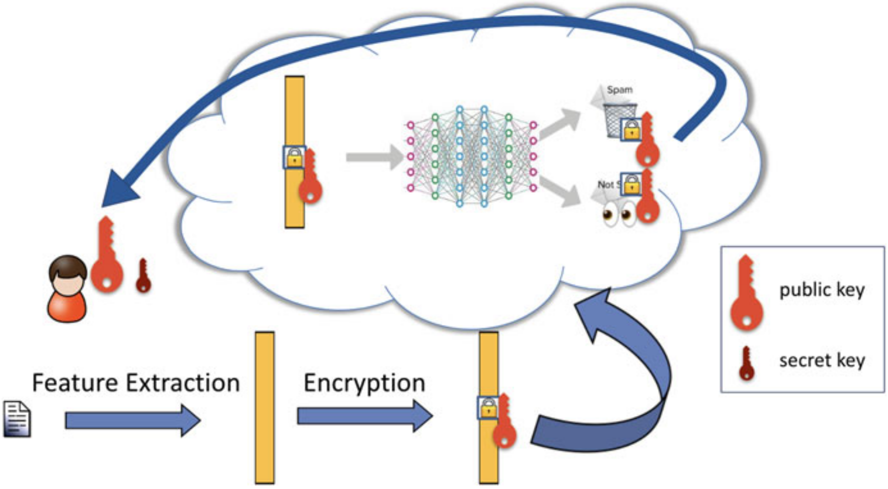


Fig. 1 Private classification

email. This is executed without revealing the content of the emails to the service provider.

When receiving an encrypted email, the user decrypts it and (automatically) generates a vector of features capturing its content. This features' vector is then encrypted under the client's public key and sent to the email provider, who classifies the email as spam or ham by using its cleartext model on the received encrypted feature vector. The provider then sends the encrypted classification back to the client, who decrypts it using its secret key and obtains an automatic and privacy-preserving classification of his emails.

3 Private Training

While private classification enables users to benefit from the automatic spam detection without hindering their privacy, it also results in service providers not obtaining more data in order to train and update their detection model. To counter this issue, we propose a solution that enables the service providers to train their model on encrypted data.

As presented in Fig. 2, email users send a feature vector of their email encrypted under the collective public key of the service providers. This key is a combination of the providers' public keys and ensures that the decryption of a ciphertext under the collective public key is only possible when all the service providers collaborate in the decryption, by using their own secret key. The encrypted material will only be decrypted once all the service providers have "partially" decrypted by using their respective secret keys.

The service provider trains and periodically updates an encrypted model for spam detection by using the encrypted vectors collected from the email users.

The private classification can then be performed as presented before, using the updated model. We observe here that this model can be periodically decrypted such

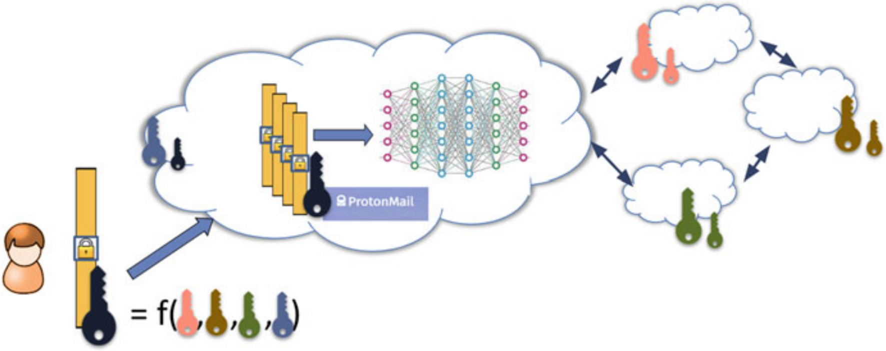


Fig. 2 Private training

that the service provider can use the cleartext model to perform the classification on encrypted data. Alternatively, at the end of every period, the updated (and decrypted) model is sent to the users, which can perform the email classification locally.

We also note that users may choose to disclose emails in plaintext to the provider for training purposes. This simplifies the continuous model training, as it is partially executed on cleartext data, thus reducing the computation complexity. With this alternative, one may train the model on plaintext spam and non-sensitive ham emails, and use homomorphic encryption for training the model only on sensitive emails, where the users themselves choose which emails are too sensitive to share in plaintext.

4 Conclusion

We propose a system that enables both the privacy-preserving classification of emails as spam or ham and the secure training of the classification model, thus providing a solution that responds to the growing usage of end-to-end encrypted emails and the increasing demand for privacy-preserving solutions.

The main remaining technical challenge lies in choosing the machine learning model and combining it with a cryptographic protocol. It is not enough for such a model to be efficient in classifying whether or not an email is spam, it should also be crypto-amenable, i.e., it should be executable through a homomorphic encryption protocol. As an example, it requires that the model's computations can be approximated with polynomial functions with a low multiplicative depth.

The primary challenge for adopting this solution in practice is probably creating a compelling (financial) incentive for email providers to offer this service. The data gathered from the analysis that service providers currently are able to run on plaintext emails have an enormous business value, as the gathered data is integral to the highly profitable targeted advertisements. Moreover, having important service providers such as Microsoft and Google to collaborate in order to distribute the trust is particularly difficult and a promising alternative would be to rely directly on the users for this task. This poses other challenges, though, most notably the management of keys and the availability of end-users. These and other challenges are interesting future works.

PRIORIS: Enabling Secure Detection of Suicidal Ideation from Speech Using Homomorphic Encryption

Deepika Natarajan, Anders Dalskov, Daniel Kales, and Shabnam Khanna

1 Introduction

Suicidal ideation, or the state of thinking about or planning a suicide, is a major public health concern in the United States. In 2015 alone, an estimated 9.8 million adults in the US reported having serious suicidal thoughts [1]. Moreover, according to the United States Center for Disease Control, the national suicide rate increased by 33% between 1999 and 2017 [8]. Early detection of suicidal ideation is critical to prevent suicide attempts and provide treatment for individuals. Yet, in spite of major advances in the fields of medical and psychological science, our ability to predict suicide has remained roughly constant for at least several decades [9].

Clinical practitioners typically rely on self-report of suicidal thoughts in order to diagnose suicidal patients. However, this method of diagnosis is problematic, since a majority of individuals who die from suicide deny suicide ideation in their last communication about the subject before death [3, 16]. Additionally, the current system relies on clinical assessment as a primary means of identifying suicidal ideation. This means that individuals who do not make a habit of regular clinical assessments, for instance, due to concerns about cost of treatment, time constraints,

D. Natarajan (✉)
University of Michigan, Ann Arbor, MI, USA
e-mail: dnataraj@umich.edu

A. Dalskov
Aarhus University, Aarhus, Denmark

D. Kales
Graz University of Technology, Graz, Austria
e-mail: daniel.kales@iaik.tugraz.at

S. Khanna
Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, UK
e-mail: skhanna01@qub.ac.uk

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_10

lack of access, feelings of depression/lack of motivation, or social stigma, do not receive adequate diagnosis and treatment [7, 22].

In order to address some of these inefficiencies, Gideon et al. [13] proposed a machine learning-based system for detecting suicidal ideation. By determining the emotions present in a subject's natural phone conversations, and noting that individuals with suicidal ideation displayed lower emotional variability than healthy controls, the authors were able to create a machine learning model that could predict the likelihood of suicidal ideation in an individual.

From a security perspective, both the phone call data and the prediction output are extremely sensitive in nature and require complete confidentiality. Any leak of medical data could dramatically affect the patient's well-being, whether through resulting social stigma, discrimination by employment or financial institutions, or other types of abuse. The approach taken by Gideon et al. to safeguard user data involves sending encrypted data from the user's phone to a server compliant with U.S. patient privacy laws, which is then able to decrypt and process the user data in the clear. Though this approach may be sufficient for a limited number of users, assuming a heavily safeguarded, small number of private servers used to process the user data, it does not scale well as the number of users increases.

For many applications, a larger user base may signal the need to move from a small private infrastructure to a larger cloud-like environment, where equipment and maintenance costs can be outsourced and/or shared amongst multiple cloud customers. However, many works have shown how seemingly secure cloud-based systems are often easily exploitable by attackers, for example, due to the difficulty of detecting bugs in large cloud operating-systems, or via the use of side-channel attacks [15, 27]. Thus, though it solves the problem of scalability, this approach has the potential to violate user security and privacy.

Recently, researchers at Microsoft have demonstrated the feasibility of using Homomorphic Encryption to securely outsource Neural Networks predictions for MNIST and CIFAR datasets [2, 5, 10]. In this work, we investigate the feasibility of securely detecting suicidal intent from speech data using privacy-preserving homomorphic encryption techniques.

Overview We begin by describing an end-to-end flow for suicidal ideation detection, first shown to be effective in [13]. We then describe a privacy-preserving cyclical system of evaluations to further improve suicidal ideation detection and treatment. We discuss at a high level the trade-offs that would need to be considered for this implementation and give a first-order approximation of the performance of our proposed approach, using [2] as a reference for homomorphic operation performance. Finally, we discuss extensions of this work and identify several interesting areas for future analysis.

2 Suicide Ideation Detection

Researchers at the University of Michigan have demonstrated the feasibility of detecting suicide ideation from natural phone call speech data using their PRIORI smartphone application [13]. In their analysis, the researchers make use of two main neural networks: a convolutional neural network (CNN), consisting of a feature encoder and an emotion classifier, and a dense neural network (DNN). The CNN and DNN are shown in Figs. 1 and 2, respectively. We use their process as a basis for our proposed application, and describe key components of their approach below.

2.1 Dataset

The Ecological Measurement of Affect, Speech, and Suicide (EMASS) dataset is a collection of natural phone conversations and regular reports of emotion, mood, and suicidal thoughts. Specifically, it consists of over 400 h of phone conversations recorded from 43 different participants, including healthy control samples and patients who experience suicidal ideation. The calls were recorded by the PRIORI phone application over the course of 8 weeks. The authors of [13] use the EMASS dataset to train and test their models. The collection of this dataset is still ongoing; however, the authors plan to publish the extracted EMASS dataset features upon completion of the study.

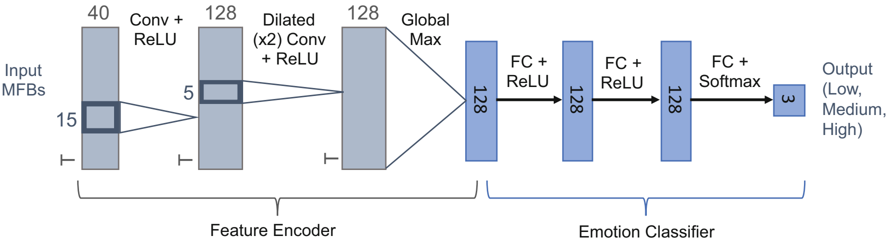


Fig. 1 MADDoG Convolutional Neural Network, which consists of a Feature Encoder (left) and an Emotion Classifier (right). Figure adapted from [12]

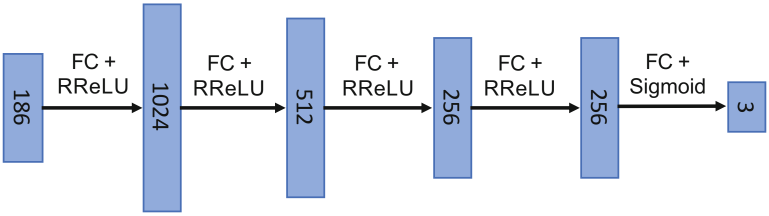


Fig. 2 Dense Neural Network used for Emotion Identification, as described in [13]

2.2 Application

The PRIORI application, as described in [13], utilizes neural networks to perform various operations. For convenience, we give the steps of the application flow below (for inference only):

1. Use the PRIORI phone application to save user-side call audio.
2. Use an algorithm for speech activity detection (such as the COMBO-SAD algorithm [23]) to extract short segments of uninterrupted speech from a call.
3. Divide each segment into overlapping frames and extract a 40-dimensional log Mel-filter bank (MFB) spectrum. This will result in a matrix of 40-by-t_i, where t_i is the number of frames in segment i.
4. Pad the above matrix with enough zero vectors to get a 40-by-T matrix, where T is the maximum number of frames in a segment for all segments in the training set.
5. Feed the 40-by-T matrix into two separate MADDoG Feature Encoders to get "segment-level" representations of the data.
6. Feed the outputs from the MADDoG Feature Encoder into two separate MADDoG Emotion Classifiers (one for valence and one for activation). Each classifier output will be a 3-element vector denoting "low", "medium", or "high" for valence or activation, resulting in a 6-element result. See Fig. 1 for more details.
7. Repeat the above two steps for all segments in a call. This will result in a 6 × N matrix, where N is the number of segments in a call.
8. Take 31 statistics (including mean, standard deviation, skewness, kurtosis, min, max, range, and statistics from performing a linear regression) across each row in the matrix from the previous step. This will result in a final feature vector of $6 \times 31 = 186$ elements per call.
9. Feed the 186-dimensional vector into five separate DNNs, where each DNN is trained to classify one of the following emotions: Guilt, Hopelessness, Anger at Others, Anger at Self, and Irritability. Each DNN consists of four hidden layers with widths of 1024, 512, 256, and 256, respectively. The activation function of the hidden layers is a RReLu (Randomized Leaky ReLu), which corresponds to a LReLu (Leaky ReLu) during model evaluation. The final layer uses a sigmoid activation function and outputs a 3-element vector, denoting a rating of 0, 0.5, or 1. These ratings correspond to ratings on a Likert scale of 1–5 for emotion intensity.
10. Repeat the above steps for a set of calls. For each of the five emotions, calculate the (within-subject) standard deviation. This is representative of emotion variability across a set of calls.
11. Take the average of the five standard deviation values found in the previous step. Use the result as a measure of the average emotion variability over a set of calls for a particular individual.
12. Use either a threshold or a linear classifier to determine whether the output from the previous step indicates suicidal ideation.

3 Use Cases

As mentioned previously, the PRIORI application-based system described in [13] sends raw speech audio to a remote server to be processed in the clear. This speech data and the resulting network prediction constitute highly sensitive information, especially since the PRIORI application records all (user-side) natural phone call conversations during day-to-day life.

We propose modifications to the PRIORI application flow that would allow for a more secure approach to suicide ideation detection. Namely, our approach would protect all user-created data as well as the result of the suicide ideation prediction from cloud adversaries. We call this approach "PRIORIS" to refer to a secure version of the PRIORI approach. In this approach, the ideation detection flow would be segmented as follows:

1. Audio recording, speech activity detection, and MFB extraction
2. Evaluation of MADDoG Feature Encoders and Emotion Classifiers
3. Calculation of statistics across result
4. Evaluation of Emotion DNNs
5. Calculation of standard deviation, average, and threshold/linear classifier

We envision that steps 1, 3, and 5 would be computed in-the-clear on the local smartphone device, while steps 2 and 4 (i.e. the neural networks) would be calculated homomorphically in the cloud. This would add an additional homomorphic encryption step and data communication step between steps (1,2) and (3,4), as well as an additional homomorphic decryption step and data communication step between steps (2,3) and (4,5).

Preserving the privacy of this speech data could persuade more people to use emotion detection recognition technology outside the context of clinical studies. Consequently, the security guarantees afforded to the application flow by our proposed modifications could render a variety of new opportunities for secure deployment. We identify three such use cases and describe them below:

3.1 Use-Case 1: Secure Detection and Response

In this scenario, the goal of the application would be to understand and respond to the mental health status of an individual. For example, when the application predicts that a user is experiencing suicidal inclinations, it could alert the user and recommend a clinical visit, potentially even displaying locations, hours of operation, and/or open appointment slots for nearby clinics. In more extreme cases (e.g. when the prediction of suicidal ideation is strong), the application could display the phone numbers of suicide prevention hotlines or even immediately connect an individual to a hotline volunteer or trained professional.

3.2 Use-Case 2: Secure Clinical Assessment Assistance

The concept of using speech patterns to identify mood disorders is not new; clinicians typically consider speech factors such intonation, conversation dominance, and voice level (e.g. quiet, loud) when diagnosing patients with mood disorders [14, 26]. Figure 3 (Block 2) shows how the application could be used to augment the capabilities of clinicians to understand the mental health of their patients from speech-level information.

Importantly, this application would allow professionals to take into account predictions made over a larger group of people. In the case that the application prediction matches that of the clinician, this could help a clinician be more confident in their diagnosis. In the case where the prediction differs, the clinician could recommend a follow-up screening. In the controlled setting of an in-person appointment, special recording equipment can be used instead of the patient's phone application.

We note that there may be differences between the ability of the network to predict suicidal ideation from structured speech (i.e. question-response, clinical assessment) versus natural speech (i.e. phone calls to friends and family). In this case, the model used for clinical assessment would have to be trained differently from the model used in use-cases 1 and 3. This will need to be investigated in future work.

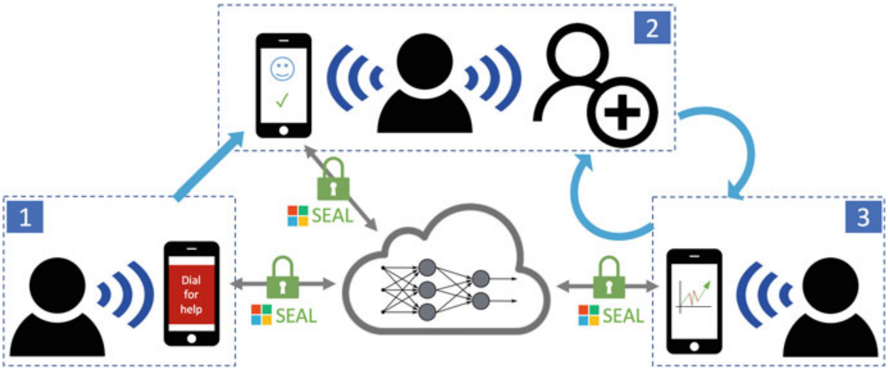


Fig. 3 Use-cases for proposed secure suicide ideation detection application. Each block number corresponds to the use-case of the same number: (1) Suicide hotline connection upon detection of suicidal ideation, (2) Validation of clinical assessment, (3) Monitoring of treatment effectiveness over time. Use cases may be related to each other as depicted by (light blue) arrows between blocks. Arrows between blocks and cloud denote HE-based data encryption and model evaluation, using Microsoft SEAL library as an example HE infrastructure

3.3 Use-Case 3: Secure Treatment Evaluation

In many cases, suicidal ideation can be linked to a mental health disorder which can be treated [6]. As is the case with many mental health disorders, the treatment procedure is usually a very iterative process [17]. For example, this may take the form of trying a certain dose of a medication for a month, re-evaluating symptoms at another clinical visit, adjusting the medication dose, re-evaluating symptoms at a clinic again, one month later, etc. Additionally, psychotherapy may be used to a varying degree before the best therapy schedule is determined.

As mentioned earlier, relying on patients to self-report suicidal ideation is problematic, as a majority of people who die from suicide deny suicide ideation in their last communication before death [4]. In addition to outright denial, patients may simply be unable to detect suicidal behavior in themselves, simply because they have not been trained to do so. Moreover, a patient may rely on memory alone to describe how their mental health has been affected as a result of the particular treatment iteration. This reliance on patient memory is problematic, as it is unrealistic to expect an individual to remember every detail of their mood changes across extended periods of time.

To help solve the above problems, the application could be used to track effectiveness of treatment over time, for example, by recording the suicidal ideation prediction values over the course of a month and plotting the change in values on a graph, as shown in Fig. 3 (Block 3). Downward trends could be interpreted as relative ineffectiveness of a treatment iteration, while upward trends could be interpreted as relative effectiveness of the iteration. The clinician could evaluate effectiveness of treatment without having to rely solely on patient recollection at the time of visit. As in use-case 1, the application may also respond with helplines or clinic availability for particularly strong predictions of suicidal inclinations.

We note that the above use cases could be proposed without any notion of security. However, we argue that the data input and application output constitute extremely sensitive information, and users would not use the application without robust security guarantees. Thus, the use cases we discuss are only possible at scale with the type of protection offered by the secure network evaluation we propose.

We also note that the above use cases are related. An individual may initially use the application for a preliminary diagnosis to decide whether they should seek further evaluation from a clinician (use-case 1). During the clinical visit, the clinician may use the application to confirm their diagnosis, utilizing the results from case 1 where helpful (use-case 2). If diagnosed with a mental illness associated with suicide ideation, the individual would use the application to monitor the effectiveness of the initial treatment plan (use-case 3). The next clinical appointment would involve use-case 2 followed by use-case 3 once again. In this way, the application could be used in a cyclical manner to enable a more accurate, effective, and efficient treatment process.

4 Network Training

This work mainly focuses on homomorphic evaluation of the described networks. Accordingly, we assume that the models referenced are trained beforehand. Nevertheless, we wish to devote some discussion as to how such models could be obtained in practice.

The authors of [13] have already demonstrated how useful models can be generated using the EMASS dataset, which we summarized in Sect. 2. The models they were able to train using the EMASS dataset have proven successful at using natural phone conversations to distinguish healthy controls from suicidal individuals, achieving an AUC (Area Under the Curve) of 0.79. Datasets such as EMASS could therefore be used to build initial networks.

For optimal performance, it is likely that much more data would need to be collected in order to further train the initial models. We imagine that successful deployment of this application would encourage enough users to volunteer their data for network training. However, the inputs and outputs of the networks described above contain highly sensitive data; thus, it may not be plausible that enough users would be willing to volunteer this information. Moreover, it is possible that selecting volunteers in this manner would significantly skew the set of training data such that it no longer resembles testing data (for example, users may only allow evaluation of more “benign” calls, such as those made to customer service lines, rather than calls they make to family and friends).

Ideally, we would like to collect enough useful data from users while protecting user privacy. In order to ensure patient privacy during the training process, a variety of approaches could be taken. Federated learning, for example, which has been popularized in recent years by Google, could allow models to be trained locally and combined later in a privacy-preserving manner [24]. Homomorphic training could also be used to preserve patient privacy. We note, however, that while some works show homomorphic training as possible, other works report the technique as practically infeasible [21]. Future work would therefore require a much deeper analysis of this component.

5 Homomorphic Network Evaluation

The PRIORI application flow involves the use of two types of neural networks: a CNN (which consists of a Feature Encoder and an Emotion Classifier) and a DNN (used to identify emotions). We now wish to analyze the amenability of each of these networks to homomorphic evaluation.

We follow the approach of CryptoNets [10], which first described how to homomorphically process each layer of a CNN used to classify MNIST images. Specifically, we approximate the ReLU activation functions with square activation functions (i.e. a low-degree polynomial), replace “pool” layers with “scaled pool” layers, and do not homomorphically evaluate any final sigmoid activation layers. We also make two further modifications: (1) we do not homomorphically evaluate any final softmax layers, since, like the sigmoid layers, these are necessary for training but not required for evaluation, and (2) we replace RReLU activations with ReLU activations (which we approximate with square activations), since these operations are similar given limited RReLU leakage [25]. We also model the dilated convolution layer the same way as a convolution layer, since both are implemented as a weighted sum. A more detailed description of the RReLU approximation is given in the next section.

Tables 1 and 2 give the modified layers for the described CNN and DNN, respectively, as well as their per-layer homomorphic evaluation runtimes. We obtain these estimates through simple scaling of the execution times of similar layers used in CryptoNets 3.2 [2], which uses the BFV encryption scheme. The CryptoNets 3.2 numbers were obtained from running the CrytoNets application on a single Intel Xeon E5-1620 CPU at 3.5GHz, with 16GB of RAM and Windows operating system. Note that these times assume that model weights and bias values are unencrypted. We set the CNN Feature Encoder dimension T to 600, which corresponds to a 6 s average segment length and 10 ms frame shift length for MFB extraction.

Using the first order approximation, we obtain full network evaluation time estimates of 16,777.178 and 194.656 s for the CNNs and DNNs, respectively. The authors of [19] use a dataset similar to the EMASS dataset for monitoring mood from speech data and report that the phone calls made consists of 24.3 ± 46.6 segments on average. Assuming a similar 24 segments per call, sequential application of the CNN should take approximately 402,652.272 s (111.848 h) per call. Sequential application of the DNN should take approximately 194.656 s (3.244 min) per call. We stress that these estimates do not include any batching, pipelining, or parallelization techniques, each of which are expected to provide significant performance benefits (up to multiple orders of magnitude). Simply processing each of the 24 segments in parallel, for example, would result in only 4.66 h per call for application of the CNN.

Table 1 Layers proposed for homomorphic evaluation of MADDoG Feature Encoder and Emotion Classifier CNN and corresponding first order approximations of evaluation times. Results were obtained through simple scaling of execution times for similar layers used by the CryptoNets v3.2 MNIST CNN [2], and assume $T = 600$

HE layer	Time estimate (s)
Conv.	11,247.291
Square activation	886.156
Dilated conv.	3749.097
Square activation	886.156
Scaled max pool	8.478
FC	3.072
Square activation	1.477
FC	3.072
Square activation	1.477
FC	0.072
Total	6476.331

Table 2 Layers proposed for homomorphic evaluation of Emotion Detection DNN and corresponding first order approximations of evaluation times. Results were obtained through simple scaling of execution times for similar layers used by the CryptoNets v3.2 MNIST CNN [2]

HE layer	Time estimate (s)
FC	35.712
Square activation	11.815
FC	98.305
Square activation	5.908
FC	24.576
Square activation	2.954
FC	12.288
Square activation	2.954
FC	0.144
Total	194.656

Activation Functions As stated above, we follow the approach of CryptoNets and replace ReLU with square activation functions. Although such low-order polynomials can be used to approximate ReLU activations, there are places in which the functions differ significantly. It is therefore vital to empirically evaluate whether a such an approximation still achieves accurate results. The CryptoNets work achieved an accuracy of 99% using this square activation approximation of ReLU (for an MNIST classification network). Therefore, it is plausible that this approximation could be used successfully in the networks we describe as well.

The case of RReLU, however, is more complicated. As noted above, we chose to replace RReLU with regular ReLU activations (which we then approximate with square). When the choice of leakage is small, the two functions are similar (i.e. $\mathsf{RReLU}_\alpha(x) = \max(\alpha x, x) \approx \max(0, x) = \mathsf{ReLU}(x)$ for small α.) The authors of [13] do not specify the particular α they use, though they do refer to [25] (which explores small α values, between 0.01 and 0.2) as motivation for the choice of activation function. Nonetheless, the authors of [13] do not compare the accuracy they achieved with RReLU to accuracy possible with ReLU. This should be explored in future work.

Finally, we note that while the proposed approximation could still yield accurate results, it may render training the network more difficult. As noted in [10], in particular, the derivative of x^2 is not bounded. Although this may result in strange behavior during gradient descent, the authors of [10] have successfully combated this issue by adding extra convolution layers without activation layers to prevent overfitting. The effect of this approximation on network training should be assessed in future work.

6 Extensions and Future Work

In the previous sections, we described PRIORIS application for secure suicide ideation detection in the context of three main use-cases. In this section, we describe some additional extensions to the application and opportunities for future work.

Adaptation of Application to Other Types of Mood Disorder Detection Suicide ideation is highly related to mood disorders, which in turn often result in altered speech patterns. This suggests that speech data may be used to identify other types of mood disorders. In fact, this type of analysis has already been shown useful for detecting disorders such as bipolar disorder, depression, and post-traumatic stress disorder [18, 20]. Future work could analyze other types of mood disorder detection for their amenability to FHE-based private-preserving machine learning. Generalizing further, it may even be useful to construct a general mood detection and tracking application in combination with therapeutic smartphone applications such as those commonly used for mindfulness and relaxation.

Determining Optimal Intervention To improve the usefulness of the proposed system, it would be beneficial to identify exact moments when medical intervention is required. The authors of [11] have already explored this question in the context of bipolar disorder. Their method involves using an initial data collection period to establish a baseline emotion level. The authors then use anomaly detection techniques to compare subsequent user behavior relative to this baseline in order to determine optimal medical intervention points. This type of investigation and fine-tuning could be extended to the context of suicide ideation detection and treatment. This is particularly significant for uses cases involving smartphone monitoring of medical symptoms, as these devices have the potential to provide intervention close to the time of need (see: for example, suicide prevention hotline connection in use-case 1 above).

Choice of FHE Scheme We perform analysis of the layer execution times with respect to the BGV homomorphic encryption scheme, since this is the scheme used by the CryptoNets 3.2 MNIST network. Microsoft SEAL also implements the CKKS scheme, which differs from BGV in its ability to efficiently compute approximate computations on real-valued data. CKKS is thus particularly amenable to machine learning use cases, as neural networks typically make use of approximate values. Future work should investigate the performance of the using the BGV scheme relative to using the CKKS scheme for the proposed application.

Analysis of Detection Segmentation Flow In Sect. 2.1, we proposed an initial segmentation of the application flow with respect to computation execution location. However, this initial segmentation is based on intuition. Future work should analyze the trade-off between executing each step of the full procedure either locally in the clear or homomorphically on the server, including the resulting impact on overall performance, energy, and storage requirements of the application.

The modifications we propose as a result of this analysis, as well as the choice of FHE scheme, will likely result in variable model accuracy. However, any loss in accuracy could potentially be offset by increasing the amount of available training data. Our proposed application places a strong emphasis on user privacy guarantees, and would thus encourage more widespread adoption of the PRIORIS application. This, in turn, could increase the amount of training data available (e.g. by informing more users about the study, some of whom may volunteer to have their data added to the training set, or via the secure training techniques mentioned in Sect. 4) and render the system robust enough to be integrated into everyday mood health monitoring and clinical assessment.

7 Conclusion

In this work, we propose a privacy-preserving based suicidal ideation detection flow. We describe how homomorphic encryption could be used for secure inference of networks previously shown useful for detecting suicidal ideation from phone speech data. We also describe multiple use-cases that are enabled as a result of the proposed security mechanisms. Finally, we give first-order approximations of homomorphic evaluation runtimes for the models used in our application, and describe several directions for future research.

Acknowledgments This work was created in collaboration with researchers at Microsoft. The authors would like to thank the organizers of Microsoft's Private AI Bootcamp for their helpful feedback and reviews. The authors would also like to thank researchers in the Computational Human Artificial Intelligence (CHAI) Lab at the University of Michigan for their helpful discussions. (Note: Authors of this work are listed in contributing order.)

References

1. Substance Abuse, Mental Health Services Administration, and Center for Behavioral Health Statistics. Suicidal thoughts and behavior among adults:results from the 2015 national survey on drug use and health.
2. Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. Low latency privacy preserving inference. In *International Conference on Machine Learning*, 2019.
3. Katie A Busch and Jan Fawcett. A fine-grained study of inpatients who commit suicide. *Psychiatric Annals*, 34(5):357–364, 2004.
4. Katie A Busch, Jan Fawcett, and Douglas G Jacobs. Clinical correlates of inpatient suicide. *The Journal of clinical psychiatry*, 2003.
5. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 483–512. Springer, 2018.
6. Louise Brådvik. Suicide risk and mental disorders, 2018.
7. Ewa K Czyz, Adam G Horwitz, Daniel Eisenberg, Anne Kramer, and Cheryl A King. Self-reported barriers to professional help seeking among college students at elevated risk for suicide, 2013.
8. Centers for Disease Control and Prevention. Webbased injury statistics query and reporting system (WISQARS). Online, accessed 2020-01-21. Available at URL: https://www.cdc.gov/injury/wisqars/index.html.

9. Joseph C Franklin, Jessica D Ribeiro, Kathryn R Fox, Kate H Bentley, Evan M Kleiman, Xieyining Huang, Katherine M Musacchio, Adam C Jaroszewski, Bernard P Chang, and Matthew K Nock. Risk factors for suicidal thoughts and behaviors: a meta-analysis of 50 years of research. *Psychological bulletin*, 143(2):187, 2017.
10. 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 *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016.
11. John Gideon, Katie Matton, Steve Anderau, Melvin G McInnis, and Emily Mower Provost. When to intervene: Detecting abnormal mood using everyday smartphone conversations, 2019.
12. J. Gideon, M. McInnis, and E. Mower Provost. Improving cross-corpus speech emotion recognition with adversarial discriminative domain generalization (ADDoG). *IEEE Transactions on Affective Computing*, 2019.
13. John Gideon, Heather T Schatten, Melvin G McInnis, and Emily Mower Provost. Emotion recognition from natural phone conversations in individuals with and without recent suicidal ideation. In *The 20th Annual Conference of the International Speech Communication Association INTERSPEECH 2019*, 2019.
14. William Guy. *ECDEU assessment manual for psychopharmacology*. US Department of Health, Education, and Welfare, Public Health Service ..., 1976.
15. Mehmet Sinan Inci, Berk Gulmezoglu, Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. Cache attacks enable bulk key recovery on the cloud. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 368–388. Springer, 2016.
16. Erkki T Isometsä, Martti E Heikkinen, Mauri J Marttunen, Markus M Henriksson, Hillevi M Aro, and Jouko K Lönnqvist. The last appointment before suicide: is suicide intent communicated? *The American journal of psychiatry*, 1995.
17. Douglas G Jacobs, Ross J Baldessarini, Yeates Conwell, Jan A Fawcett, Leslie Horton, Herbert Meltzer, Cynthia R Pfeffer, and Robert I Simon. Assessment and treatment of patients with suicidal behaviors. 2010.
18. Z. N. Karam, E. M. Provost, S. Singh, J. Montgomery, C. Archer, G. Harrington, and M. G. Mcinnis. Ecologically valid long-term mood monitoring of individuals with bipolar disorder using speech. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4858–4862, May 2014.
19. Zahi N Karam, Emily Mower Provost, Satinder Singh, Jennifer Montgomery, Christopher Archer, Gloria Harrington, and Melvin G Mcinnis. Ecologically valid long-term mood monitoring of individuals with bipolar disorder using speech. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4858–4862. IEEE, 2014.
20. Charles R. Marmar, Adam D. Brown, Meng Qian, Eugene Laska, Carole Siegel, Meng Li, Duna Abu-Amara, Andreas Tsiartas, Colleen Richey, Jennifer Smith, Bruce Knoth, and Dimitra Vergyri. Speech-based markers for posttraumatic stress disorder in us veterans. *Depression and Anxiety*, 36(7):607–616, 2019.
21. Karthik Nandakumar, Nalini K. Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *CVPR Workshops*, page 0. Computer Vision Foundation/IEEE, 2019.
22. Thomas Niederkrotenthaler, Daniel J Reidenberg, Benedikt Till, and Madelyn S Gould. Increasing help-seeking and referrals for individuals at risk for suicide by decreasing stigma: The role of mass media. *American journal of preventive medicine*, 47(3):S235–S243, 2014.
23. S. O. Sadjadi and J. H. L. Hansen. Unsupervised speech activity detection using voicing measures and perceptual spectral flux. *IEEE Signal Processing Letters*, 20(3):197–200, March 2013.
24. Aaron Segal, Antonio Marcedone, Benjamin Kreuter, Daniel Ramage, H. Brendan McMahan, Karn Seth, Keith Bonawitz, Sarvar Patel, and Vladimir Ivanov. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.

25. Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.
26. Robert C Young, Jeffery T Biggs, Veronika E Ziegler, and Dolores A Meyer. A rating scale for mania: reliability, validity and sensitivity. *The British journal of psychiatry*, 133(5):429–435, 1978.
27. Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 305–316, 2012.

Gimme That Model!: A Trusted ML Model Trading Protocol

Laia Amorós, Syed Mahbub Hafiz, Keewoo Lee, and M. Caner Tol

1 Introduction

Machine learning (ML) has achieved a tremendous success in making breakthroughs in various real-life problems from areas such as medicine, finances or social sciences, to mention a few. It has created a lucrative business model called Machine-Learning-as-a-Service (MLaaS), where big technological companies provide artificial intelligence (AI) services to customers. One of the functions of the MLaaS platform allows customers to purchase an ML model on demand. In the foreseeable future, we think that the market of trading ML models is going to grow significantly, and the security and privacy of the trade and the ML models will be crucial [7].

Let us illustrate this with an example. Suppose a car company B is interested in adding autonomous driving technology to its self-driving cars. B would like to buy a computer vision model, a pre-trained ML model, from an AI technological company A. For instance, Mobileye, an advanced driver-assistance system (ADAS) provider, sells computer vision models installed on chips to automobile companies such as

L. Amorós
Aalto University, Espoo, Finland
e-mail: laia.amoros@aalto.fi

S. M. Hafiz
Indiana University-Bloomington, Bloomington, IN, USA
e-mail: shafiz@iu.edu

K. Lee (✉)
Seoul National University, Seoul, Republic of Korea
e-mail: activecondor@snu.ac.kr

M. C. Tol
Worcester Polytechnic Institute, Worcester, MA, USA
e-mail: mtol@wpi.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_11

Hyundai or Nissan. In this scenario, B must have the ML model available without the need to go online: the system might lose the internet connection by accident or have no connection at all.

In another similar situation, imagine that a pharmaceutical company A has an ML model trained on the medical profiles of its patients. An IoT company B builds medical devices for its end-users. At some point, the end-users want to know whether they are prone to a specific disease through the medical device. B has neither the model nor the training data to provide such a service to its end-users, and it probably cannot share their private data due to privacy issues. B cannot use homomorphic encryption (HE) or secure multi-party computation (MPC) as privacy-preserving ML solutions offered previously because of computational resource requirements of the ML task. Therefore, B might be interested in purchasing the ML model from A and install it on the medical devices, so that the end-users can check their medical status.

During the trade of ML models, some particular interactions between two parties, e.g., a seller company A and a buyer company B, need to be done before the final transaction. For instance, A wants to ensure that once the model is given to B, B will buy it. In addition, B needs to confirm that the model is valid and accurate for its purposes before purchasing it. We propose an HE-based protocol, which will secure the transaction for both parties, allowing A to sell its ML algorithm to B keeping the model secret, and at the same time protecting B from a possible *bad* ML model before buying it.

There are two benchmarks of ML models to be tested before a transaction: accuracy and efficiency. Whereas the timing result is independent of the input data (characteristic that ML models share with traditional software services), the accuracy of ML models depends excessively on training and test datasets. In this work we will focus on guaranteeing the accuracy of the ML models and leaving the efficiency issue on the background.

This report is organized as follows. In Sect. 2, we present the situation described above and propose some possible non-cryptographic solutions that turn out to be problematic. In Sect. 3, we recommend our HE-based solution and describe possible improvements to the protocol to make the overall transaction more efficient and secure. In Sect. 4, we start by discussing the feasibility of our HE-based solution, we then compare it to other potential existing cryptographic solutions, and we finish by considering a dual scenario: trading datasets instead of the ML models.

2 Non-cryptographic Approaches and Their Drawbacks

In this section, we describe some non-cryptographic approaches (and their associated problems) to enable a company A to sell a pre-trained machine learning model to a customer B. The situation is as follows: A is a company that owns a pre-trained ML model that a costumer B would like to purchase. In this simple situation, one can think of (at least) three different approaches that do not make use of any cryptographic measures, and the consequent problems that might arise.

Approach I: The first approach is simple, but naive: B makes the payment before getting any service from A. After the payment, A sends the pre-trained model to B. The potential problem in this situation is that B has no idea about the model quality before the payment, as training and test accuracies or generalization ability cannot be tested beforehand. The model may not be powerful enough to deploy on the client's side due to poor hyperparameters or model selection. A may not even send a trained model at all.

Approach II: In the second approach, when a client B wants to try the model, A sends the pre-trained model to B. If the model fits well on B's test dataset, B can make the payment for the model. The problem arising here is that B can deploy the model without making any payment, claiming that the model does not fit its needs, but keeping the model.

Approach III: The third approach provides a test session before the contract to try to avoid the previous situations. Before any compromise between A and B, B sends a test dataset to A. A evaluates the trained model using B's dataset and sends the results back. If the results are satisfying enough for B, the payment is made to A. The problem here is that B cannot know whether the predicted results are obtained from a well-trained ML model or, for instance, by hand. For example, a computer vision task may be manually achieved by crowdsourcing.

3 Our HE-Based Cryptographic Solution

In this section, we propose a potential solution for trading ML models that overcomes all the problems stated before. This solution is based on HE. After presenting the protocol, we continue by describing possible improvements to make the protocol more efficient. We then demonstrate some possible attacks to our protocol and suggest possible defenses. Finally, we summarize which ML model would be compatible with the proposed protocol while satisfying suitable efficiency and security.

3.1 The Protocol

The steps of the proposed protocol in Fig. 1 are as follows.

1. First A runs `Setup` to obtain the public HE parameters (base cyclotomic ring, modulus, number of levels, etc.). Note that B cannot run `Setup` if we use SHE

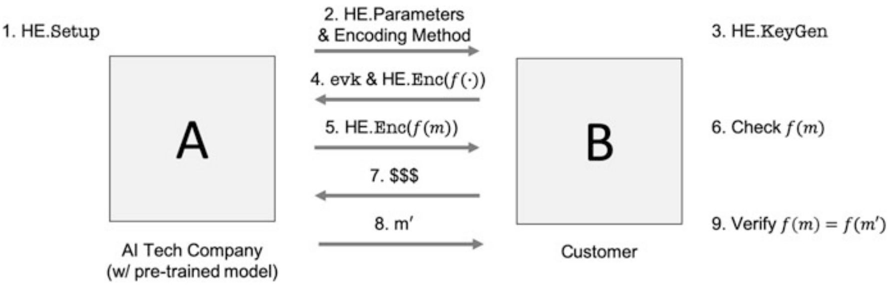


Fig. 1 Our HE-based solution

since B does not know the model and the required depth accordingly. If we use FHE, B can run `Setup`.[1]

2. A sends the HE parameters to B. A also sends a data encoding method to B. Agreement on the encoding method is essential since the encoding method considerably affects the performance, and A cannot manipulate the received ciphertexts easily.
3. B runs `KeyGen` after checking that the security level of the received HE parameters is sufficient.
4. B sends the test data m in encrypted form, i.e., $\texttt{Enc}(m)$, with evaluation keys `evk`.
5. A performs homomorphic inference on $\texttt{Enc}(m)$ using the evaluation keys, i.e., computes $\texttt{Enc}(f(m))$, and sends it back to B.
6. B decrypts the received ciphertext and gets $f(m)$, and checks if the algorithm works as expected.
7. If B decides to buy the model $f(\cdot)$, B proceeds to do the payment to A.
8. A sends the model $f'(\cdot)$ to B.
9. B checks if $f(m)$ is equal to $f'(m)$. If not, B now has the evidence that A did not send the proper model.

Our HE-based protocol shows substantial advantages over the other non-cryptographic approaches presented in Sect. 2. As the protocol is built on *homomorphic* encryption, B can have a test session without even knowing A's model before the contract. The *security* of the HE scheme ensures that A cannot cheat during the test session as in the non-cryptographic Approach III, because A does not see the test data m. The received message $f(m)$ in step 6 ensures the *commitment* by allowing B to notice in step 9 if A cheats by not sending a proper model in step 8.

[1] SHE stands for *somewhat* homomorphic encryption. FHE stands for *fully* homomorphic encryption.

3.2 Efficiency of the Protocol

In general, the main drawback of HE in practical usage is its slow speed. One attractive feature of our solution as an application of HE is that, in our setting, latency issues are more acceptable, since no real-time computation is necessary. Depending on the importance of the ML model transaction, a seller and a buyer may take their time for securing the trade. However, requiring an excessive amount of inference time would cause inconvenience. In this section, we discuss possible optimizations for our solution.

Optimization methods for general HE applications are also applicable to our solution. For example, a buyer can batch several test data into a ciphertext, and then the seller can evaluate the ML model only once for multiple test data. One would probably want to use the CKKS scheme for approximate homomorphic computations, and approximate the ML model into an HE-friendly version, i.e., into multivariate polynomials. Since these may cause other errors to the output, one should carefully consider this trade-off between accuracy and efficiency. In this case, the buyer should keep in mind that a new glitch has occurred, and the original model will be slightly more precise than the homomorphically evaluated results. However, since many ML algorithms are robust to errors in general, this might not cause a big problem.

Another possible optimization is to use HE parameters with a reduced security level, rather than HE parameters with conventional security level (e.g., 128-bit security). Since we need the security of ciphertexts only until the agreement of the contract is made, we can use more efficient parameters.

3.3 Towards the Perfect Model Protection

Even though HE provides strong security in our scenario, there are still some remaining issues that need to be considered due to the nature of the ML model trading situation and the lack of circuit privacy in concurrent HE schemes. In this section, we consider possible attacks and suggest defenses for these attacks.

To begin with, a malicious buyer may perform numerous test queries to obtain nontrivial information about the ML model (e.g., a model extraction attack [8]). To prevent a model extraction attack, the seller must limit the number of trials that a buyer can query. This can be done both explicitly and implicitly. Explicitly, a seller can regulate the amount of queries. Implicitly, a seller can require costs to a buyer for each query. This cost works as a client puzzle: for a small number of queries, the prices are negligible; but for a large number of queries to perform the model extraction attack, the costs are infeasible to pay.

Another natural concern on the ML model trading scenario is malicious redistribution. That is, once an ML model is sold, the buyer can illegally resell or redistribute it. In this respect, a possible solution is to use *neural network watermarking* [1, 4].

A third concern is the circuit privacy of HE schemes: since concurrent HE schemes do not provide circuit privacy, there exists a possibility of information leakage of ML models from the output of homomorphic computations. However, there are well-known countermeasures such as noise-flooding or bootstrapping [2].

3.4 Compatible ML Models

In terms of security and efficiency, the following properties of an ML model need to be taken into account in order to work with the proposed protocol.

Criteria 1: The ML model needs to be complex enough so that it can resist a model extraction attack.

Criteria 2: The homomorphically encrypted ML model should have plausible inference time to evaluate the test dataset.

For binary and multi-class logistic regression models, the size of the feature vector should be much greater than $k + 1$ and $c\,(k + 1)$ respectively, where k is the number of allowed queries and c is the number of classes. Neural networks like DNN, CNN, RNN, etc., should have network parameters much larger than k (e.g., more than $100\,k$ regarding the experimental results of [8]).

4 Discussions

In this section, we first discuss the plausibility of our trading ML models scenario. We then consider possible alternative cryptographic solutions to the problem like secure multiparty computation or zero-knowledge arguments. We end the section by considering a dual situation: trading datasets instead of ML models.

4.1 Plausibility of Trading ML Models

Machine learning-based solutions have become a rather conventional way of tackling many problems. However, training ML models requires a substantial amount of computational power and a vast amount of training data. Therefore, selling such trained ML models can be seen as a rosy business model, since MLaaS market is growing fast. Many companies in this area propose to train and tune the models of a customer at a reasonable price (compared to the cost of computing resources and massive datasets required if the customer were to train the model by itself).

One may ask why the company should sell the model itself instead of providing online access and sell subscriptions. We are going to answer this question from various perspectives. First of all, there are several ML applications where the service needs to work also when the system goes offline (e.g., driverless car). Another disadvantage is the privacy issue. A customer has to send always its private data to the service provider if this only offers a subscription. Of course, we can use HE or MPC to solve this privacy issue via PPML. However, this PPML approach suffers from slow speed and there are some ML applications where such latency cannot be tolerated (e.g., driverless car).

Another point is that a customer might want to train the model further to improve its performance for new data points using online learning, or the model can be used in multitask learning [3]. Without trading the model itself, this would not be possible for a customer. We also note that trading ML models is a one-time sale, but the company can retrain the model with up-to-date datasets after a while and sell updates.

4.2 Alternative Cryptographic Solutions

Secure multiparty computations (MPC) can be used as an alternative solution to our HE-based approach for the safe ML model trading scenario. When using MPC, one can hide the model weights, but it is a non-trivial task to protect the whole ML model, including its hyperparameters. That is, a seller and a buyer would share the ML model structure and perform secure multiparty computations on it, without revealing the model weights and the test data to each other. This leads to the solution being more susceptible to model extraction attacks. Moreover, the selection of suitable neural network architectures for a given problem and dataset is often difficult and resource-consuming [5], meaning that sharing the structure of the ML model might already be a substantial loss of intellectual property for the seller.

Another issue for an MPC-based solution is communication overhead. The communication cost between the buyer and the seller becomes more significant as the ML model becomes more complex, and the two parties should remain online during the computation. This might be a minor problem if two large companies do the transaction, but might be an essential issue if a customer is an individual with low computing power.

One might also consider zero-knowledge arguments (e.g., zk-SNARK) for another possible solution to the model trading scenario. In particular, when test data is given, the seller can use zk-SNARK to prove that it has the ML model, which outputs one specific result without revealing the parameters (zero-knowledge). However, in this case, the buyer has to disclose its test data to the seller, leading to the same problems as the non-cryptographic Approach III described in Sect. 2.

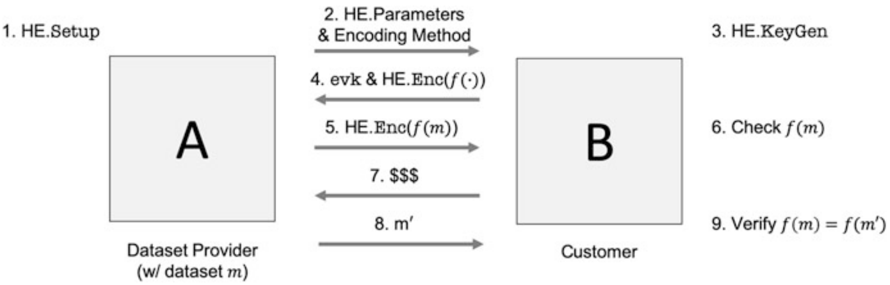


Fig. 2 Our HE-based solution for the dual scenario

4.3 Dual Scenario: Trading Datasets

One interesting point of view about our solution is that one can also consider its *dual* scenario. In the original situation, the seller has a function (e.g., a ML model), and the buyer wants to check the output of the function on particular data before the transaction. On the other hand, in the dual scenario, the seller has a dataset, and the buyer wants to check the output of a particular function on the dataset before the transaction. Thus, the roles of functions and data are switched. We can directly apply our protocol, which is designed for the original scenario, to the dual situation by just interchanging the roles of function and data. Our HE-based solution for the dual scenario is summarized in Fig. 2.

However, in this case, our solution seems to have a less significant advantage over an existing MPC-based solution [6]. This is because, unlike the original solution, an ML model is being sent over a homomorphically encrypted channel. In this scenario, we probably want to encrypt only the model weights and biases and send the model hyperparameters unencrypted. This invalidates the first advantage of the HE-based solution over the MPC-based solution, which is described above. If not, we can homomorphically encrypt the whole circuit description. However, this approach does not look practically sound again.

References

1. Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 1615–1631, USA, 2018. USENIX Association.
2. Florian Bourse, Rafaël Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In *Proceedings, Part II, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9815*, pages 62–89, Berlin, Heidelberg, 2016. Springer-Verlag.
3. Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

4. Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS'19, pages 485–497, New York, NY, USA, 2019. Association for Computing Machinery.
5. Debadeepta Dey. Microsoft research blog: Project petridish: Efficient forward neural architecture search, 2019.
6. Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Peter Rindal, and Mike Rosulek. Secure data exchange: A marketplace in the cloud. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, CCSW'19, pages 117–128, New York, NY, USA, 2019. Association for Computing Machinery.
7. Technavio. Global machine learning-as-a-service (MLAAS) market 2019–2023, 2019.
8. Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIS. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 601–618, USA, 2016. USENIX Association.

HEalth: Privately Computing on Shared Healthcare Data

Leo de Castro, Erin Hales, and Mimee Xu

1 Introduction and Motivation

Healthcare in the US is notoriously expensive. Compared to other Organisation for Economic Co-operation and Development (OECD) countries, US healthcare costs are one-third higher or more relative to GDP [6]. According to the Centre for Disease Control and Prevention [5], the average per capita cost of healthcare was $10,739 for the year 2017. Several Machine Learning (ML) startups aim to improve healthcare by bringing automated expertise to hospitals, in areas such as brain imaging and cancer detection. Additionally, hospitals in the US cannot easily share data. The existing ML solutions often focus on training a model using data which has been obtained through an existing collaboration, which has been pre-processed to the required format. These solutions work around the data-sharing challenge for training rather than tackling it and The result is a static inference model which does not continuously adapt to changes.

It is important to maintain ethical standards for healthcare professionals. This is one of the goals of regulatory agencies working to protect patients. Such agencies can often have problems accessing appropriate data to compare between hospitals and doctors, as well as to assess data across many hospitals. This can lead to difficulties in fulfilling a mandate to audit doctors and hospitals. There are other

L. de Castro
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: ldecastr@mit.edu

E. Hales (✉)
Information Security Group, Royal Holloway, University of London, Egham, Surrey, UK
e-mail: peai011@live.rhul.ac.uk

M. Xu
Courant Institute of Mathematics, New York University, New York, NY, USA

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_12

Table 1 Summary data for all US hospital admissions in the year 2018, from [2]

Number of hospitals	Staffed beds	Total Admissions	Total expenses
6146	924,107	36,353,946	$1,112,207,387,000

healthcare use cases for which assessing data across many hospitals is useful. Data sharing can be beneficial for managing epidemics and treating rare diseases.

Since the data gathered by healthcare providers is often sensitive, it is essential to protect it while in use by a ML algorithm. Homomorphic encryption can be brought to bear on these situations by allowing meaningful computation on encrypted data. This enables secure data sharing and computation, as well as private training.

The initial goal of our work, in the context of healthcare data, is to allow anomaly detection and discovery in a shared private data-set. To do this we propose calculating aggregate statistics across data shared between many hospitals to consider 'fairness' of hospital admissions.

First, let us consider hospital admissions across the US. Some summary data is provided in Table 1. With such a large number of admissions across the year, there is a lot of data generated. If this data could be shared securely across several hospitals, or even all hospitals, it would enable secure computation on a scale not seen before.

We present a scenario where hospitals share records to compute anomalies and audit fairness. We make three contributions. Firstly, the possibility of auditing fairness at scale. Secondly, enabling continuous sharing of data without the need to refresh keys, removing the requirement for pairwise contracts. Finally, we discuss how anomaly detection might be applied to finding causes and correlations in medical research. This has many applications, for example rare diseases, epidemic management and chronic illness research.

We use 'fairness' as an initial calculation goal which would demonstrate the effectiveness of simple statistics. It would then be possible to extend the techniques to other calculations, achieving different goals.

2 Our Scenario

Suppose we have a group of hospitals who wish to combine their data to calculate aggregate statistics. Each hospital has an interest in keeping their data private from the other hospitals, so we introduce a method whereby the hospitals can work together to create a shared secret key to encrypt the shared data on which computation takes place. Once private computation has taken place, the hospitals are able to collectively decrypt the results using their shares of the decryption key.

The initial goal of calculating these aggregate statistics is to compute anomalies and audit fairness in admissions statistics. This example illustrates a situation which could bring a benefit to patients. The same techniques could be applied to other scenarios where data is shared between several parties and private computation takes place on the data.

We now consider the motivation of hospitals to take part in this process. Firstly, this data sharing process will allow hospitals to have access to statistics calculated using far more data than they would have access to if working independently. Additionally, hospitals will be able to compare themselves to others, obtaining more data for research and improvement.

A potential deployment issue is that different hospitals may store their data differently. So there may be some requirement for participating hospitals to share their data in a particular format. For example, we may need to format the data gathered in the format of Fig. 1 as a matrix or a table. It is possible for this data pre-processing to be automated from the existing form. Efficiency could be improved by gathering data in a uniform way across participating hospitals to remove the need for pre-processing.

3 A Discussion of the Underlying Cryptography

We propose to use the method of multiparty communication for threshold FHE, as introduced in [1]. Asharov et al.'s work is an extension of existing FHE schemes [4, 7]. In the multiparty communication setting, Key Generation and Decryption become N-party protocols. Since we have a threshold encryption scheme, we require N out of the M parties to cooperate in order to encrypt and decrypt.

Each of the M participating hospitals would be a party of the threshold scheme, and we would require N hospitals to cooperate in order to decrypt the results of the computations.

We must consider the case where a hospital no longer wishes to participate. In this case, the hospital can destroy their share of the key, and other parties will still be able to decrypt the results. The data of the non-participating hospital will remain in the dataset until that round of computations on the data is complete. It will then be possible to remove the data of one of the hospitals and begin computations again with fresh keys on a new set of data.

4 The Initial Goal: Fairness

Our initial goal is to compute statistics based on fairness. Here it is possible to compute relatively simple statistics on encrypted data which will have a large impact on patients and their care.

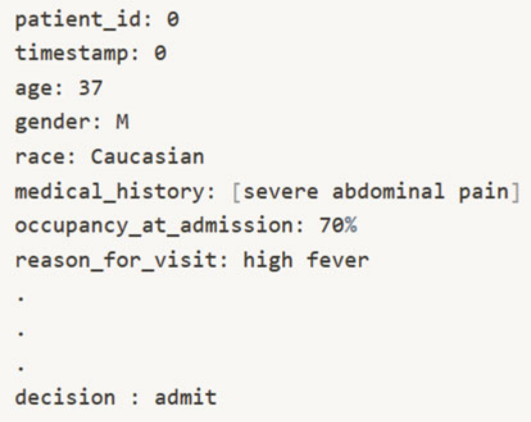
We will measure fairness within the context of protected characteristics, since these are recorded by hospitals already. Using these characteristics as indicators will allow complex statistical information to be summarised and made accessible to a non-technical audience [11].

We have many competing frameworks available, coming from the different protected characteristics and policy domains, it is challenging to conceptualise fairness. For example, what weight do we give to different 'strands' of protected characteristics when we evaluate fairness? We aim to consider equality as "an outcome of equal treatment" [11]. In the context of medical data, the concept of 'treatment' takes on additional significance.

We conclude that while fairness may vary between different frameworks, if an admission decision lies too far from the norm then it is more likely to be due to unfair practices rather than noise. The volume of data that our method allows anomalies to be identified more easily.

We will require data submitted by hospitals to be in the same form, for example in Fig. 1. We must consider that the features we choose to record should encapsulate what the doctor is seeing and the information they use to make their decisions, as well as being similar to what a doctor would previously have recorded. We can ensure the data submitted by hospitals takes the same format without pre-processing by adapting the interface the data is entered into.

We wish to calculate density-based statistics, and to begin with we calculate simple statistics such as averages and histograms. In our work so far using histograms we have only calculated one dimensional metrics, but these simple implementations highlight the potential for managing higher dimensional data in a usable manner. In addition, when compared to regression based methods our density-based statistics have a much weaker selection bias since averaging controls for bias.



```
patient_id: 0
timestamp: 0
age: 37
gender: M
race: Caucasian
medical_history: [severe abdominal pain]
occupancy_at_admission: 70%
reason_for_visit: high fever
.
.
.
decision : admit
```

Fig. 1 Example admissions data

5 Discussion

This application of threshold HE allows us to calculate healthcare statistics at a scale not seen before. The results of our calculations aim to be readily interpretable, with adaptive decision making rather than a blind prediction or classification. This is a novel use case for the existing research taking place in Threshold FHE [1, 3, 9, 10]. Our application will hopefully be able to leverage developments and optimisations in the field of threshold encryption.

In terms of encrypting and sharing the data, the key benefit of our approach is that it allows hospitals to share data securely. Participating hospitals are provided an assurance that the data cannot be decrypted without hospitals cooperating.

Hospitals have a strong privacy incentive to engage with the calculations, since uploaded data is encrypted and cannot be decrypted by one of the hospitals. This is because we require participation of many parties to decrypt, since each hospital only has access to their own share of the decryption key.

Another particular benefit of our approach is that as long as the hospitals retain their keys, additional algorithms could be developed to compute collective statistics, including implementing additional algorithms [8] on historical data without revealing secrets.

As it stands, our scheme does not consider malicious users. Future work would analyse how the system would behave if users were malicious or honest but curious, and how such malicious users could work together to compromise security. Additionally, future work could explore what would happen if hospitals were to submit some false or corrupted data, or an entirely false dataset. It would be useful to see how much false or corrupted data the system could tolerate.

Future work could also extend the existing 'fairness' use case to other desirable statistics on medical data.

References

1. Asharov G, Jain A, López-Alt A, Tromer E, Vaikuntanathan V, Wichs D (2012) Multiparty computation with low communication, computation and interaction via threshold FHE. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp 483–501
2. Association AH ((accessed January 14, 2020)) Fast Facts on U.S. Hospitals, 2020. https://www.aha.org/statistics/fast-facts-us-hospitals
3. Boneh D, Gennaro R, Goldfeder S, Jain A, Kim S, Rasmussen PM, Sahai A (2018) Threshold cryptosystems from threshold fully homomorphic encryption. In: Annual International Cryptology Conference, Springer, pp 565–596
4. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) 6(3):13
5. for Disease Control C, Prevention (2017 (accessed January 14, 2020)) Health Expenditures. https://www.cdc.gov/nchs/fastats/health-expenditures.htm
6. for Economic Co-operation O, Development (2020 (accessed January 14, 2020)) Health expenditure and financing. https://stats.oecd.org/Index.aspx?DataSetCode=SHA

7. Fan J, Vercauteren F (2012) Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012:144
8. Graham S, Estrin D, Horvitz E, Kohane I, Mynatt E, Sim I (2011) Information technology research challenges for healthcare: From discovery to delivery. ACM SIGHIT Record 1(1):4–9
9. Jain A, Rasmussen PM, Sahai A (2017) Threshold fully homomorphic encryption. IACR Cryptology ePrint Archive 2017:257
10. Schoenmakers B (2011) Threshold Homomorphic Cryptosystems, Springer US, Boston, MA, pp 1293–1294. https://doi.org/10.1007/978-1-4419-5906-5_13
11. Walby S, Armstrong J (2011) Developing key indicators of 'fairness': Competing frameworks, multiple strands and ten domains – an array of statistics. Social Policy and Society 10(2):205–218, https://doi.org/10.1017/S1474746410000552

Private Movie Recommendations for Children

Anh Pham, Mohammad Samragh, Sameer Wagh, and Emily Wenger

1 Introduction

Data-driven business models such as recommender systems (Netflix, Pandora) and targeted advertising (Facebook, Google) rely on consumer data and the information they contain on individuals' behavioral patterns and preferences. This reliance effectively opens door to the longstanding conflict of privacy versus convenience: as customers expect the rendered goods to be content-relevant to their specific needs, a certain degree of user data exploitation by service providers is mandatory. At the same time, the mounting number of data collection and data breaches has prompted the public to grow hostile towards the tech sector; a recent case is the $170 million fine imposed in September 2019 on Google and YouTube Kids for violating federal requirements for child privacy protection [1]. Homomorphic encryption offers a solution to this pressing problem: a fully homomorphic encryption (FHE) scheme can be used to construct a private recommender system with which user data is not exposed to service providers in the raw form, and only data "masked" by encryption is sent to providers for recommendation. In this project, we construct

A. Pham
Department of Biomedical Informatics, UC San Diego, La Jolla, CA, USA
e-mail: anp055@eng.ucsd.edu

M. Samragh
Electrical and Computer Engineering, UC San Diego, La Jolla, CA, USA
e-mail: msamragh@ucsd.edu

S. Wagh (✉)
RISE Lab, UC Berkeley, Berkeley, CA, USA
e-mail: swagh@alumni.princeton.edu

E. Wenger
Computer Science, University of Chicago, Chicago, IL, USA
e-mail: ewenger@uchicago.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_13

a general framework for an FHE-backed recommender that can be extended to different applications, with the particular use case of YouTube Kids as a proof-of-concept.

To this end, we propose a *private video recommendation system*, appropriate for a platform like YouTube Kids. This system, built on fully homomorphic encryption, would allow the platform to make tailored recommendations to children without exposing their private information. In this writeup, we briefly describe the motivation and construction of our system. We then discuss the novelty, soundness, feasibility, and impact of such a system.

1.1 Background

The Child Online Protection Privacy Act (COPPA) was passed in the US in 1999 and significantly revised in 2011 [2]. It provides legal protection for children's activity and data shared online. Among its many requirements, it mandates that online providers "…establish and maintain reasonable procedures to protect the confidentiality, security, and integrity of the personal information collected from children under age 13". The recent legal action against Google is a sober reminder that such privacy-protection procedures are often not in place. Nonetheless, the fact remains that a recommender system requires collecting of personal data.

FHE implementation within the recommender pipeline can resolve the tension between privacy preservation and pattern mining over personal data. FHE allows the ability to compute on encrypted data and yields results that, when decrypted, would match the computation as if it has been done on plaintexts. This capacity enables recommenders to execute their algorithms while respecting the privacy of users. Prior work on private recommendation algorithms informs our prototype of a private video recommendation system for children [3].

2 Proposed Implementation

The abstract design of our recommender system is shown in Fig. 1. In this setting, the server wishes to utilize the client's confidential feature vector $x \in \mathbb{R}^n$ and provide a proper recommendation. The protocol involves the following steps:

1. The client encrypts its private message $x \rightarrow \mathsf{Enc}_{\mathsf{pk}}(x)$.
2. The server receives the encrypted message and computes $f(\mathsf{Enc}_{\mathsf{pk}}(x))$ homomorphically. During this process, no information about input x or the analysis result is revealed to the server since only the client has the decryption keys.
3. The client decrypts $f(\mathsf{Enc}_{\mathsf{pk}}(x)) \rightarrow f(x)$ using her secret key and retrieves the recommendation.

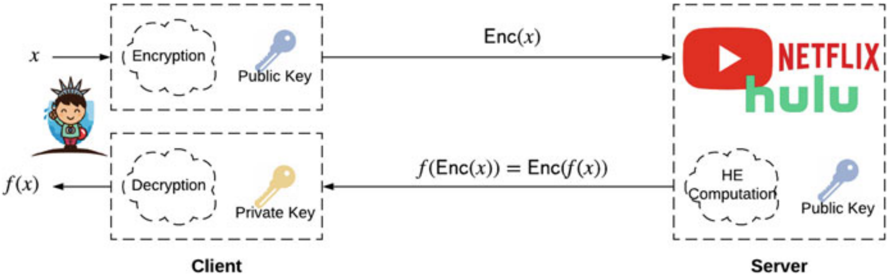


Fig. 1 Our proposed design for a private recommender system based on homomorphic encryption

In our prototype, the training of the recommender model is done on public ratings from n clients, i.e., $x_1, \dots, x_n$. The data matrix $X = [x_1|x_2 \dots |x_n]$ is formed by stacking the data from the users. The element located at the i-th row and j-th column, $X_{i,j}$, is the rating that the j-th user provides for the i-th movie. For recommendation, we utilize Content Based Filtering (CBF) and Collaborative Filtering (CF) which are widely adopted in recommendation systems. The key steps are as follows:

Offline/Training Phase The server computes a similarity matrix $S \in \mathbb{R}^{m\times m}$ which will later be used for making recommendation. Formally, the similarity matrix is computed as

$$S = \frac{X \cdot X^T}{W \cdot W^T} \tag{1}$$

where $X \in \mathbb{R}^{m\times n}$ is the data matrix and $W \in \mathbb{R}^m$ is the norm of ratings computed by the following equation:

$$W = \sqrt{\sum_{i=1}^{n}(x_i^2)} \tag{2}$$

For this phase of the prototype, we perform the computation of S in plain text. However, this step can be done homomorphically to enhance privacy.

Online/Inference Phase Depending on the underlying data, the server either uses CBF or CF for recommendation making. For a feature vector x, the server computes $f(x)$ as follows:

$$f(x) = \begin{cases} \frac{S\cdot x}{Y} & \text{CBF} \\ A - \frac{S\cdot A}{Y} + \frac{S\cdot x}{Y} & \text{CF} \end{cases} \tag{3}$$

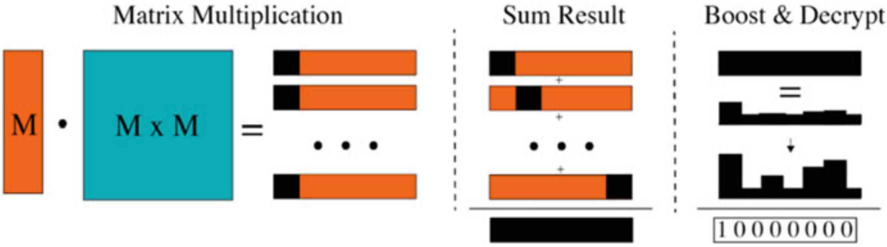


Fig. 2 Our proposed design for a private recommender system based on homomorphic encryption. The underlying algorithm is described in Sect. 2.1. We use boosting to increase the weight of the "high" recommendations and reduce that of "low" there should be a period after recommendations

where $Y \in \mathbb{R}^m$ is achieved by summing the columns of S and $A \in \mathbb{R}^m$ is the average rating for each movie. Note that the term $A - \frac{S \cdot A}{Y}$ in the CF method is a constant and can be computed offline. In both CF and CBF methods, the core computation involving users' data is $\frac{S \cdot x}{Y}$. To ensure user privacy, this step should be done homomorphically. Figure 2 summarizes the required HE-based operations. In the next section, we elaborate on the design of the homomorphic encryption portion of our application.

2.1 HE Technical Details

We choose the non-interactive model to be fairly optimal in terms of communication. The total communication is the size of two ciphertexts which is $2 \cdot \log q \cdot N \approx$ 364 KB each direction, where q is the ciphertext modulus and N is the degree of the cyclotomic polynomial. In terms of computation, the server performs matrix-vector multiplication with the vector encrypted and the matrix in plaintext. This requires Galois keys from the user and consumes one depth of computation. Further, to compress all the products into one ciphertext, we use a sequential masking (all slots encode the dot product so no further rotation required) and then add these up into one ciphertext. This consumes one depth of computation.

The boosting is a simple observation that for a given vector $\{a_1, a_2, \ldots, a_n\}$, computing the vectors $\{a_1^k, \ldots a_n^k\}$ for a given k will increase the separation between the top values of a_i from the rest. In our work, we set $k = 2$ and hence require 2 depths of HE computations. The performance bottleneck is the matrix-vector multiplication which we parallelize using eight cores. Overall, we require parameters to be set that support at least depth 4 computation. This puts a lower bound on the degree of the cyclotomic polynomial N. We need to use at least $N = 8192$. This gives us about 218-bits of space for the ciphertext modulus. For performance reasons, we use these values and enable computations over a movie corpus of up to 2048 movies. The keys are stored locally on the end devices and the Galois keys are communicated in an offline/set-up phase.

3 Discussion

In this section, we evaluate our solution under the proposed comparison metrics.

Novelty While previous work has designed a private recommendation system (see [3]), current recommender systems do not incorporate privacy-preserving computations on user data. Our work designs and tests a *practical* and low-overhead implementation of such a system. Furthermore, the implementation we propose can extend to domains beyond video recommendation. This opens the door to a wide variety of new applications at the nexus of homomorphic encryption and machine learning.

Soundness The security of our system is inherent due to the underlying homomorphic encryption scheme. The polynomial degree and ciphertext modulus are set to prevent information leakage, as discussed in Sect. 2.1. In this short article we only discuss non-interactive HE-based scenarios where the functionality of the recommender system is solely rendered by linear operations. Designing interactive protocols that support more nonlinear operations is a promising future direction. In fact, it may be reasonable to assume an interactive user for video recommendation systems since the user is present during the process. Such interactive protocols will allow for private evaluation of more complex machine learning models, e.g., deep neural networks, which can provide a higher quality of service.

Feasibility Section 2.1 describes the technical details of our implementation. The size of the recommendation matrix scales linearly with the number of users. The size of the matrix may become unwieldy with very large numbers of users, so we recommend optimization of this matrix computation as future work.

Impact Our work creates alternative ways for companies like YouTube to recommend content to users without violating their privacy. As data-sharing scandals continue to surface, privacy-centric systems become increasingly important.

References

1. “Google and YouTube Will Pay Record $170 Million for Alleged Violations of Children’s Privacy Law”, FTC, 2019. https://www.ftc.gov/news-events/press-releases/2019/09/google-youtube-will-pay-record-170-million-alleged-violations
2. “Children’s Online Privacy Protection Act”, 2019. https://en.wikipedia.org/wiki/Children’s_Online_Privacy_Protection_Act
3. “A practical privacy-preserving recommender system”, Shahriar Badsha and Xun Yi and Ibrahim Khalil, 2016. *Data Science and Engineering.* Springer, Volume 1, number 3, pp 161–177.

Privacy-Preserving Prescription Drug Management Using Fully Homomorphic Encryption

Aria Shahverdi, Ni Trieu, Chenkai Weng, and William Youmans

1 Introduction

According to the CDC [8], 46 people die every day from overdoses involving prescription opioids. In an attempt to reduce abuse of controlled medications like opioids many states implement a Prescription Drug Management Program (PDMP). The program is realized as a central database accessible to healthcare providers who can query for a record of a patient's most recently dispensed controlled medications. This allows providers to make more intelligent decisions about when to prescribe or dispense these medications. It also helps prevent doctor shopping, where a patient sees multiple doctors for the same condition to obtain more controlled medications, either with the intention of abusing or redistributing the medication.

Currently, patient data in this system can be accessed by pharmacists, physicians, insurance companies, law enforcement agencies, and others. After registering with the program users are given access to all patient records. Granting access to such sensitive records to so many parties is an obvious security concern, and it comes as no surprise that a breach has already occurred. In Florida in 2013 [10], over

A. Shahverdi
University of Maryland, College Park, MD, USA
e-mail: ariash@umd.edu

N. Trieu
Arizona State University, Tempe, AZ, USA
e-mail: nitrieu@asu.edu

C. Weng
Northwestern University, Evanston, IL, USA
e-mail: ckweng@u.northwestern.edu

W. Youmans (✉)
University of South Florida, Tampa, FL, USA
e-mail: wyoumans@usf.edu

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_14

3000 patient records were shared with county prosecutors as part of a criminal investigation, when many were irrelevant to the case. An attorney involved in the investigation discovered a friend on the list and gave them the data so they could pursue legal action.

We propose a solution utilizing Privacy-Preserving Machine Learning (PPML) and Fully Homomorphic Encryption (FHE) to preserve the benefits of the PDMP database while eliminating the risk of leaking sensitive patient information. We accomplish this by transferring control of the data back to the patient. Patient records will be stored in a central server and encrypted using the patient's key. Prescribers or pharmacists can submit a patient's encrypted prescription requests to this central server. Here, thanks to advances in FHE, we can perform PPML on the patient's encrypted data to produce an encrypted certificate authorizing or denying the prescribing or dispensing of the medication based on their history. Once decrypted, this certificate can be authenticated by the healthcare provider to prevent potential tampering or counterfeiting. Finally, the server can update the patient's encrypted record without the need for decryption by using the properties of FHE. This protocol will prevent the use of the database for any other purpose than determining if a patient is eligible for a controlled medication.

2 Our Model

For simplicity, we will describe a scheme which only considers three parties: the patient, pharmacist, and server. We assume the server holds a database of patient records—each encrypted under a separate key—as well as a machine learning model trained to detect potential abuse of medication. In practice, other parties might want to be involved, such as prescribers that want to ensure a patient is not potentially abusing their medication. The scheme described below can easily be extended to account for this scenario.

Since we desire that each patient encrypts their data with their own key, we need a method of ensuring that encrypted responses from the server decrypted by the patient and delivered to the pharmacy have not been tampered with. We propose that the server and pharmacy create a shared secret key for some symmetric scheme that can be efficiently evaluated homomorphically for our choice of FHE scheme. The details of the FHE scheme, PPML model, and symmetric scheme for authentication will be discussed later.

The overview of our scheme is presented in Fig. 1. It consists of two phases: (1) submitting a prescription request, where a PPML model determines if the medication is safe to dispense, and (2) authentication of the result in the pharmacy. The first phase can be done entirely between the patient and the server, see Fig. 2. Consider an example where a patient has a prescription for a controlled medication. They send to the server the hash of their identity $H(x)$ as well as the encryption $\textsf{\textbf{Enc}}_{pk}(P)$ of their prescription details (either scanned or manually entered, for example). The server evaluates the PPML model on the patient's existing encrypted

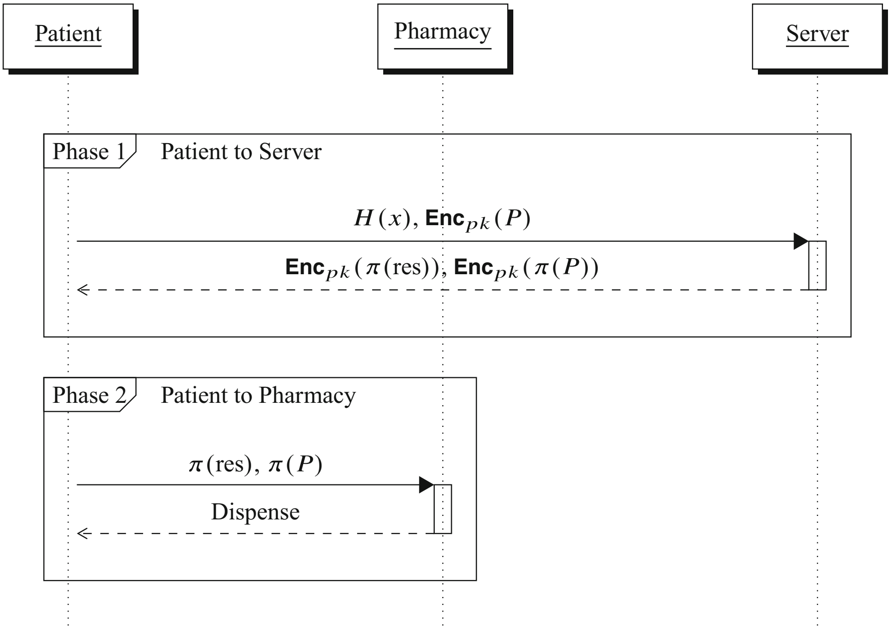

Patient
Pharmacy
Server
Phase 1
Patient to Server
$H(x)$, $\mathsf{Enc}_{pk}(P)$
$\mathsf{Enc}_{pk}(\pi(\text{res}))$, $\mathsf{Enc}_{pk}(\pi(P))$
Phase 2
Patient to Pharmacy
$\pi(\text{res})$, $\pi(P)$
Dispense

Fig. 1 The overview of our scheme

INPUT: Hashed patient identifier $H(x)$, encrypted patient prescription $\mathsf{Enc}_{pk}(P)$.

- Locate the patient's encrypted record $\mathsf{Enc}_{pk}(R)$ using their identifier $H(x)$.
- Obtain the output $\mathsf{Enc}_{pk}(\text{res})$ of the PPML model with input $\mathsf{Enc}_{pk}(R)$ and $\mathsf{Enc}_{pk}(P)$.
- Homomorphically compute $\mathsf{Enc}_{pk}(\pi(\text{res}))$, $\mathsf{Enc}_{pk}(\pi(P))$ for a symmetric scheme π.

OUTPUT: Send to the patient $\mathsf{Enc}_{pk}(\pi(\text{res}))$, $\mathsf{Enc}_{pk}(\pi(P))$.

Fig. 2 Patient-server interaction in phase 1

record $\mathsf{Enc}_{pk}(R)$ and new encrypted prescription to produce an encrypted result $\mathsf{Enc}_{pk}(\text{res})$ authorizing or denying the new prescription based on the patient's history. We will refer to this result of the model as the "label".

In the second phase, the patient decrypts the server response to obtain $\pi(\text{res})$ and $\pi(P)$ which they provide to the pharmacist. Optionally, the server can return $\mathsf{Enc}_{pk}(\text{res})$ as well, if it's desired that the patient sees the label at this step. The pharmacist decrypts $\pi(\text{res})$ and $\pi(P)$ with the shared secret key to learn the result of the model as well as verify that the prescription data P sent to the server matches the prescription brought to them by the patient. We discuss how to prevent tampering or a dishonest patient from providing an incorrect or previously used label in Sect. 5.

Lastly, the server updates the patient's encrypted record with the new encrypted prescription data. We can assume the new prescription data is stored from the initial round of communication or sent again by the pharmacy. In either case, the technique for merging encrypted records is covered in Sect. 3.

3 Fully Homomorphic Encryption

There are three major aspects of our design which influence our choice of FHE scheme:

1. The server must be able to make predictions on a patient's encrypted data.
2. Healthcare providers will need a method of authenticating the server response to prevent tampering by the patient.
3. Upon dispensing a medication the patient's record will need to be updated.

3.1 Our Choice of FHE Scheme

In order to accommodate all of these needs we chose to use the Brakerski-Gentry-Vaikuntanathan (BGV) FHE scheme as described in [2]. While other schemes can handle requirements 1 and 3, the authentication step mentioned in requirement 2 means we will need an FHE-friendly symmetric encryption scheme. LowMC [1] is a block cipher with comparatively low multiplicative depth designed for use with FHE and multi-party computation. Lattice-based symmetric encryption schemes based on Learning With Errors (LWE), Learning Parity with Noise (LPN), and Learning With Rounding (LWR) were described in [5] and provide another option. All of these schemes were demonstrated to be efficient to evaluate homomorphically using BGV as implemented in HElib [7]. The details of this will be discussed in Sect. 5.

The next most important requirement is that we can homomorphically evaluate the machine learning model. For the most accurate models, this means approximating non-linear functions in a manner suitable to homomorphic operation. We give a more detailed survey of the potential solutions in Sect. 4.

3.2 Updating the Encrypted Records

The last usage of our FHE scheme is updating patient records. For this, we propose taking advantage of the ciphertext packing techniques of Smart and Vercauteren [9] which allow for single instruction multiple data (SIMD) operations. With this we can perform operations on a single ciphertext which translates to performing parallel operations coefficient-wise on a vector of plaintext slots. We also use homomorphic rotations of ciphertexts which corresponds to rotating the underlying plaintext slots. Both techniques are available in the current implementation of BGV in HElib.

Consider a simplified example where a patient prescription contains only the drug name, quantity, and date. We represent FDA approved drugs by their NDC or National Drug Code. We will write a patient's plaintext prescription as a vector $P = \langle n, q, d, 0, \dots, 0\rangle$ of length m determined by the number of plaintext slots available with the given BGV parameters. Let n, q, and d represent the NDC, quantity, and date respectively. We will represent a patient record containing the NDC, quantity, and date of the last m prescriptions as a triple (R_1, R_2, R_3) where $R_1 = \langle n_1, \dots, n_m\rangle$, $R_2 = \langle q_1, \dots, q_m\rangle$, and $R_3 = \langle d_1, \dots, d_m\rangle$. We will assume the records are in order from newest to oldest. A prescription encrypted under a patient's public key pk takes the form $\mathbf{Enc}_{pk}(P) = \mathbf{Enc}_{pk}(n, q, d, 0, \dots, 0)$, and an encrypted patient record $\mathbf{Enc}_{pk}(R) = (\mathbf{Enc}_{pk}(R_1), \mathbf{Enc}_{pk}(R_2), \mathbf{Enc}_{pk}(R_3))$. Lastly write $\mathbf{0}_i$ for the plaintext vector which is 0 everywhere and 1 in the i-th slot, and $\mathbf{1}_i$ for the plaintext vector which is 1 everywhere and 0 in the i-th slot. In Algorithm 1 we demonstrate a possible approach to updating encrypted patient records by rotating the ciphertexts and overwriting the m-th prescriptions data.

Algorithm 1 Update database

```
Input: encrypted patient prescription $\mathbf{Enc}_{pk}(P)$, encrypted patient record $\mathbf{Enc}_{pk}(R)$.
Output: updated encrypted patient record $\mathbf{Enc}_{pk}(R')$.
for $i \in \{1, 2, 3\}$ do
    $A \leftarrow \mathbf{Enc}_{pk}(P * \mathbf{0}_i) = \mathbf{Enc}_{pk}(P) * \mathbf{Enc}_{pk}(\mathbf{0}_i)$      ▶ Erase all but the $i$-th slot of $P$.
    $A \leftarrow \mathrm{rot}(A, m - i + 1)$                                   ▶ Rotate slot $i$ to slot 1.
    $B \leftarrow \mathbf{Enc}_{pk}(R_i * \mathbf{1}_m) = \mathbf{Enc}_{pk}(R_i) * \mathbf{Enc}_{pk}(\mathbf{1}_m)$    ▶ Erase slot $m$ of $R_i$.
    $B \leftarrow \mathrm{rot}(B, 1)$                                         ▶ Rotate slot $m$ to slot 1.
    $x_i \leftarrow \mathbf{Enc}_{pk}(R'_i) = A + B$                            ▶ Insert the prescription data.
end for
return $(x_1, x_2, x_3)$
```

3.3 Parameters

Choosing parameters for use with any FHE scheme is a delicate task involving many factors, often reducing to experimentation for fine tuning. Since we currently lack an implementation determining the parameters is even more challenging. However, we note that BGV supports switching back and forth between plaintext moduli of the form 2^k via a re-encryption process.

We outline a general approach inspired by Crawford et al. [4]. By storing our data (prescriptions and prescription records) in packed ciphertexts with plaintext modulus 2^k for some large enough k, we can extract the k encrypted bits as necessary at the cost of some predictable amount of homomorphic capacity depending on the starting parameters. This technique allowed the authors of [4] to implement more efficient approximation of non-linear functions using homomorphic look-up tables. This also simplifies the homomorphic evaluation of the decryption circuit of a symmetric scheme that works bit-wise. In [4] BGV was used with plaintext space the m-th cyclotomic integer ring for $m = 2^{15} - 1$ and plaintext modulus 2^{11}. This corresponds to lattices of dimension $\phi(m) = 27{,}000$ and a recryption step costing 20 levels. This is the cost of extracting the 11 encrypted bits from each slot. They chose to use 29 levels in the BGV moduli-chain resulting in a ciphertext modulus q of roughly 1030 bits and overall security of more than 80 bits. These parameters result in 1800 plaintext slots per ciphertext, each able to contain 11 bit integers. We expect this should also be sufficient for our case, and the ability to extract encrypted bits will allow us to directly use the existing implementation of the LowMC block cipher demonstrated in [1] to homomorphically operate on encrypted bits using BGV.

4 The Machine Learning Model

The machine learning model should take as input the patient's encrypted record and new encrypted prescription. Then it runs a classification algorithm resulting in a decision which tells the pharmacy whether or not they should dispense the medication. The traditional models used for prediction in the healthcare industry involve logistic regression, support vector machines (SVM), and random forests. Homomorphic encryption is not well suited to the branching computations involved in random forests. SVM requires keeping all of the training data to make predictions [3], which may or may not be suitable depending on how we choose to train the model. Logistic regression is a feasible option that is well studied in the context of BGV [4]. Neural networks are also a candidate, assuming we can approximate the non-linear functions involved in a way suitable for homomorphic operations. In any case, it will be necessary to either extract feature vectors from the encrypted patient records using the techniques of Algorithm 1 or to store patient records as feature vectors directly.

4.1 Training the Model

Our options for training the model are severely restricted by our requirement that each patient's data be encrypted under a distinct key. Training on encrypted data in this setting is not straightforward. Instead, we propose to generate anonymous patient records and prescriptions and ask health professionals to assign a label depending on if the prescription should be approved or not. This way the model can be trained on clear data and only predictions involve computation on encrypted data.

4.2 A Remark on Using ML

There are some concerns that should be taken into consideration when using machine learning in this context. Since the training data must be labeled by health professionals it will, unfortunately, capture any biases in their decisions. Ideally, our system would not just improve privacy but also the accuracy and fairness in the decision to prescribe a patient controlled medications. To this end, it may be more efficient to encode a set of rules for approving or denying controlled prescriptions and bypass machine learning altogether. This will need to be taken into consideration in any practical application.

5 Authentication

Assume that a patient has submitted an encrypted prescription $\textsf{Enc}_{pk}(P)$ to the server. The server feeds the prescription and encrypted patient record $\textsf{Enc}_{pk}(R)$ to the model, which produces a encrypted label $\textsf{Enc}_{pk}(\text{res})$. Now the server is tasked with sending this data to the patient, to be decrypted with the patient's secret key, and presented at the pharmacy. It is possible that the patient may try to modify the result of the model, make up a fake result altogether, or mix and match labels and prescriptions to their advantage. In order to prevent patient tampering during this transaction, we introduce an authentication step under the assumption that the server and pharmacy have previously agreed on a shared secret key.

5.1 The Shared Secret Key

One may consider letting the server sign a digital signature over the ciphertext. However, this is not enough for the pharmacy to verify the authenticity of the plaintext result. One solution is homomorphic evaluation of the decryption circuit of a symmetric-key encryption scheme. Implementations of AES [6] as well as LowMC [1]—a block cipher with low AND depth and low multiplicative complexity specially developed for use with FHE—have been developed using HElib. Lattice based schemes have also been studied in this context and show promise [5]. For this work, we will use LowMC.

Assume that each pharmacy shares a secret key ss with the server, which is also the encryption key of the algorithm LowMC, $\pi()$. In phase 1 of our scheme, the server will return to the patient the output of the model as well as the encryption under the shared secret key: $\textsf{Enc}_{pk}(\text{res})$, $\textsf{Enc}_{pk}(\pi_{ss}(\text{res}))$, and $\textsf{Enc}_{pk}(\pi_{ss}(P))$. The patient then decrypts the response using the FHE secret key sk and obtains res, $\pi_{ss}(\text{res})$, and $\pi(P)$. The patient can see the output of the model themselves, and sends $\pi_{ss}(\text{res})$ and $\pi_{ss}(P)$ to the pharmacy. The pharmacy decrypts to recover the result of the model, and checks that the prescription P matches that presented by the patient.

5.2 Prevent Patient Tampering

Lastly, we need to ensure the patient can not mix and match the results of the model with different prescriptions. One possible solution is to preserve a plaintext slot for a serial number or timestamp, i. Then before encryption with $\pi()$, the server inserts i into the plaintext slot using the same strategy used previously to update a patient record. The patient receives $\textbf{Enc}_{pk}(\text{res}, i)$, $\textbf{Enc}_{pk}(\pi_{ss}(\text{res}, i))$, and $\textbf{Enc}_{pk}(\pi_{ss}(P, i))$. On final decryption, the pharmacy can verify that the slots match. Using timestamps has the added advantage of allowing a potential expiration for the authentication. If the check passes, the pharmacy can fill the prescription or not based on the decision made by the machine learning model.

References

1. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *Advances in cryptology—EUROCRYPT 2015. Part I*, volume 9056 of *Lecture Notes in Comput. Sci.*, pages 430–454. Springer, Heidelberg, 2015.
2. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. https://eprint.iacr.org/2011/277.
3. Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. In *Machine Learning*, pages 273–297, 1995.
4. Jack L.H. Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing Real Work with FHE: The Case of Logistic Regression. Cryptology ePrint Archive, Report 2018/202, 2018. https://eprint.iacr.org/2018/202.
5. Pierre-Alain Fouque, Benjamin Hadjibeyli, and Paul Kirchner. Homomorphic Evaluation of Lattice-Based Symmetric Encryption Schemes. In Thang N. Dinh and My T. Thai, editors, *Computing and Combinatorics*, pages 269–280, Cham, 2016. Springer International Publishing.
6. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. Cryptology ePrint Archive, Report 2012/099, 2012. https://eprint.iacr.org/2012/099.
7. S. Halevi and V. Shoup. HElib - an implementation of homomorphic encryption, September 2014.
8. H. Hedegaard, B.A. Bastian, J.P. Trinidad, M. Spencer, and M. Warner. Drugs most frequently involved in drug overdose deaths: United States, 2011–2016. *National Vital Statistics Reports*, 67, 2018.
9. N.P. Smart and F. Vercauteren. Fully Homomorphic SIMD Operations. Cryptology ePrint Archive, Report 2011/133, 2011. https://eprint.iacr.org/2011/133.
10. Times Staff Writer. Tampa Bay Times, 2013. https://www.tampabay.com/news/politics/did-floridas-prescription-pill-database-really-spring-a-leak/2130108/.

Correction to: Introduction to Homomorphic Encryption and Schemes

Jung Hee Cheon, Anamaria Costache, Radames Cruz Moreno, Wei Dai, Nicolas Gama, Mariya Georgieva, Shai Halevi, Miran Kim, Sunwoong Kim, Kim Laine, Yuriy Polyakov, and Yongsoo Song

Correction to:
Chapter 1 in: K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*, https://doi.org/10.1007/978-3-030-77287-1_1

The original version of this book has been revised because it was inadvertently published with the following errors which have now been updated.

Chapter 1
Page 7: In Fig. 1, the term "selction" has been updated to "selection"
Page 7: "(R) LWE" has been updated to "(R)LWE"
Page 8: The symbol "n" has been set as italics
Page 9: In footnote, "rotations, see the" has been updated to "rotations. See the"
Page 10: "secret-key" has been updated to "secret key"
Page 20: "a • x" has been updated to "a * x"
Page 21: "Such operations are described in library-specific whitepapers" has been removed
Page 24: "(this operation is implicit on representations modulo 1, where the precision of the ciphertext representation is always of the order of α)" has been updated to "This operation is implicit on representations modulo 1, where the precision of the ciphertext representation is always of the order of α"

The updated online versions of this chapter can be found at
https://doi.org/10.1007/978-3-030-77287-1_1

K. Lauter et al. (eds.), *Protecting Privacy through Homomorphic Encryption*,
https://doi.org/10.1007/978-3-030-77287-1_15