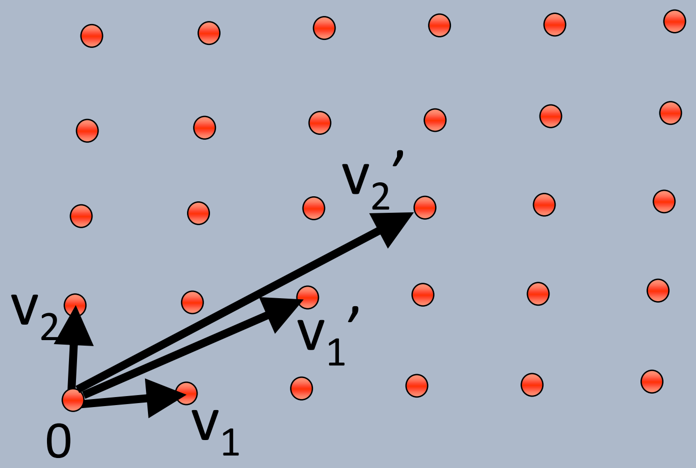
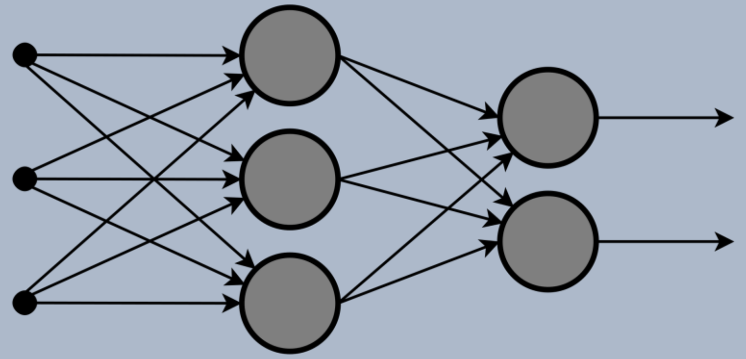
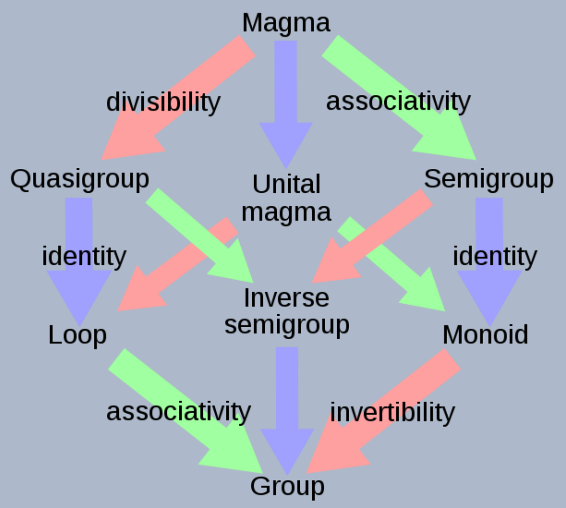
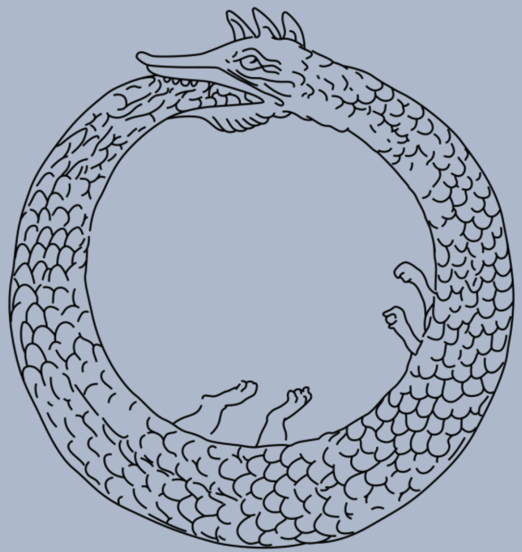




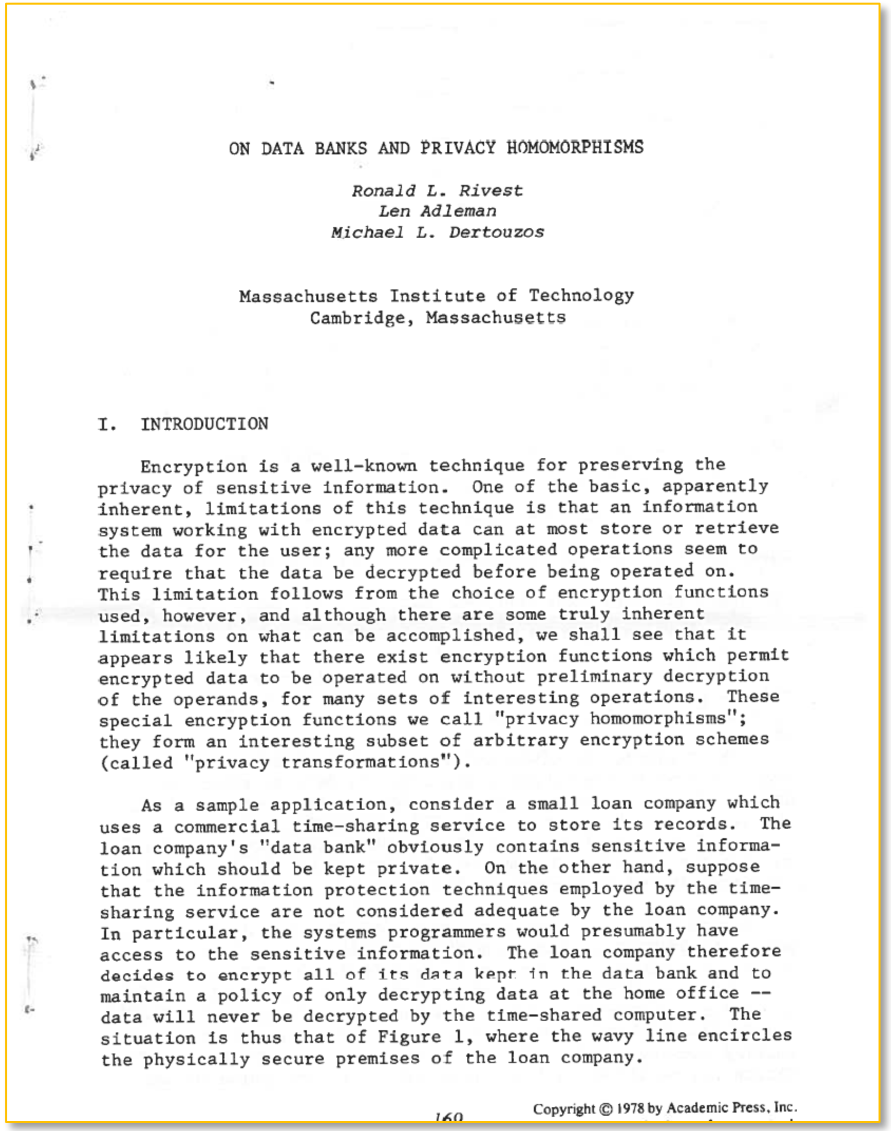


A Decade (or So) of Fully Homomorphic Encryption



Craig Gentry
Algorand Foundation
Eurocrypt 2021

Homomorphic Encryption



ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

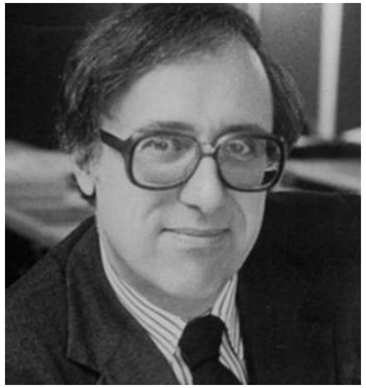
I. INTRODUCTION

Encryption is a well-known technique for preserving the privacy of sensitive information. One of the basic, apparently inherent, limitations of this technique is that an information system working with encrypted data can at most store or retrieve the data for the user; any more complicated operations seem to require that the data be decrypted before being operated on. This limitation follows from the choice of encryption functions used, however, and although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call "privacy homomorphisms"; they form an interesting subset of arbitrary encryption schemes (called "privacy transformations").

As a sample application, consider a small loan company which uses a commercial time-sharing service to store its records. The loan company's "data bank" obviously contains sensitive information which should be kept private. On the other hand, suppose that the information protection techniques employed by the time-sharing service are not considered adequate by the loan company. In particular, the systems programmers would presumably have access to the sensitive information. The loan company therefore decides to encrypt all of its data kept in the data bank and to maintain a policy of only decrypting data at the home office -- data will never be decrypted by the time-shared computer. The situation is thus that of Figure 1, where the wavy line encircles the physically secure premises of the loan company.

160

Copyright © 1978 by Academic Press, Inc.

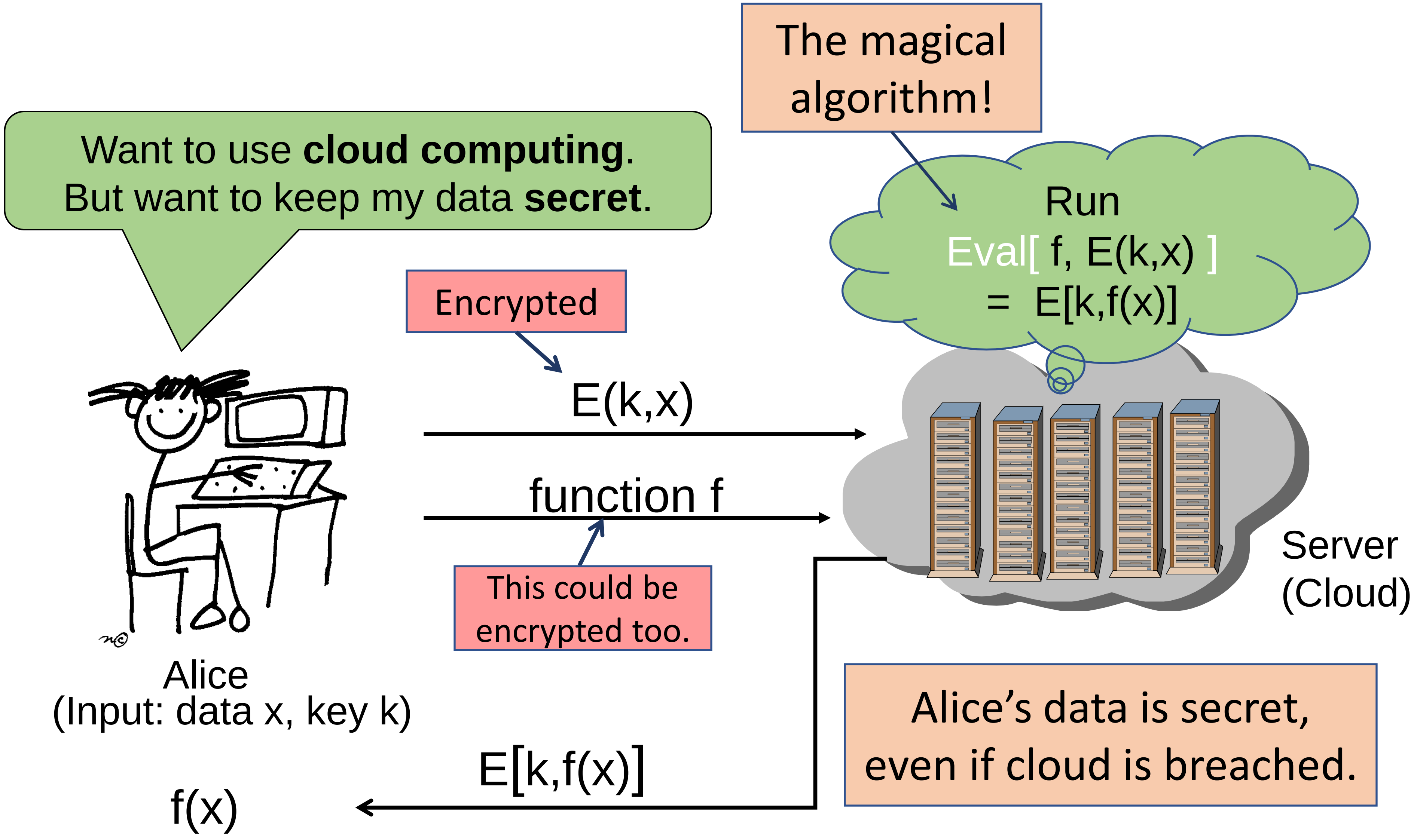


1978: Rivest, Adleman, Dertouzos,
"On Data Banks and $\boxed{\text{Privacy Homomorphisms}}$"

Homomorphic Encryption

Can we delegate the *processing* of data
without giving away *access* to it?

Cloud Computing on Encrypted Data



A Fully Homomorphic Encryption Scheme (FHE)

Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

ABSTRACT

We propose a fully homomorphic encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. Our solution comes in three steps. First, we provide a general result – that, to construct an encryption scheme that permits evaluation of *arbitrary circuits*, it suffices to construct an encryption scheme that can evaluate (slightly augmented versions of) its *own decryption circuit*; we call a scheme that can evaluate its (augmented) decryption circuit *bootstrappable*.

Next, we describe a public key encryption scheme using *ideal lattices* that is *almost* bootstrappable. Lattice-based cryptosystems typically have decryption algorithms with low circuit complexity, often dominated by an inner product computation that is in NC1. Also, *ideal* lattices provide both additive and *multiplicative* homomorphisms (modulo a public-key ideal in a polynomial ring that is represented as a lattice), as needed to evaluate general circuits.

Unfortunately, our initial scheme is not quite bootstrappable – i.e., the depth that the scheme can correctly evaluate can be logarithmic in the lattice dimension, just like the depth of the decryption circuit, but the latter is greater than the former. In the final step, we show how to modify the scheme to reduce the depth of the decryption circuit, and thereby obtain a bootstrappable encryption scheme, without reducing the depth that the scheme can evaluate. Abstractly, we accomplish this by enabling the *encrypter* to start the decryption process, leaving less work for the decrypter, much like the server leaves less work for the decrypter in a server-aided cryptosystem.

Categories and Subject Descriptors: E.3 [Data Encryption]: Public key cryptosystems

General Terms: Algorithms, Design, Security, Theory

1. INTRODUCTION

We propose a solution to the old open problem of constructing a *fully homomorphic encryption scheme*. This notion, originally called a *privacy homomorphism*, was introduced by Rivest, Adleman and Dertouzos [54] shortly after the invention of RSA by Rivest, Adleman and Shamir [55]. Basic RSA is a multiplicatively homomorphic encryption scheme – i.e., given RSA public key pk $= (N, e)$ and ciphertexts $\{\psi_i \leftarrow \pi_i^e \bmod N\}$, one can efficiently compute $\prod_i \psi_i = (\prod_i \pi_i)^e \bmod N$, a ciphertext that encrypts the product of the original plaintexts. Rivest et al. [54] asked a natural question: What can one do with an encryption scheme that is *fully* homomorphic: a scheme $\mathcal{E}$ with an efficient algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that, for any valid public key pk, *any* circuit C (not just a circuit consisting of multiplication gates), and any ciphertexts $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, outputs

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \psi_1, \ldots, \psi_t) ,$$

a valid encryption of $C(\pi_1, \ldots, \pi_t)$ under pk? Their answer: one can arbitrarily *compute on encrypted data* – i.e., one can process encrypted data (query it, write into it, do anything to it that can be efficiently expressed as a circuit) without the decryption key. As an application, they suggested private data banks: a user can store its data on an untrusted server in encrypted form, yet still allow the server to process, and respond to, the user's data queries (with responses more concise than the trivial solution: the server just sends all of the encrypted data back to the user to process). Since then, cryptographers have accumulated a list of "killer" applications for fully homomorphic encryption. However, prior to this proposal, *we did not have a viable construction.*

1.1 Homomorphic Encryption

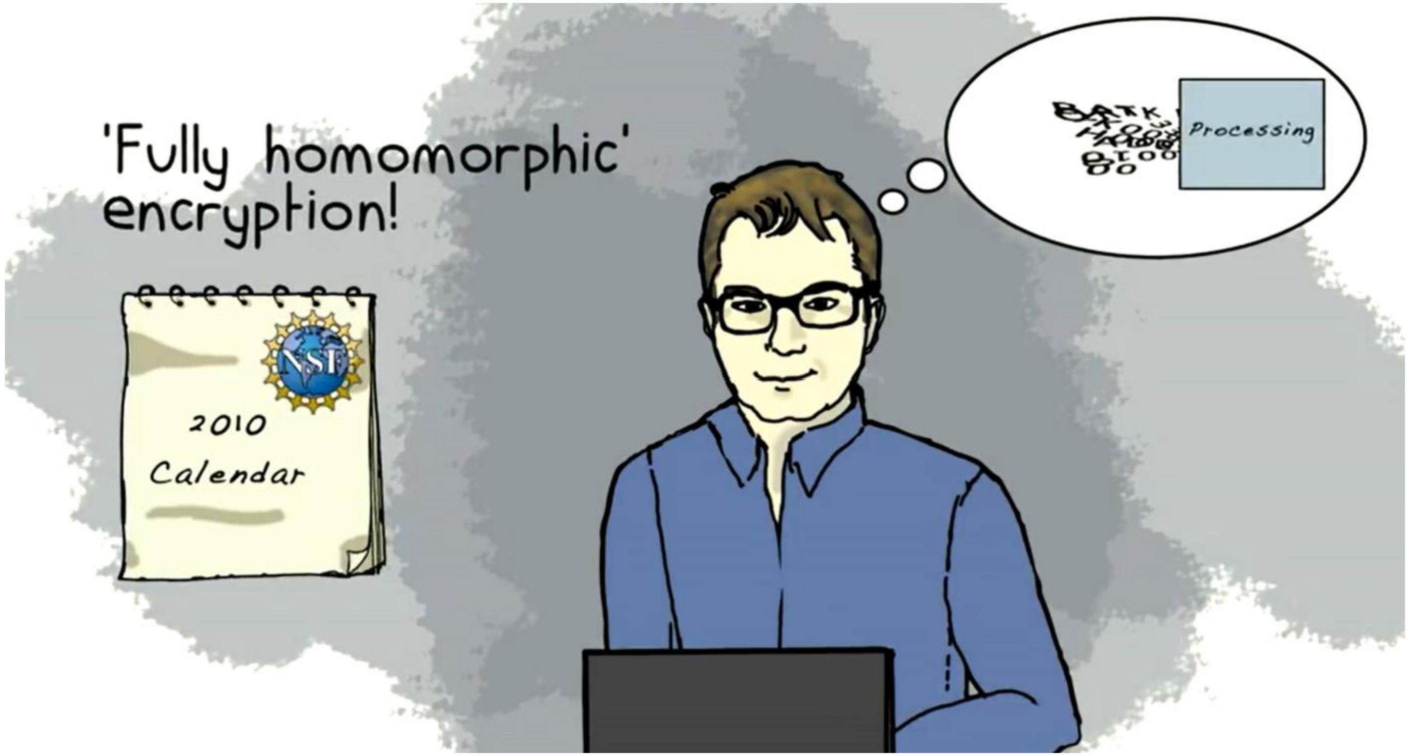
A homomorphic public key encryption scheme $\mathcal{E}$ has four algorithms $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Decrypt}_{\mathcal{E}}$, and an additional algorithm $\mathsf{Evaluate}_{\mathcal{E}}$ that takes as input the public key pk, a circuit C from a permitted set $\mathcal{C}_{\mathcal{E}}$ of circuits, and a tuple of ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$; it outputs a ciphertext ψ. The computational complexity of all of these algorithms must be polynomial in security parameter λ and (in the case of $\mathsf{Evaluate}_{\mathcal{E}}$) the size of C. $\mathcal{E}$ is *correct* for circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any key-pair (sk, pk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, any plaintexts $\pi_1, \ldots, \pi_t$, and any ciphertexts $\Psi = \langle \psi_1, \ldots, \psi_t \rangle$ with $\psi_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, \pi_i)$, it is the case that:

$$\psi \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi) \Rightarrow C(\pi_1, \ldots, \pi_t) = \mathsf{Decrypt}_{\mathcal{E}}(\mathrm{sk}, \psi)$$

By itself, mere correctness does not exclude trivial schemes.[1] So, we require ciphertext size and decryption time to be up-

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

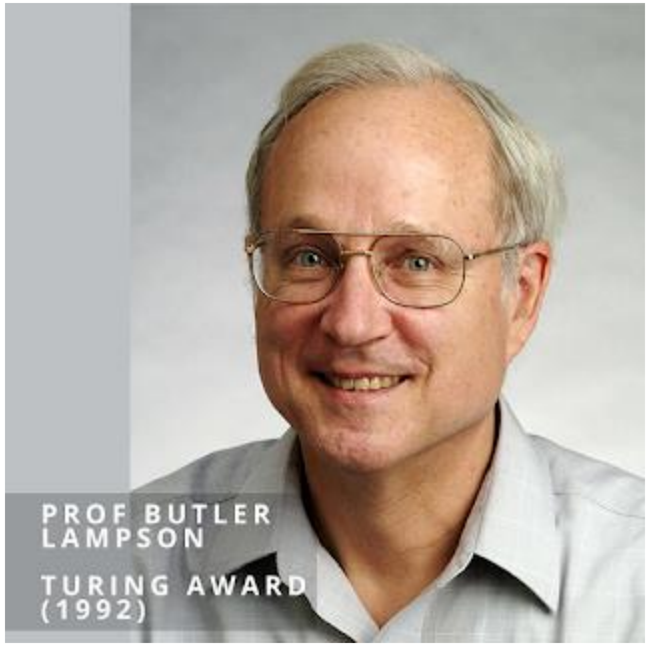
[1] In particular, we could define $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, C, \Psi)$ to just output (C, Ψ) without "processing" the circuit or ciphertexts at all, and $\mathsf{Decrypt}_{\mathcal{E}}$ to decrypt the component ciphertexts and apply C to results.



2009: Gentry,
"Fully Homomorphic Encryption Using Ideal Lattices"

What if a homomorphic encryption scheme could decrypt itself with an encrypted secret key?

OK, it's slow. How slow could it be?



"I don't think we'll see anyone using Gentry's solution in our lifetimes."

He's right!

Level Up

Homomorphic Encryption Standardization Workshop 2017

Current solutions are so much better!

(with amazing contributions from many people)

Organization of this Talk

Past

Present

Future

Proof of Concept → Usable Tool

Homomorphism

Attacks (like linear algebra)

Semantically Secure Encryption

New Homomorphisms to Explore?

Proof of Concept → Usable Tool

First 2 Generations of FHE

Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

Fully Homomorphic Encryption over the Integers

Marten van Dijk[1], Craig Gentry[2], Shai Halevi[2], and Vinod Vaikuntanathan[2]

[1] MIT CSAIL
[2] IBM Research

Implementing Gentry's Fully-Homomorphic Encryption Scheme

Craig Gentry* and Shai Halevi*

IBM Research

Abstract. We describe a working implementation of a variant of Gentry's fully homomorphic encryption scheme (STOC 2009), similar to the variant used in an earlier implementation effort by Smart and Vercauteren (PKC 2010). Smart and Vercauteren implemented the underly-

EFFICIENT FULLY HOMOMORPHIC ENCRYPTION FROM (STANDARD) LWE*

ZVIKA BRAKERSKI† AND VINOD VAIKUNTANATHAN‡

(Leveled) Fully Homomorphic Encryption without Bootstrapping

Zvika Brakerski* Craig Gentry† Vinod Vaikuntanathan‡

Fully Homomorphic SIMD Operations

N.P. Smart[1] and F. Vercauteren[2]

Fully Homomorphic Encryption with Polylog Overhead

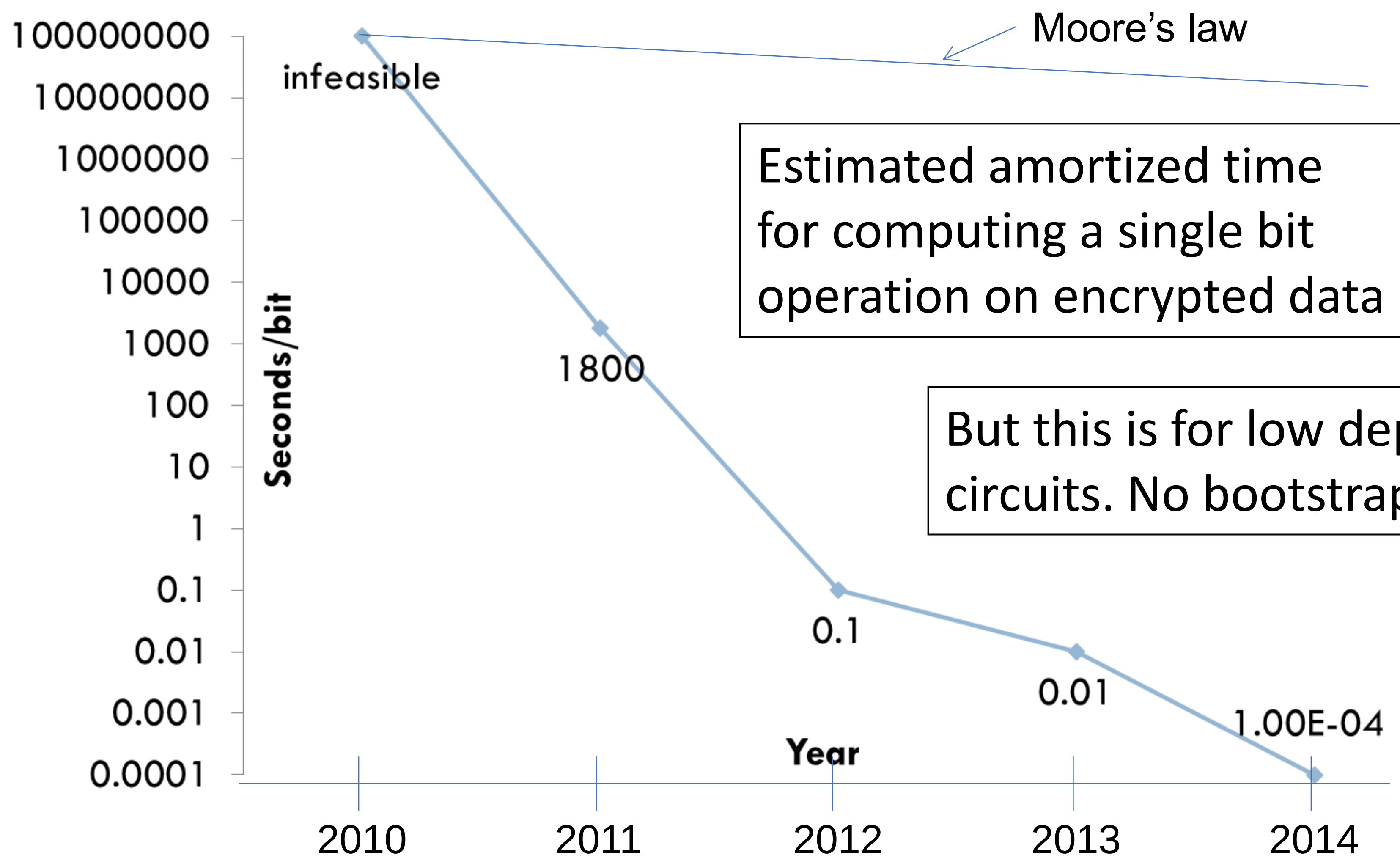
Craig Gentry[1], Shai Halevi[1], and Nigel P. Smart[2]

[1] IBM T.J. Watson Research Center, Yorktown Heights, New York, U.S.A.
[2] Dept. Computer Science, University of Bristol, Bristol, United Kingdom

Abstract. We show that homomorphic evaluation of (wide enough) arithmetic circuits can be accomplished with only polylogarithmic overhead. Namely, we present a construction of fully homomorphic encryption (FHE) schemes that for security parameter λ can evaluate any width-$\Omega(\lambda)$ circuit with t gates in time $t \cdot \mathrm{polylog}(\lambda)$.

To get low overhead, we use the recent batch homomorphic evaluation techniques of Smart-Vercauteren and Brakerski-Gentry-Vaikuntanathan,

Speed of Computing on Encrypted Data on IBM's HElib Platform (1st and 2nd Gens)



Estimated amortized time for computing a single bit operation on encrypted data

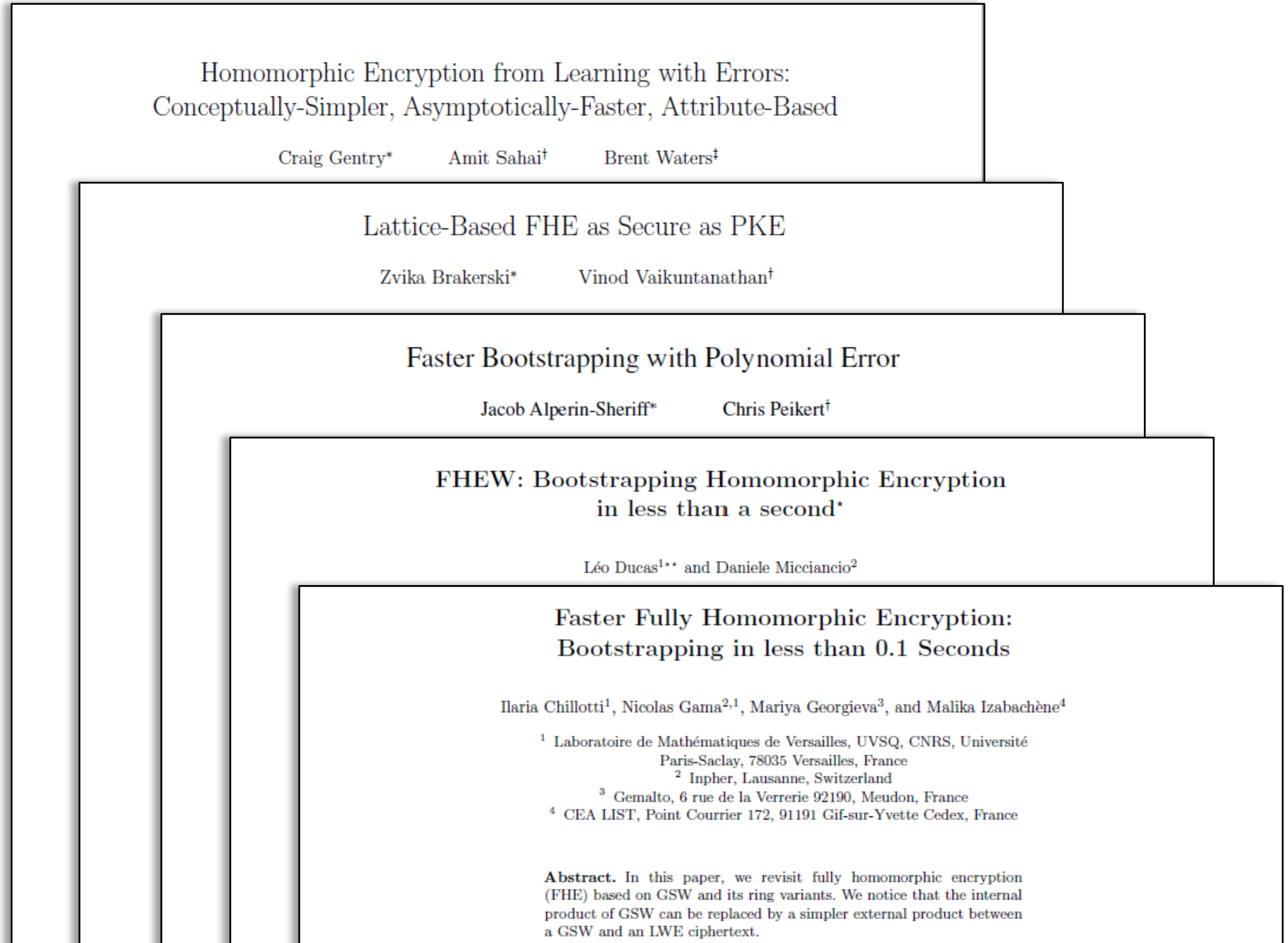
But this is for low depth (e.g., 10) circuits. No bootstrapping here!

Bootstrapping in HElib as of 2020 (BGV Scheme)

cyclotomic ring m	21845 $=257{\cdot}5{\cdot}17$	18631 $=601{\cdot}31$	28679 $=241{\cdot}7{\cdot}17$	35113 $=37{\cdot}13{\cdot}73$
lattice dim. $\phi(m)$	16384	18000	23040	31104
plaintext space	GF(2)	GF(2)	GF(2)	GF(2)
number of slots	1024	720	960	864
recrypt params e/e'	12/4	16/9	12/4	12/4
before capacity	491	489	578	820
after capacity	247	298	329	580
min capacity	41.1	34.2	41.9	32.6
bits per level	15.2	15.4	15.8	15.9
usable levels	13	17	18	34
linear transforms (sec)	4	3	4	10
total recrypt (sec)	15	11	19	40
amortized time (ms)	1.1	0.9	1.1	1.4
space usage (GB)	1.8	1.8	1.6	3.5

Experimental results with thin bootstrapping.
"Thin bootstrapping" is a technique by Chen and Han (Eurocrypt '18).

3rd Generation FHE



Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based

Craig Gentry* Amit Sahai† Brent Waters‡

Lattice-Based FHE as Secure as PKE

Zvika Brakerski* Vinod Vaikuntanathan†

Faster Bootstrapping with Polynomial Error

Jacob Alperin-Sheriff* Chris Peikert†

FHEW: Bootstrapping Homomorphic Encryption in less than a second*

Léo Ducas[1,**] and Daniele Micciancio[2]

Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds

Ilaria Chillotti[1], Nicolas Gama[2,1], Mariya Georgieva[3], and Malika Izabachène[4]

[1] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
[2] Inpher, Lausanne, Switzerland
[3] Gemalto, 6 rue de la Verrerie 92190, Meudon, France
[4] CEA LIST, Point Courrier 172, 91191 Gif-sur-Yvette Cedex, France

Abstract. In this paper, we revisit fully homomorphic encryption (FHE) based on GSW and its ring variants. We notice that the internal product of GSW can be replaced by a simpler external product between a GSW and an LWE ciphertext.

Bootstrapping is faster!

But ciphertext packing not natively supported.

4th Generation FHE (CKKS Scheme)

Homomorphic Encryption for Arithmetic of Approximate Numbers

Jung Hee Cheon[1], Andrey Kim[1], Miran Kim[2], and Yongsoo Song[1]

[1] Seoul National University, Republic of Korea
{jhcheon, kimandrik, lucius05}@snu.ac.kr
[2] University of California, San Diego
mrkim@ucsd.edu

Abstract. We suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new *rescaling* procedure for managing the magnitude of plaintext. This procedure truncates a ciphertext into a smaller modulus, which leads to rounding of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. As a result, our decryption structure outputs an approximate value of plaintext with a predetermined precision.
We also propose a new batching technique for a RLWE-based construction. A plaintext polynomial is an element of a cyclotomic ring of char-

1st and 2nd Gens: mod-p numbers, arithmetic circuits

3rd Gen: bits, boolean circuits

4th Gen: real (or complex) numbers, approximate (floating pt) arithmetic

Works great in apps that use floating point, like neural networks

Libraries for FHE

Library/Scheme	FHEW	TFHE	BGV	BFV	CKKS
cuFHE		✔			
FHEW	✔				
FV-NFLlib				✔	
HEAAN					✔
HElib			✔		(✔)
PALISADE			✔	✔	(✔)
SEAL				✔	✔
TFHE(-Chimera)	✔	✔		(✔)	(✔)

$\Lambda \circ \lambda$ (LOL): Haskell-based implementation

Chimeric FHE

TFHE: Fast Fully Homomorphic Encryption over the Torus*

Ilaria Chillotti[1], Nicolas Gama[3,2], Mariya Georgieva[4,3], and Malika Izabachène[5]

Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE

Ilaria Chillotti[1], Damien Ligier[1], Jean-Baptiste Orfila[1], and Samuel Tap[1]

CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes

Christina Boura[1,4], Nicolas Gama[1,2], Mariya Georgieva[2,3], and Dimitar Jetchev[2,3]

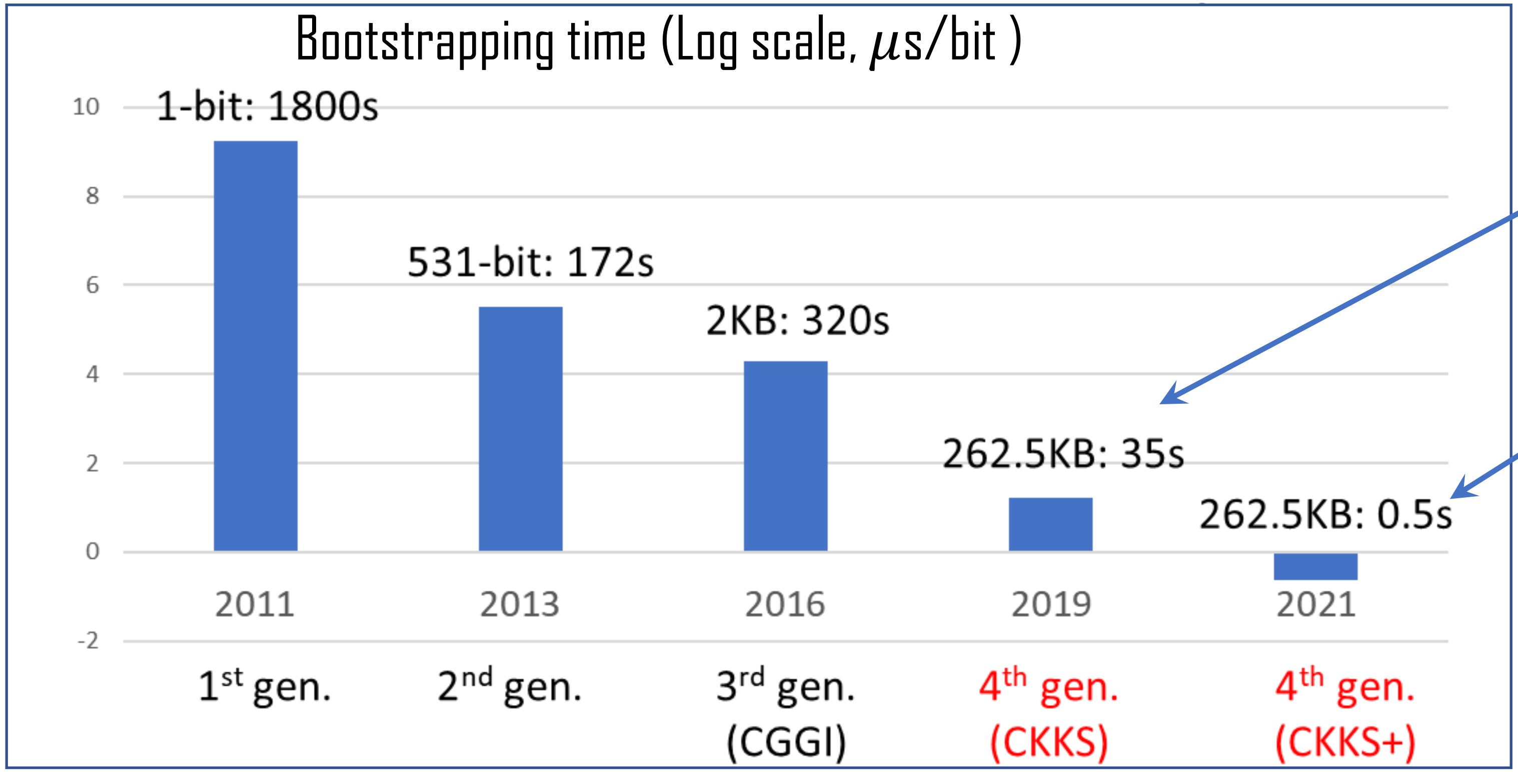
[1] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, Versailles, France
[2] Inpher, Lausanne, Switzerland
[3] EPFL, Lausanne, Switzerland

Switch b/w 2[nd], 3[rd], and 4[th] gen schemes as appropriate

HE Speedup (slide courtesy of Jung Hee Cheon)

- HE is getting faster 8 times every year

  e.g. Bootstrapping time: the most time-consuming operation in HE



$19\mu s/\text{bit}$ bootstrapping time! (amortized)

$0.29\mu s/\text{bit}$ bootstrapping time! (amortized)

CKKS: CPU-based, CKKS+: GPU-based

서울대학교 SEOUL NATIONAL UNIVERSITY CRYPTO LAB

App: Genome-Wide Association Studies (GWAS)

Secure large-scale genome-wide association studies using homomorphic encryption

Marcelo Blatt[a,1], Alexander Gusev[a,b,1], Yuriy Polyakov[a,1,2], and Shafi Goldwasser[a,c,1,2]

[a]Duality Tech
Theory of Co

Contributed

Genome-wi genetic var approach f for GWASs typically ha for method ipants. Her on several strate that dataset of data encry lations sho and 500,00 a single se in parallel). magnitude multiparty actions, wi homomorp domains w data are ne

genome-wid homomorphi

A assoc polym notype or typically p regression, and *P* valu stream use reported as mechanism risk score a A critic

Ultra-Fast Homomorphic Encryption Models enable Secure Outsourcing of Genotype Imputation

Miran Kim[1,+], Arif Harmanci[2,+,*], Jean-Philippe Bossuat[3], Sergiu Carpov[4,5], Jung Hee Cheon[6,7], Ilaria Chillotti[8], Wonhee Cho[6], David Froelicher[3], Nicolas Gama[4], Mariya Georgieva[4], Seungwan Hong[6], Jean-Pierre Hubaux[3], Duhyeong Kim[6], Kristin Lauter[9], Yiping Ma[10], Lucila Ohno-Machado[11], Heidi Sofia[12], Yongha Son[13], Yongsoo Song[9], Juan Troncoso-Pastoriza[3], and Xiaoqian Jiang[1,*]

[1]Center for Secure Artificial intelligence For hEalthcare (SAFE), School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX, 77030, USA.
[2]Center for Precision Health, School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX, 77030, USA.
[3]École polytechnique fédérale de Lausanne, Switzerland.
[4]Inpher, EPFL Innovation Park Bàtiment A, 3rd Fl, 1015 Lausanne, Switzerland.
[5]CEA, LIST, 91191 Gif-sur-Yvette Cedex, France.
[6]Department of Mathematical Sciences, Seoul National University, Seoul, 08826, Republic of Korea.
[7]Crypto Lab Inc., Seoul, 08826, Republic of Korea.
[8]Zama, Paris, France and imec-COSIC, KU Leuven, Leuven, Belgium.
[9]Microsoft Research, Redmond, WA, 98052, USA.
[10]School of EECS, Peking University, Beijing, People's Republic of China.
[11]UCSD Health Department of Biomedical Informatics, University of California, San Diego, CA, 92093, USA.
[12]National Institutes of Health (NIH) - National Human Genome Research Institute, Bethesda, MD, 20892, USA.
[13]Samsung SDS, Seoul, Republic of Korea.
*Corresponding authors: Arif.O.Harmanci@uth.tmc.edu, Xiaoqian.Jiang@uth.tmc.edu
+These authors contributed equally to this work

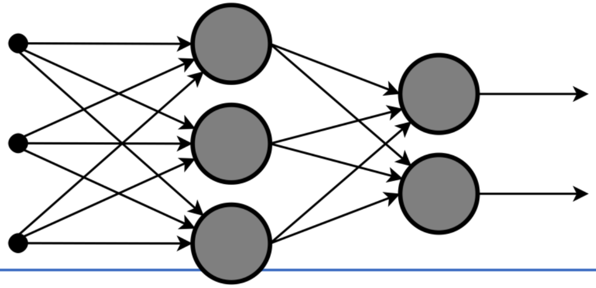
ABSTRACT

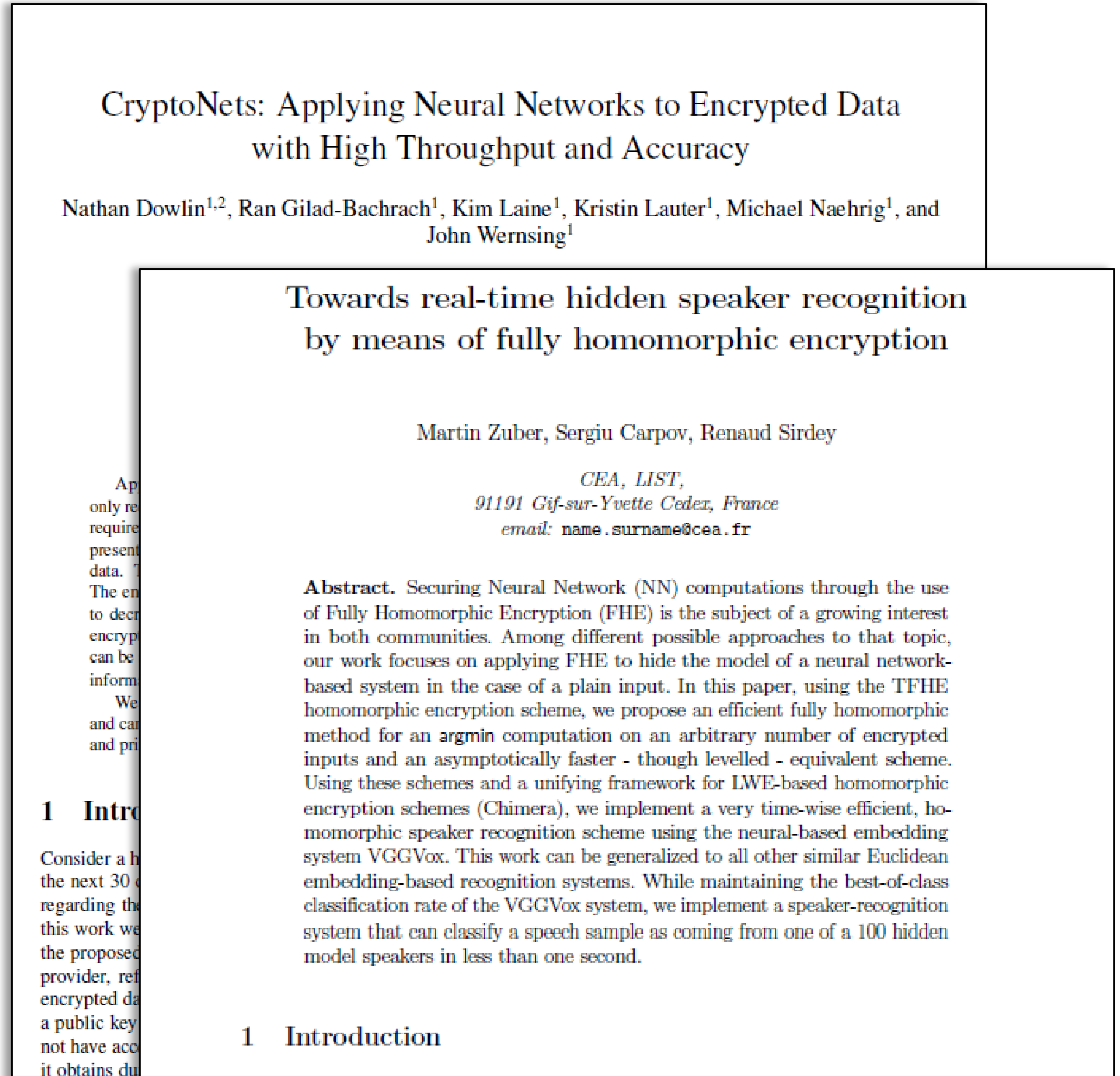
2nd paper by 4 winning teams of 2019 iDASH Genomic Privacy Challenge.

Algorithms: linear regression, logistic regression, and neural networks.

< 25s evaluation of imputation model for 80K SNPs

App: Neural Networks





CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy

Nathan Dowlin[1,2], Ran Gilad-Bachrach[1], Kim Laine[1], Kristin Lauter[1], Michael Naehrig[1], and John Wernsing[1]

Towards real-time hidden speaker recognition by means of fully homomorphic encryption

Martin Zuber, Sergiu Carpov, Renaud Sirdey

CEA, LIST,
91191 Gif-sur-Yvette Cedex, France
email: name.surname@cea.fr

Abstract. Securing Neural Network (NN) computations through the use of Fully Homomorphic Encryption (FHE) is the subject of a growing interest in both communities. Among different possible approaches to that topic, our work focuses on applying FHE to hide the model of a neural network-based system in the case of a plain input. In this paper, using the TFHE homomorphic encryption scheme, we propose an efficient fully homomorphic method for an argmin computation on an arbitrary number of encrypted inputs and an asymptotically faster - though levelled - equivalent scheme. Using these schemes and a unifying framework for LWE-based homomorphic encryption schemes (Chimera), we implement a very time-wise efficient, homomorphic speaker recognition scheme using the neural-based embedding system VGGVox. This work can be generalized to all other similar Euclidean embedding-based recognition systems. While maintaining the best-of-class classification rate of the VGGVox system, we implement a speaker-recognition system that can classify a speech sample as coming from one of a 100 hidden model speakers in less than one second.

1 Introduction

CryptoNets: First to evaluate neural networks including non-linear activation functions

Many papers since then on using FHE to evaluate neural nets

Recent example: Encrypted speaker recognition in <1 second

App: Private Information Retrieval (PIR)

On the Computational Practicality of Private Information Retrieval

Radu Sion *
Network Security and Applied Cryptography Lab
Computer Sciences, Stony Brook University
sion@cs.stonybrook.edu

Bogdan Carbunar
Pervasive Platforms and Architectures
Motorola Labs
carbunar@motorola.com

Abstract

We explore the limits of single-server computational private information retrieval (PIR) for the purpose of preserving client access patterns leakage. *We show that deployment of non-trivial single server PIR protocols on real hardware of the recent past would have been orders of magnitude less time-efficient than trivially transferring the entire database. We stress that these results are beyond existing knowledge of mere "impracticality" under unfavorable assumptions. They rather reflect an inherent limitation with respect to modern hardware, likely the result of a communication-cost centric protocol design. We argue that this is likely to hold on non-specialized traditional hardware in the foreseeable future. We validate our reasoning in an experimental setup on modern off-the-shelf hardware. Ultimately, we hope our results will stimulate practical designs.*

1 Introduction

Private Information Retrieval, (PIR) has been proposed as a primitive for accessing outsourced data over a network,

Here we discuss single-server computational PIR *for the purpose of preserving client access patterns leakage*. We show that deployment of non-trivial single server private information retrieval protocols on real hardware of the recent past would have been orders of magnitude more time-consuming than trivially transferring the entire database. The deployment of computational PIR would in fact *increase* overall execution time, as well as the probability of *forward* leakage, when the deployed present trapdoors become eventually vulnerable – e.g., today's queries will be revealed once factoring of today's values will become possible in the future.

We stress that this is beyond existing knowledge of mere "impracticality" under unfavorable assumptions. On real hardware, *no* existing non-trivial single server PIR protocol could have possibly had outperformed the trivial client-to-server transfer of records in the past, and is likely not to do so in the future either. This is due to the fact that on any known past general-purpose Von Neumann hardware, it is simply more expensive to PIR-process one bit of information than to transfer it over a network.

In particular, this impacts the type of complexity reasoning as found in [28] (section 2.4, page 971). The complexities discussed there do not consider the *significant* computa-

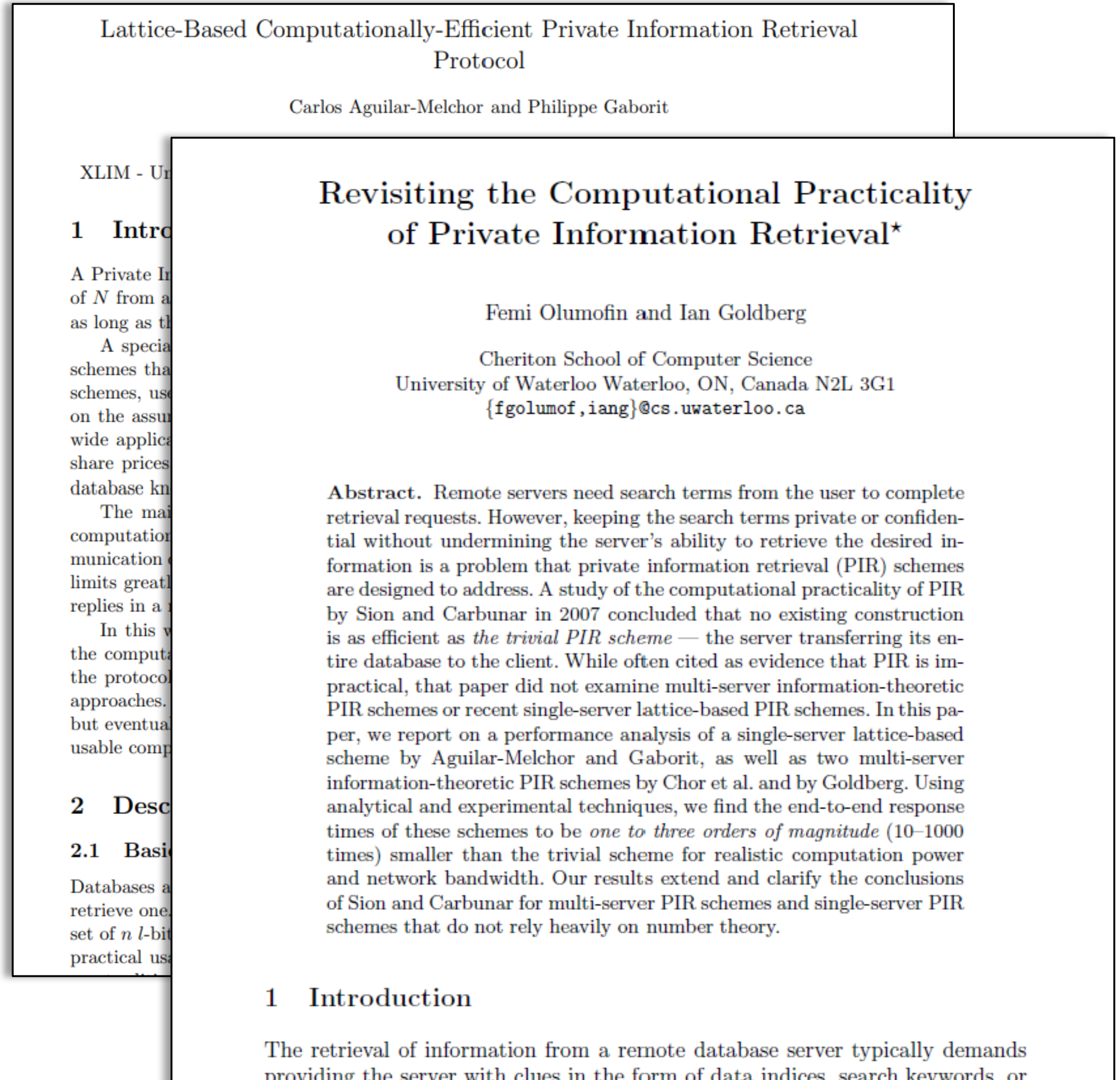
PIR: Get 1 item from a DB privately.

In response to query, Server must touch every item in DB.

Server overhead is a big concern.

Sion-Carbunar: Less expensive for Server to just send entire DB.

App: Private Information Retrieval (PIR)



Lattice-Based Computationally-Efficient Private Information Retrieval Protocol

Carlos Aguilar-Melchor and Philippe Gaborit

XLIM - Ur

1 Intro

2 Desc

2.1 Basi

Revisiting the Computational Practicality of Private Information Retrieval*

Femi Olumofin and Ian Goldberg

Cheriton School of Computer Science
University of Waterloo Waterloo, ON, Canada N2L 3G1
{fgolumof,iang}@cs.uwaterloo.ca

Abstract. Remote servers need search terms from the user to complete retrieval requests. However, keeping the search terms private or confidential without undermining the server's ability to retrieve the desired information is a problem that private information retrieval (PIR) schemes are designed to address. A study of the computational practicality of PIR by Sion and Carbunar in 2007 concluded that no existing construction is as efficient as *the trivial PIR scheme* — the server transferring its entire database to the client. While often cited as evidence that PIR is impractical, that paper did not examine multi-server information-theoretic PIR schemes or recent single-server lattice-based PIR schemes. In this paper, we report on a performance analysis of a single-server lattice-based scheme by Aguilar-Melchor and Gaborit, as well as two multi-server information-theoretic PIR schemes by Chor et al. and by Goldberg. Using analytical and experimental techniques, we find the end-to-end response times of these schemes to be *one to three orders of magnitude* (10–1000 times) smaller than the trivial scheme for realistic computation power and network bandwidth. Our results extend and clarify the conclusions of Sion and Carbunar for multi-server PIR schemes and single-server PIR schemes that do not rely heavily on number theory.

1 Introduction

The retrieval of information from a remote database server typically demands providing the server with clues in the form of data indices, search keywords, or

Sion-Carbunar does not apply to lattice-based PIR schemes, which are *much* faster.

App: Private Information Retrieval (PIR)

Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian

XPIR : Private Information Retrieval for Everyone

PIR with compressed queries and amortized query processing

Sebastian Angel[*†], Hao Chen[‡], Kim Laine[‡], and Srinath Setty[‡]

[*]The University of Texas at Austin [†]New York University [‡]Microsoft Research

Compressible FHE with Applications to PIR

Craig Gentry and Shai Halevi

Algorand Foundation*

cbgentry@gmail.com, shaih@alum.mit.edu

Abstract. Homomorphic encryption (HE) is often viewed as impractical, both in communication and computation. Here we provide an additively homomorphic encryption scheme based on (ring) LWE with nearly optimal rate ($1 - \epsilon$ for any $\epsilon > 0$). Moreover, we describe how to compress many FHE ciphertexts that may have come from a homomorphic evaluation (e.g., of the Gentry-Sahai-Waters (GSW) scheme), into fewer high-rate ciphertexts.
Using our high-rate HE scheme, we are able for the first time to describe a single-server private information retrieval (PIR) scheme with sufficiently low computational overhead so as to be practical for large databases. Single-server PIR inherently requires the server to perform at least one bit operation per database bit, and we describe a rate-(4/9) scheme with computation which is not so much worse than this inherent lower bound. In fact it is probably faster than whole-database AES encryption – specifically under 1.8 mod-q multiplication per database byte, where q is about 50 to 60 bits. Asymptotically, the computational overhead of our PIR scheme is $\tilde{O}(\log\log\lambda + \log\log\log N)$, where λ is the security parameter and N is the number of database files, which are assumed to be sufficiently large.

1 Introduction

XPIR and SealPIR even faster

GH19 scheme super-fast, with plaintext/ciphertext ratio of 4/9.

Preliminary implementation by Samir Menon and David Wu on one core (AWS c5n.2xlarge):

- PIR query on $2^{20} \times$ 30KB DB in 86.21s
- Compare: unaccelerated AES ECB encryption takes 85.63s.

PIR-time ≈ AES time!

Hardware Acceleration

DARPA Selects Researchers to Accelerate Use of Fully Homomorphic Encryption

Four research teams take on development of novel hardware accelerator to enable new levels of data and privacy protection

OUTREACH@DARPA.MIL
3/8/2021

70M (over 4 years)

Intel to Collaborate with Microsoft on DARPA Program

Intel today announced that it has signed an agreement with Defense Advanced Research Projects Agency (DARPA) to perform in its Data Protection in Virtual Environments (DPRIVE) program.

Hardware Acceleration

F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption (Extended Version)

Axel Feldmann[1*], Nikola Samardzic[1*], Aleksandar Krastev[1], Srini Devadas[1], Ron Dreslinski[2], Karim Eldefrawy[3], Nicholas Genise[3], Chris Peikert[2], Daniel Sanchez[1]

Over 100x Faster Bootstrapping in Fully Homomorphic Encryption through Memory-centric Optimization with GPUs

Wonkyung Jung[1], Sangpyo Kim[1], Jung Ho Ahn[1], Jung Hee Cheon[1] and Younho Lee[2]

[1] Seoul National University, Seoul, Republic of Korea, {jungwk,vnb987,gajh,jhcheon}@snu.ac.kr

[2] Seoul National University of Science and Technology, Seoul, Republic of Korea, younholee@seoultech.ac.kr

Abstract. Fully Homomorphic encryption (FHE) has been gaining popularity as an emerging way of enabling an unlimited number of operations on the encrypted message without decryption. A major drawback of FHE is its high computational cost. Especially, a *bootstrapping* that refreshes the noise accumulated through consequent FHE operations on the ciphertext is even taking minutes. This significantly limits the practical use of FHE in numerous real applications.

By exploiting massive parallelism available in FHE, we demonstrate the first GPU implementation for bootstrapping CKKS, one of the most promising FHE schemes that support arithmetic of approximate numbers. Through analyzing FHE operations, we discover that the major performance bottleneck is their high main-memory bandwidth requirement, which is exacerbated by leveraging existing optimizations targeted to reduce computation. These observations lead us to extensively utilize memory-centric optimizations such as kernel fusion and reordering primary functions.

Our GPU implementation shows a 7.02× speedup for a single FHE-multiplication compared to the state-of-the-art GPU implementation and 0.423us of amortized bootstrapping time per bit, which corresponds to a speedup of 257× over a single-threaded CPU implementation. By applying this to a logistic regression model training, we achieved a 40.0× speedup compared to the previous 8-thread CPU

0.423 **micro**seconds per bit (amortized) for bootstrapping!!

Usability

Toolkits (list from github.com/jonaschn/awesome-he)

- ALCHEMY - Haskell-based DSLs and interpreters/compilers, build on top of the lattice crypto library Lol.
- AWS HE toolkit - Simplifies the process of designing circuits for the CKKS scheme.
- Cingulata - Compiler toolchain and RTE for running C++ programs over encrypted data.
- E3 - Encrypt-Everything-Everywhere framework for compiling C++ programs with encrypted operands.
- Google's FHE Repository - Libraries and tools to perform FHE operations on an encrypted data set.
- IBM FHE toolkit - Including FHE ML inference with a Neural Network and a Privacy-Preserving key-value search.
    - fhe-toolkit-android - IBM FHE toolkit for Android
    - fhe-toolkit-ios - IBM FHE toolkit for iOS
    - fhe-toolkit-linux - IBM FHE toolkit for Linux (Docker based Centos, Fedora, Ubuntu & Alpine editions)
    - fhe-toolkit-macos - IBM FHE toolkit for macOS
- Marble - C++ framework that translates between nearly plaintext-style user programs and FHE computations.
- SHEEP - HE evaluation platform with a set of native benchmarks and a library agnostic language

Usability: Google Transpiler

A General Purpose Transpiler for Fully Homomorphic Encryption

Shruthi Gorantala, Rob Springer, Sean Purser-Haskell, William Lam, Royce Wilson, Asra Ali, Eric P. Astor, Itai Zukerman, Sam Ruth, Christoph Dibak, Phillipp Schoppmann, Sasha Kulankhina, Alain Forget, David Marn, Cameron Tew, Rafael Misoczki, Bernat Guillen, Xinyu Ye, Dennis Kraft, Damien Desfontaines, Aishe Krishnamurthy, Miguel Guevara, Irippuge Milinda Perera, Yurii Sushko, and Bryant Gipson

fhe-open-source@google.com

June 14, 2021

Abstract

Fully homomorphic encryption (FHE) is an encryption scheme which enables computation on encrypted data without revealing the underlying data. While there have been many advances in the field of FHE, developing programs using FHE still requires expertise in cryptography. In this white paper, we present a fully homomorphic encryption transpiler that allows developers to convert high-level code (e.g., C++) that works on unencrypted data into high-level code that operates on encrypted data. Thus, our transpiler makes transformations possible on encrypted data.

Our transpiler builds on Google's open-source XLS SDK [1] and uses an off-the-shelf FHE library, TFHE [2], to perform low-level FHE operations. The transpiler design is modular, which means the underlying FHE library as well as the high-level input and output languages can vary. This modularity will help accelerate FHE research by providing an easy way to compare arbitrary programs in different FHE schemes side-by-side. We hope this lays the groundwork for eventual easy adoption of FHE by software developers. As a proof-of-concept, we are releasing an experimental transpiler [3] as open-source software.

High-level code

$\downarrow$ No crypto expertise needed

Code that operates on encrypted data

Standardization: NIST (post-quantum crypto)

2015: 82 submissions

2020: 7 finalists

Type	PKE/KEM	Signature
Lattice[a]	• CRYSTALS-KYBER • NTRU • SABER	• CRYSTALS-DILITHIUM • FALCON
Code-based	• Classic McEliece	
Multivariate		• Rainbow

Lattice-based cryptography is here to stay.

Standardization: homomorphicencryption.org

Draft standard for homomorphic encryption schemes

Ongoing whitepapers on security, APIs, and applications

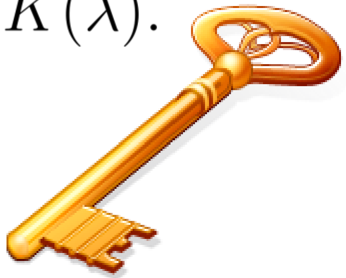
Homomorphism

$\downarrow$

Attacks (like linear algebra)

Semantically Secure Encryption

Public Key Encryption (PKE)



- **Key Generation**: $(sk, pk) \leftarrow K(\lambda)$.
- **Encryption**: $c \leftarrow E(pk, m)$.
- **Decryption**: $m \leftarrow D(sk, c)$.
  - K and E are probabilistic (randomized) algorithms.
- **Correctness**: $m = D(sk, E(pk, m))$ for all key pairs, messages, and encryption randomness.
- **Semantic security [Goldwasser-Micali '82] (for $m \in \{0, 1\}$)**: Distributions $(pk, E(pk, 0))$, $(pk, E(pk, 1))$ are indistinguishable (computationally hard to distinguish).
  - Any secure PKE must be *probabilistic*: many c's per m.

Homomorphic Encryption (HE)

- **Procedures**: K, E, D, and V (for "Evaluate")

- **Correctness**: For any function $f \in \mathcal{F}$ (a family of functions), and any ciphertexts $c_1 \leftarrow E(pk, m_1), \ldots, c_t \leftarrow E(pk, m_t)$,

  Set $c \leftarrow V(pk, f, c_1, \ldots, c_t)$, Then $D(sk, c) = f(m_1, \ldots, m_t)$.

- **Security**: Same as basic PKE. Again, many c's per m.

The Homomorphism in HE

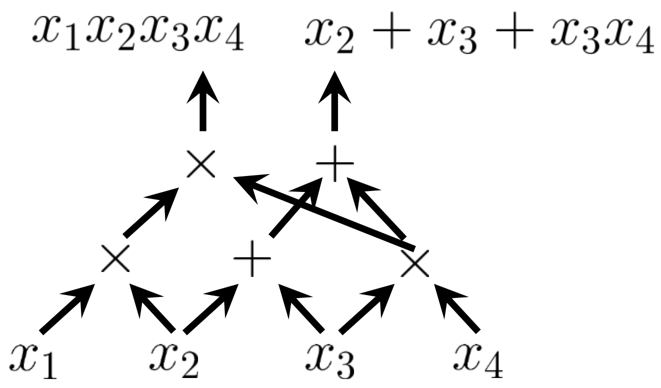
$\mathcal{M}$ = set of messages, $\mathcal{C}$ = set of ciphertexts

$$\begin{tikzcd}
\mathcal{C}^t \arrow[r, "V(pk, f, \cdot, \ldots, \cdot)"] \arrow[d, "D(sk, \cdot, \ldots, \cdot)"] & \mathcal{C} \arrow[d, "D(sk, \cdot)"] \\
\mathcal{M}^t \arrow[r, "f(\cdot, \ldots, \cdot)"] & \mathcal{M}
\end{tikzcd}$$

The order of D and f doesn't matter: either way we get $f(m_1, \ldots, m_t)$.

How Homomorphic?

- **Family of functions**: A homomorphic encryption scheme only works for $f \in \mathcal{F}$ (a family of functions).
- **Arithmetic Circuit:** A layered graph of $+$ and $\times$ gates



- ***Fully* Homomorphic Encryption:** Family of functions = all efficiently computable functions – e.g., all arithmetic circuits.

Two Ways to Develop HE Schemes

Cryptographic Way

1. Start with established crypto assumption
2. Build PKE based on the assumption
3. Try to make the PKE homomorphic
   (Build Eval with adds/mults)

Pro: Less likely to get insecure scheme
Con: Not likely to get any HE at all

Mathematical Way

1. Start with a set of homomorphisms $\{\delta\}$
2. Build a HE scheme where $\mathsf{D}(\mathsf{sk}, \cdot) \approx \delta_{sk}$
3. Try to make the HE scheme secure
   (Make ciphertexts look indist. from random)

Pro: Uncovers new useful alg. structure
Con: Risks crackpottery (and broken schemes)

Homomorphism

Homomorphism: A structure-preserving map between two algebraic structures that "preserves" the operations.

$$\delta : A \to B \text{ where } \delta(x *_A y) = \delta(x) *_B \delta(y).$$

Example:
$\delta(x) = e^x$

$$\begin{tikzcd}
\mathbb{R} \times \mathbb{R} \arrow[r, "+"] \arrow[d, "\text{component-wise } \delta"] & \mathbb{R} \arrow[d, "\delta"] \\
\mathbb{R}^+ \times \mathbb{R}^+ \arrow[r, "\times"] & \mathbb{R}^+
\end{tikzcd}$$

Homomorphisms + Complexity Theory

operations

homomorphisms

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\delta_1"] & B \arrow[r, "g"] \arrow[d, "\delta_2"] & C \arrow[d, "\delta_3"] & \cdots & \text{Ciphertext level} \arrow[d, "\mathsf{D}(sk, \cdot)"] \\ R \arrow[r, "u"] & S \arrow[r, "v"] & T & & \text{Plaintext level} \end{tikzcd}$$

Suppose: Rightward arrows (ops) are easy to compute
Downward arrows (homs) are hard to compute (w/o trapdoor)

You get homomorphic encryption!

Homomorphic encryption = homomorphisms + complexity theory

Using the Homomorphism for KeyGen & Encrypt

Thm. [Rothblum '11]: Semantically-secure private key HE (can do + mod 2) → Semantically-secure public key HE (can do + mod 2)

Technique:

1. **KeyGen**: pk includes k random encryptions of 0 and 1 each. Label them $\{c_i\}$ and $\{d_i\}$.
2. **Encrypt** $m \in \{0,1\}$: Pick random $S \subseteq [k]$, $m = |S| \bmod 2$. Output $c = \text{HE-add}(\{c_i : i \in \bar{S}\}, \{d_i : i \in S\})$.

Bonus: Only need to prove indist. of ciphertexts in public key.

Example: Building Goldwasser-Micali the Mathematical Way

- **Homomorphism**: Legendre symbol $\left(\frac{c}{p}\right)$ from $G = (\mathbb{Z}/p\mathbb{Z})^* \to \{-1, 1\}$

$$\begin{tikzcd}[column sep=huge]
c, d & G \times G \arrow[r, "\times"] \arrow[d, "\text{component-wise } \left(\frac{\cdot}{p}\right)"'] & G \arrow[d, "\left(\frac{\cdot}{p}\right)"] & c \cdot d \\
\left(\frac{c}{p}\right), \left(\frac{d}{p}\right) & \{\pm 1\} \times \{\pm 1\} \arrow[r, "\times"] & \{\pm 1\} & \left(\frac{c \cdot d}{p}\right) = \left(\frac{c}{p}\right) \cdot \left(\frac{d}{p}\right)
\end{tikzcd}$$

- **How to Make Secure?** If p is known, anyone can compute Legendre symbol
- **Possible Answer**: Hide p by using $N = p \cdot q$ with secret factorization
- **Quadratic Residuosity Assumption**: Given N (but not p, q) and x with $\left(\frac{x}{N}\right) = 1$, hard to tell whether $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$ or $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$.

Goldwasser-Micali: Rothblum Version

- **Secret key**: The factorization (p, q) of N.
- **Public key (a la Rothblum)**: N, $\{x_i, y_i \in \mathbb{Z}/N\mathbb{Z}\}$ such that $\left(\frac{x_i}{p}\right) = \left(\frac{x_i}{q}\right) = 1$ and $\left(\frac{y_i}{p}\right) = \left(\frac{y_i}{q}\right) = -1$.
- **Encryption (a la Rothblum)** of $m \in \{-1, 1\}$:
  1. Pick random subset $S \subseteq [k]$ with $m = (-1)^{|S|}$.
  2. Set $c \leftarrow \prod_{i \in \bar{S}} x_i \cdot \prod_{i \in S} y_i \bmod N$.
- **Decryption**: Output $m \leftarrow \left(\frac{c}{p}\right)$.
- **Homomorphism**: $\times$ ciphertexts $\Rightarrow$ $\times$ plaintexts
- **Security**: Secure if x_i's and y_i's indist. (quadratic residuosity).

Shortcomings of G-M, ElGamal, Paillier

- **Not fully homomorphic**
- **Not post-quantum**: Quantum kills these schemes

Quantum kills all HE schemes using abelian groups!

Thm. [Watrous] Let G be a solvable (e.g. abelian) group given by generators. There is a poly-time quantum algorithm to compute $|G|$ (with small error prob.)

The attack [Armknecht et al. '14]:

1. Encryptions of 0 (group identity) are a subgroup H of $\mathcal{C}$
2. Compute generators $h_1, \ldots, h_t$ of H by encrypting 0's
3. Challenge ciphertext $c^* \in H$ iff $|\langle h_1, \ldots, h_k \rangle| = |\langle c^*, h_1, \ldots, h_k \rangle|$

Attempt at FHE: Ring Homomorphisms

Ring Homomorphism: Map $\delta : R \to S$ (both rings):

- $\delta(r_1 + r_2) = \delta(r_1) + \delta(r_2)$, and
- $\delta(r_1 \times r_2) = \delta(r_1) \times \delta(r_2)$.

$$\begin{tikzcd} R \times R \arrow[r, "{+, \times}"] \arrow[d, "\text{component-wise } \delta"] & R \arrow[d, "\delta"] \\ S \times S \arrow[r, "{+, \times}"] & S \end{tikzcd}$$

Examples: n an integer

- Modular reduction: $\delta : \mathbb{Z} \to \mathbb{Z}/(n\mathbb{Z})$ given by $\delta(r) = r \bmod n$.
- Evaluation: $\delta : \mathbb{Z}[x] \to \mathbb{Z}$ given by $\delta(r(x)) = r(n)$.

Idea: Decryption algorithm D is a secret ring homomorphism δ_{sk}.
D will "commute" with $+$ and $\times$.

Problems with Ring Homomorphism Approach

Quantum strikes again!

- Let $\mathcal{Z} \subset \mathcal{C}$ be the encryptions of 0.
- $\mathcal{Z} = \ker(\delta)$.
- $\mathcal{Z}$ is an ideal of $\mathcal{C}$ – in particular, an additive subgroup of $\mathcal{C}$
- Quantum group-order-finding attack applies

Worse: Linear algebra (always?) breaks it

- Assume $\mathcal{C}$ presented as vector space of not-too-high dimension
- Linear algebra can distinguish $\mathcal{Z}$ and $\mathcal{C}$

But maybe the approach can be "patched"?

Early Candidate by Rivest, Adelman, Dertouzos

- **Homomorphism**: $\delta : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ (reduction modulo p, $N = pq$)

$$\begin{tikzcd}
\mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z} \arrow[r, "{+, \times}"] \arrow[d, "\text{component-wise } \delta"] & \mathbb{Z}/N\mathbb{Z} \arrow[d, "\delta"] \\
\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \arrow[r, "{+, \times}"] & \mathbb{Z}/p\mathbb{Z}
\end{tikzcd}$$

- **How to Make Secure?**
- **Possible Answer**: Hide p by using $N = p \cdot q$ with secret factorization
- **Not enough for semantic security**: $\gcd(\text{encryptions of } 0) = p$
- **One-way encryption**: Could work if plaintexts have enough min-entropy

Masking the Homomorphism with "Noise"

- **Homomorphism**: $\delta : \mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ (reduction modulo p, p odd)

$$\begin{tikzcd} \mathbb{Z} \times \mathbb{Z} \arrow[r, "{+, \times}"] \arrow[d, "\text{component-wise } \delta"] & \mathbb{Z} \arrow[d, "\delta"] \\ \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \arrow[r, "{+, \times}"] & \mathbb{Z}/p\mathbb{Z} \end{tikzcd}$$

- **How to Make Secure?**
- **Better Answer**: Hide p. Add noise / entropy to messages to defeat gcd
- **Approximate GCD Assumption**: Samples $\{x_i = q_i \cdot p + 2r_i : |r_i| \ll p\}$ are indistinguishable from random integers of the same size

Prior informal versions by Levieil-Naccache, Bram Cohen

van Dijk et al. Integer-Based HE Scheme [2010]

- **Secret key**: Large odd integer p.
- **Public key** (a la Rothblum): Generate random approx-GCD instance $\{x_i : q_i \cdot p + 2r_i : |r_i| \ll p, i \in [2k]\}$. For $i \in [k]$, set:
  1. $y_i \leftarrow x_i$ (encryptions of 0)
  2. $z_i \leftarrow x_{i+k} + 1$ (encryptions of 1)
- **Encryption** of $m \in \{0, 1\}$ (a la Rothblum):
  1. Pick random subset $S \subseteq [k]$ with $m = |S| \bmod 2$.
  2. Set $c \leftarrow \sum_{i \in \bar{S}} y_i + \sum_{i \in S} z_i$.
- **Decryption**: Set $m' = [c \bmod p] = \sum_{i \in \bar{S}} 2r_i + \sum_{i \in S} (2r_i + 1)$. Output $m = m' \bmod 2$.
- **Homomorphism**: $+, \times$ ciphertexts $\Rightarrow +, \times$ parity of mod-p values
- **Security**: Rothblum, plus approx-GCD assumption

Two Growing Problems:

1. Noise: might wrap modulo p!
2. Size of ciphertext integers.

Commutative Diagram Confusion

$$\begin{tikzcd} \mathbb{Z} \times \mathbb{Z} \arrow[r, "+,\times"] \arrow[d, "\text{component-wise } \delta"] & \mathbb{Z} \arrow[d, "\delta"] \\ \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \arrow[r, "+,\times"] & \mathbb{Z}/p\mathbb{Z} \end{tikzcd}$$

But decryption has second step: reduction mod 2!

Also decryption is incorrect if noise "wraps" mod p!

This is what we want!

(Homomorphism to correct plaintext space)

(No decryption errors)

$$\begin{tikzcd} \mathcal{C}^2 \arrow[r, "V(pk, + \text{ or } \times, \cdot, \cdot)"] \arrow[d, "D(sk, \cdot, \cdot)"] & \mathcal{C} \arrow[d, "D(sk, \cdot)"] \\ \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \arrow[r, "+,\times"] & \mathbb{Z}/2\mathbb{Z} \end{tikzcd}$$

Another Example: Polynomial Evaluation

- **Homomorphism**: Let $R = \mathbb{Z}/q\mathbb{Z}$.

  $\delta : R[\vec{x}] \to R$ given by evaluation at secret $\vec{s} = (s_1, \ldots, s_n)$.

$$\begin{tikzcd}[column sep=huge, row sep=huge]
R[\vec{x}] \times R[\vec{x}] \arrow[r, "{+, \times}"] \arrow[d, "\text{component-wise } \delta"] & R[\vec{x}] \arrow[d, "\delta"] \\
R \times R \arrow[r, "{+, \times}"] & R
\end{tikzcd}$$

Fellow-Koblitz '93: this system as a basic encryption scheme (no homomorphic ops)

- **How to Make Secure?** Hide $\vec{s}$?
- **Not enough for semantic security**:
  - Encryptions of 0 are polynomials with common root $\vec{s}$
  - Proper subspace of ciphertext polynomials.
  - Linear algebra can distinguish (if dimension of $\mathcal{C}$ is manageable)

Masking the Homomorphism with "Noise"

- **Homomorphism**: Let $R = \mathbb{Z}/q\mathbb{Z}$.
  $\delta : R[\vec{x}] \to R$ given by evaluation at secret $\vec{s} = (s_1, \ldots, s_n)$.

$$\begin{tikzcd}
R[\vec{x}] \times R[\vec{x}] \arrow[r, "{+, \times}"] \arrow[d, "\text{component-wise } \delta"] & R[\vec{x}] \arrow[d, "\delta"] \\
R \times R \arrow[r, "{+, \times}"] & R
\end{tikzcd}$$

- **How to Make Secure?** Linear algebra recovers $\vec{s}$
- **Better Answer**: Add noise / entropy to messages to linear algebra
- **Learning with Errors (LWE) Assumption** [Regev '05]:
  Linear polynomials $\{f_i(\vec{x})\}$ with $|f_i(\vec{s}) \bmod q| \ll q$ are indist. from random linear polynomials. (Linear algebra defeated.)

Brakerski-Vaikuntanathan HE Scheme (Variant)

- **Secret key**: Random $\vec{s} \in (\mathbb{Z}/q\mathbb{Z})^n$
- **Public key** (a la Rothblum): Generate random linear polys $\{f_i(\vec{x}) : e_i = f_i(\vec{s}) \bmod q, |e_i| \ll q, i \in 2k\}$. For $i \in [k]$, set:
  1. $g_i(\vec{x}) \leftarrow 2f_i(\vec{x}) \bmod q$ (encryptions of 0)
  2. $h_i(\vec{x}) \leftarrow 2f_{i+k}(\vec{x}) + 1 \bmod q$ (encryptions of 1)
- **Encryption** of $m \in \{0,1\}$ (a la Rothblum):
  1. Pick random subset $S \subseteq [k]$ with $m = |S| \bmod 2$.
  2. Set $c(\vec{x}) \leftarrow \sum_{i \in \bar{S}} g_i(\vec{x}) + \sum_{i \in S} h_i(\vec{x})$.
- **Decryption**: Set $m' = [c(\vec{s}) \bmod q] = \sum_{i \in \bar{S}} 2e_i + \sum_{i \in S} (2e_i + 1)$. Output $m = m' \bmod 2$.
- **Homomorphism**: $+, \times$ ciphertext polys $\Rightarrow +, \times$ parity of evaluations
- **Security**: Rothblum, and Learning with Errors

Two Growing Problems:
1. Noise: might wrap modulo q!
2. Degree of ciphertext polys.

Ciphertext degree problem (relinearization) [BV11]

$c_1(\vec{x})$
linear polynomial
$[c_1(\vec{s}) \bmod q] = e_1$
$e_1 = m_1 \bmod 2$

MULT

$c_2(\vec{x})$
linear polynomial
$[c_2(\vec{s}) \bmod q] = e_2$
$e_2 = m_2 \bmod 2$

$C(\vec{x}) = c_1(\vec{x}) \cdot c_2(\vec{x})$
quadratic polynomial
$[C(\vec{s}) \bmod q] = e_1 \cdot e_2$ (hopefully)
$e_1 \cdot e_2 = m_1 \cdot m_2 \bmod 2$

How to "subtract off" quadratic terms?

$C(\vec{x}) = Q(\vec{x}) + L(\vec{x}) + \text{const}$

Augment pk
$p_{j,k,t}(\vec{x}) = 2^t x_j x_k + L_{j,k,t}(\vec{x})$
"slightly quadratic" polynomials
$[p_{j,k,t}(\vec{s}) \bmod q] = 2e_{j,k,t}$ (small)

Use $p_{j,k,t}$'s to subtract off quadratic terms
$C_{linear}(\vec{x}) = C(\vec{x}) - \sum_{j,k,t} b_{j,k,t} \cdot p_{j,k,t}(\vec{x})$
$[C_{linear}(\vec{s}) \bmod q] = [C(\vec{s}) \bmod q] \bmod 2$ (hopefully)

Ciphertext noise problem (bootstrapping) [Gen09]

$$c(\vec{x})$$
$$[c(\vec{s}) \bmod q] = e$$
$$e = m \bmod 2$$
e about to wrap mod q!

Ideas?

~~Reduce noise by subtracting well-chosen encryptions of 0?~~

Well, Decryption and sk remove noise... Use them somehow?

Ciphertext noise problem (bootstrapping) [Gen09]

"Double encryption" of m

$$\{X_i = E(pk, c_i)\}$$

$c^\star$ encrypts m if $D(\cdot,\cdot) \in \mathcal{F}$, but with less noise?

$= V(pk, D(\cdot,\cdot), \{S_i\}, \{X_i\})$ where $S_i = E(pk, sk_i)$

$$c^\star \neq V(pk, D(sk,\cdot), \{X_i\})$$

Need encrypted secret key bits

$$\begin{tikzcd}
\mathcal{C}^t \arrow[r, "{V(pk, f, \cdot, \cdot)}"] \arrow[d, "{D(sk, \cdot, \cdot)}"] & \mathcal{C} \arrow[d, "{D(sk, \cdot)}"] \\
(\mathbb{Z}/2\mathbb{Z})^t \arrow[r, "f"] & \mathbb{Z}/2\mathbb{Z}
\end{tikzcd}$$

c

m

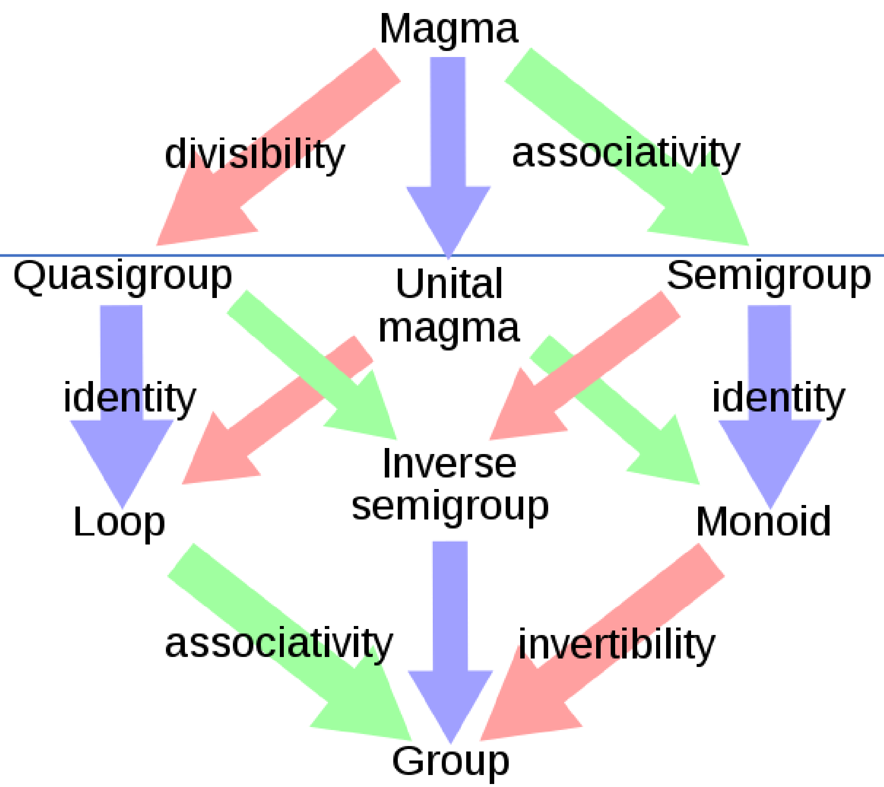
$c \in (\mathbb{Z}/2\mathbb{Z})^t$ $\quad f = D(sk, \cdot)$

$$f \in \mathcal{F}$$

Goal: "Fresh" ciphertext that encrypts m

How else can we complete diagram, considering m is unknown?

Final Thoughts on Noisy FHE



Bootstrapping (recrypting) works

We can get $D(\cdot,\cdot) \in \mathcal{F}$.

And $D \circ \mathrm{NAND}(\cdot,\cdot) \in \mathcal{F}$.

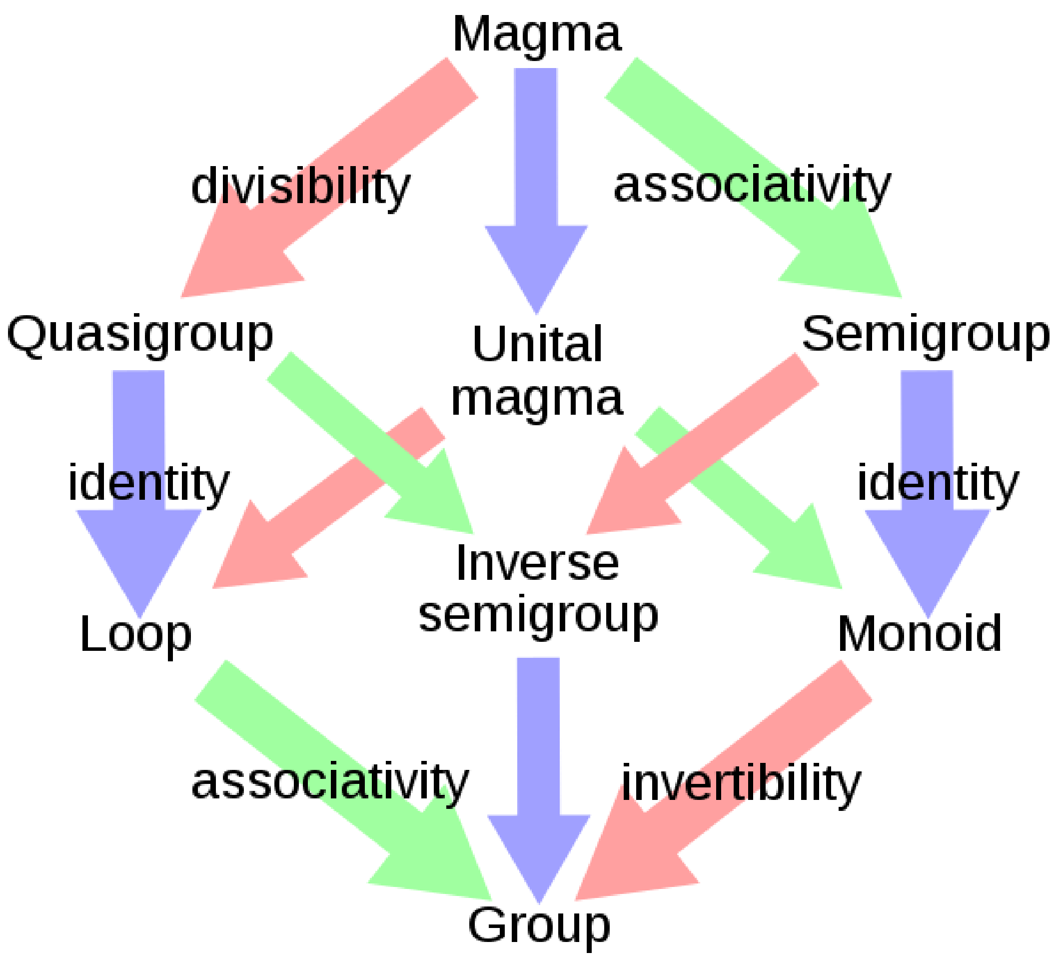
$$\begin{tikzcd}
\mathcal{C}^2 \arrow[r, "{V(pk, \mathrm{NAND}, \cdot, \cdot)}"] \arrow[d, "{D(sk, \cdot, \cdot)}"] & \mathcal{C} \arrow[d, "{D(sk, \cdot)}"] \\
(\mathbb{Z}/2\mathbb{Z})^2 \arrow[r, "\mathrm{NAND}"] & \mathbb{Z}/2\mathbb{Z}
\end{tikzcd}$$

$\mathcal{C}$ is a commutative "magma": binary relation is commutative, but not associative, etc.

$V(pk, \mathrm{NAND}, \cdot, \cdot)$ applies NAND and then recrypts.

New Homomorphisms to Explore?

Unstructured Ciphertext Space?

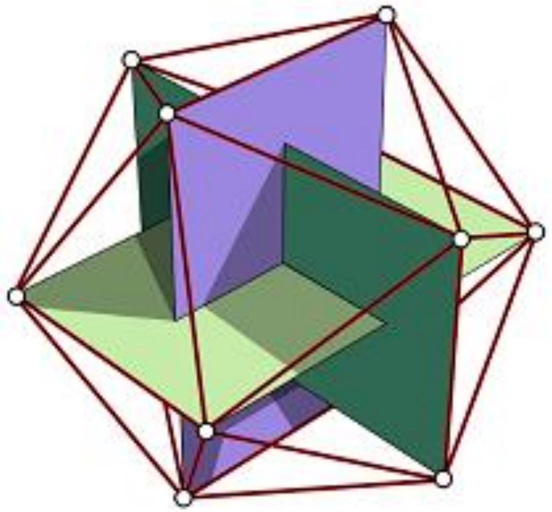


Solvable (e.g., abelian) groups

Quantum kills all HE schemes using solvable groups!

Subgroup attack [Armknecht et al. '14]: Use Watrous quantum algorithm to distinguish subgroup of ciphertexts encrypting 0.

Non-solvable groups: Why Appealing?



Barrington's Thm:

(See also Nuida, "Towards Constructing FHE without Ciphertext Noise from Group Theory".)

$H \triangleleft G$ and G non-solvable

monotone (AND,OR) circuits $\longrightarrow$ products of G elements

H elements = '0' and $G \setminus H$ elements = '1'

De Morgan's law: monotone circuits $\rightarrow$ general circuits

$ghg^{-1}h^{-1} \in H$ (AND operation)

$\langle g_1, \ldots, g_t, h_1, \ldots, h_t \rangle \in G \setminus H$ whp (OR operation)

Non-solvable groups: Why Unappealing?

Subgroup attack? Compare: Distribution of $|\langle g \rangle|$ for $g \leftarrow G$

vs.

Distribution of $|\langle h \rangle|$ for $h \leftarrow G$

***H* must also be non-solvable:** Easy to distinguish solvable vs. non-solvable groups from generators

Finite groups (almost) all "linear":

"Linear group" $\cong$ matrix group

Representation theory powerful

Braid groups broken b/c "linear"

Decryption must be non-linear:

Unlearnable (e.g., by linear algebra)

Lattice-based: non-linearity from rounding

Magmas from Multivariate Crypto?

Take a MV cryptosystem (say, HFE) $\longrightarrow$ Sprinkle in some additive homomorphism $\longrightarrow$ Might at least get semantic security (a la Rothblum)

Binary relations: Any hope for
$V(pk, \mathrm{op}, c_1, c_2) = f[f^{-1}(c_1) \text{ op } f^{-1}(c_2)]$ for non-linear f?

Conclusion

Great progress!

Future breakthroughs?

Thanks!