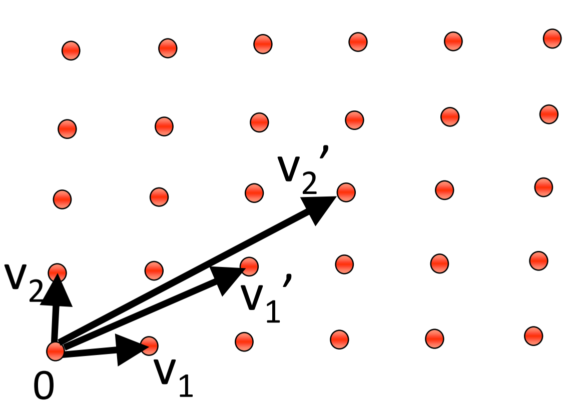
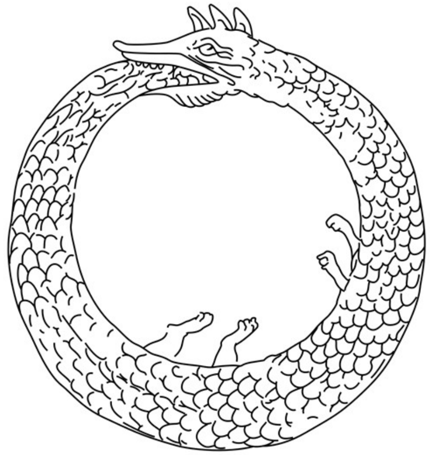
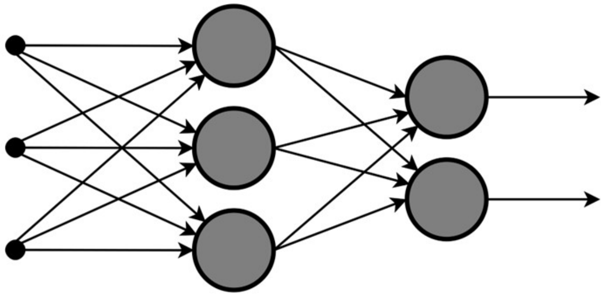
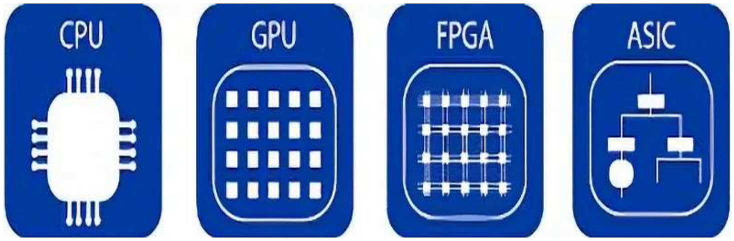




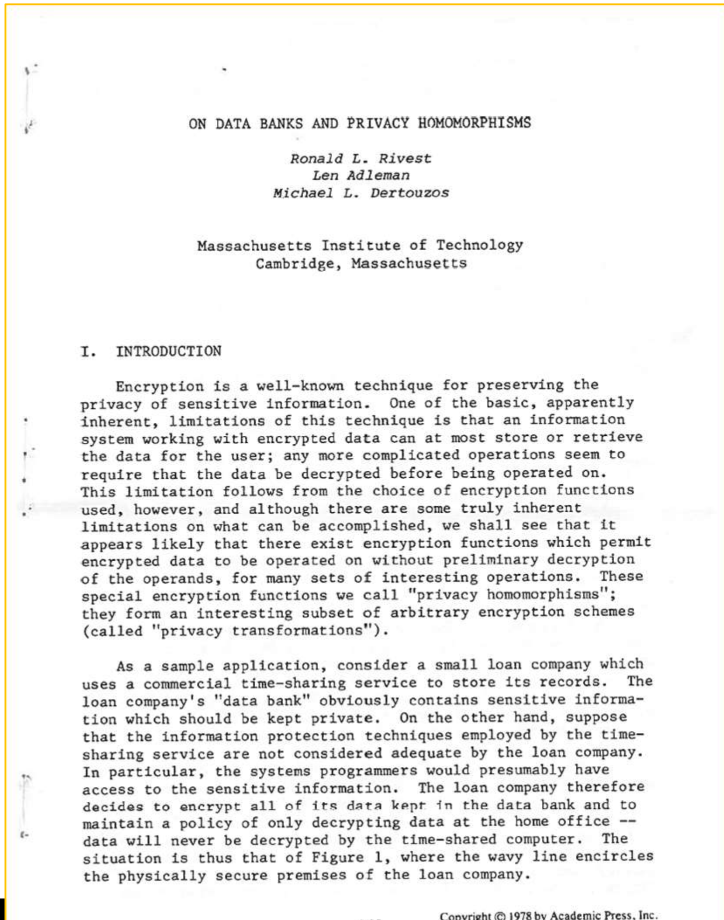

FHE: Past, Present and Future

Craig Gentry, TripleBlind

Past

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").

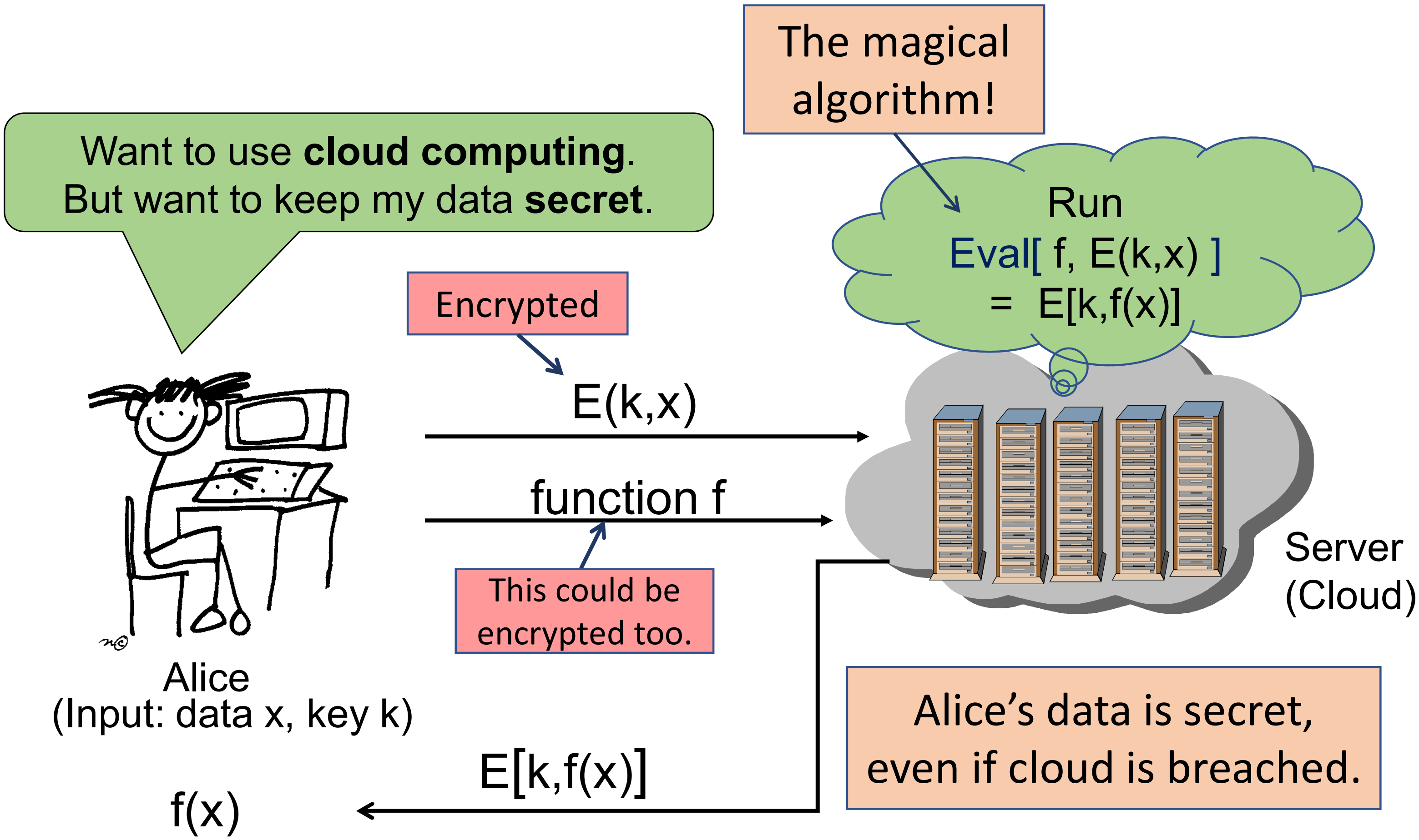
As a sample application, consider a small loan company which uses a commercial time-sharing service to store its records. The loan company's "data bank" obviously contains sensitive information which should be kept private. On the other hand, suppose that the information protection techniques employed 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.

1978: Rivest, Adleman, Dertouzos,
"On Data Banks and Privacy Homomorphisms"

Homomorphic Encryption

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

Computing on Encrypted Data



Early days of lattice-based crypto: Good for cryptanalysts!

Factoring Polynomials with Rational Coefficients

A. K. Lenstra[1], H. W. Lenstra, Jr.[2], and L. Lovász[3]

1 Mathematisch Centrum, Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands
2 Mathematisch Instituut, Universiteit van Amsterdam, Roetersstraat 15, NL-1018 WB Amsterdam, The Netherlands
3 Bolyai Institute, A. József University, Aradi vértanúk tere 1, H-6720 Szeged, Hungary

In this paper we present a polynomial-time algorithm to solve the following problem: given a non-zero polynomial $f \in \mathbb{Q}[X]$ in one variable with rational coefficients, find the decomposition of f into irreducible factors in $\mathbb{Q}[X]$. It is well

- LLL Algorithm (1982): Finds a 2^n approximation of the shortest nonzero vector in an n-dim lattice in poly(n) time
- Used to break many knapsack-based cryptosystems

Early days of lattice-based crypto: Good for cryptanalysts!

Key Recovery and Message Attacks on NTRU-Composite

Craig Gentry

Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001

Craig Gentry[1], Jakob Jonsson[2], Jacques Stern[3*], and Michael Szydlo[2]

Cryptanalysis of the Revised NTRU Signature Scheme

Craig Gentry[1] and Mike Szydlo[2]

[1] DoCoMo USA Labs, San Jose, CA, USA,
cgentry@docomolabs-usa.com
[2] RSA Laboratories, Bedford, MA, USA,
mszydlo@rsasecurity.com

Abstract. In this paper, we describe a three-stage attack against Revised NSS, an NTRU-based signature scheme proposed at the Eurocrypt

Pairing-based crypto: Good for constructions!

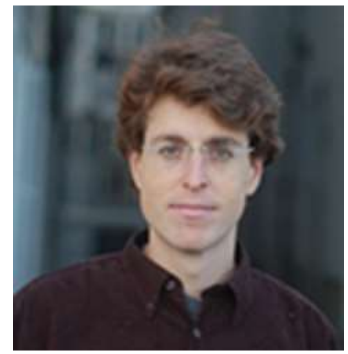
Alice Silverberg and Dan Boneh

- Boneh, Franklin (2001): Identity-based encryption (IBE), and pairing-based crypto
- **But Boneh, Silverberg on multilinear maps** (2003):

"We also give evidence that such maps might have to either come from outside the realm of algebraic geometry, or occur as 'unnatural' computable maps arising from geometry."

Lattice-based cryptography grows up

NTRU: Hoffstein, Pipher, Silverman



Regev

Ajtai, Dwork

On Lattices, Learning with Errors, Random Linear Codes, and Cryptography

Oded Regev *

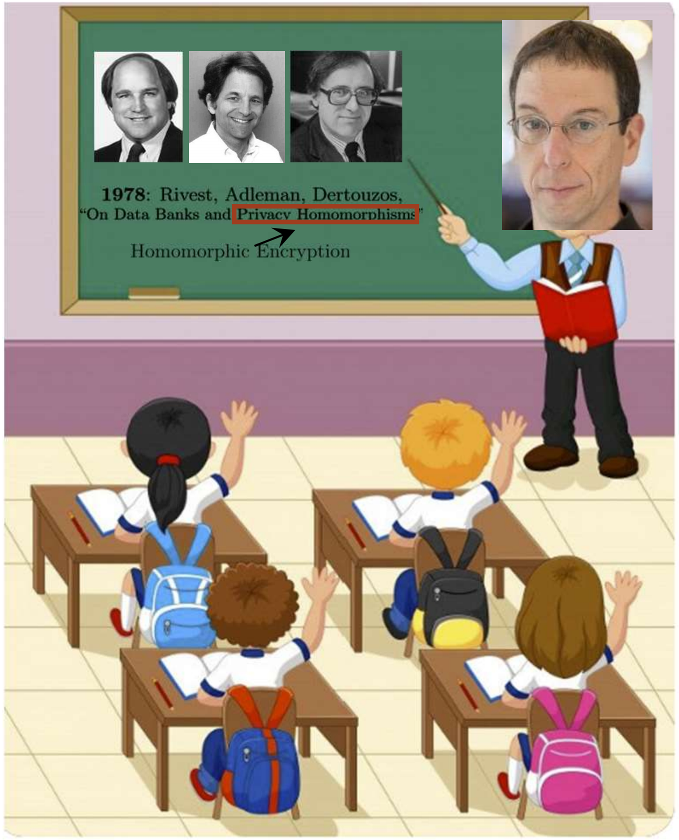
June 24, 2005

Abstract

Our main result is a reduction from worst-case lattice problems such as SVP and SIVP to a certain learning problem. This learning problem is a natural extension of the 'learning from parity with error' problem to higher moduli. It can also be viewed as the problem of decoding from a random linear code. This, we believe, gives a strong indication that these problems are hard. Our reduction, however, is quantum. Hence, an efficient solution to the learning problem implies a *quantum* algorithm for SVP and SIVP. A main open question is whether this reduction can be made classical.

A new mission, should you choose to accept it

Instant PhD for anyone that solves it!



IBE or not IBE

Stanford
Plan B Thesis

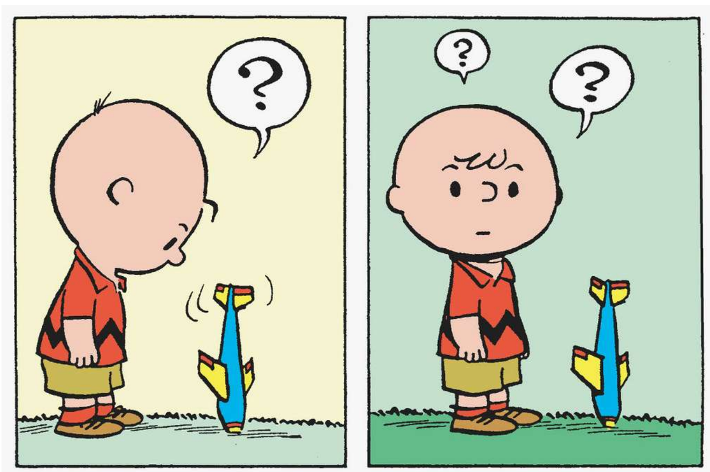
IBE Schemes
Galore!

- Including IBE from lattices (LWE) [GPV08]

Vaikuntanathan

Peikert

Rediscovering one’s ideals



FHE from lattices?

Add lattice vectors 

Multiply lattice vectors 

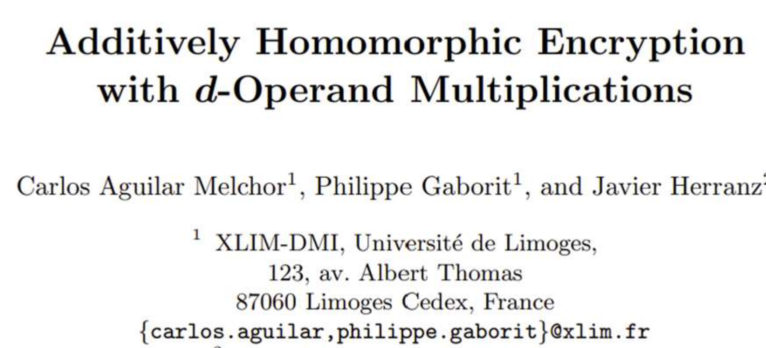
Ideal lattices! 

$(m_1+2r_1+I)\ (m_2+2r_2+I)$ is in $(m_1m_2+2r_3+I)$

 
Levieil-Naccache


Cohen

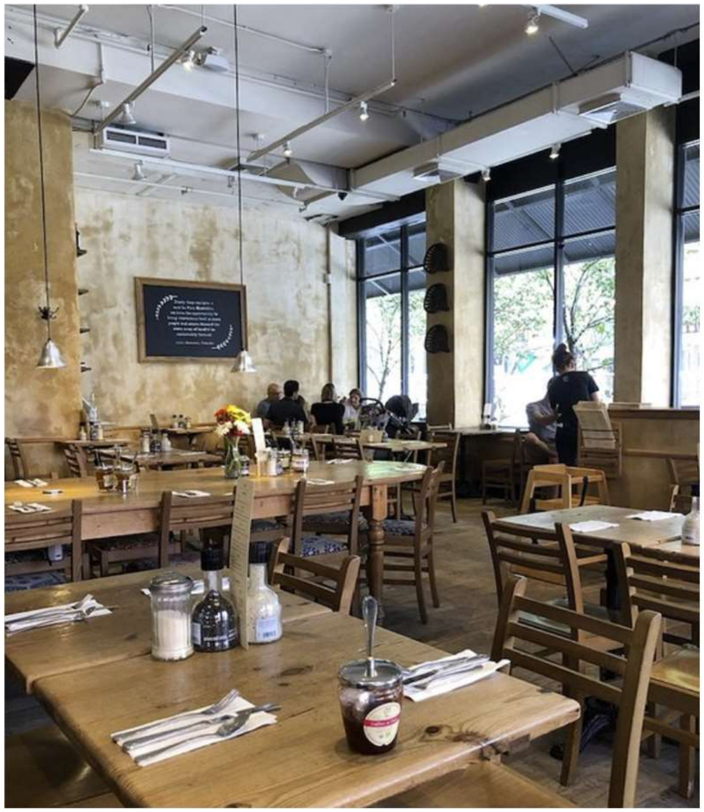

Van Dijk


Additively Homomorphic Encryption with d-Operand Multiplications

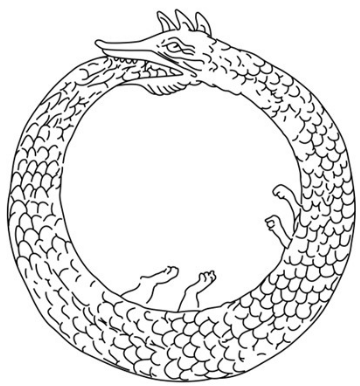
Carlos Aguilar Melchor[1], Philippe Gaborit[1], and Javier Herranz[2]

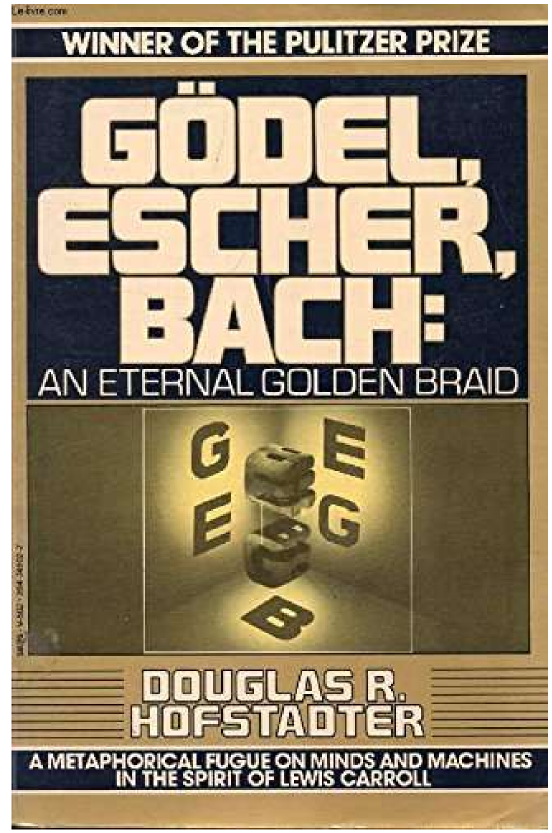
[1] XLIM-DMI, Université de Limoges,
123, av. Albert Thomas
87060 Limoges Cedex, France
{carlos.aguilar,philippe.gaborit}@xlim.fr

Bootstrapping by evaluating oneself



L E Pain QUOTIDIEN





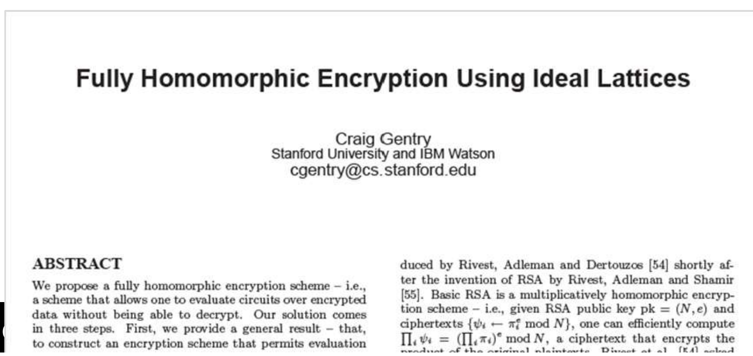
Walk through the Eval of the shadow of Dec

Must reduce the depth of
this decryption circuit… ugh!

Good enough!

A divine intervention

STOC 2009
Program Committee



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

duced 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

Shafi is my shepherd,
I shall not want

A return home to an unfamiliar place

At Dagstuhl

Where's the beef?

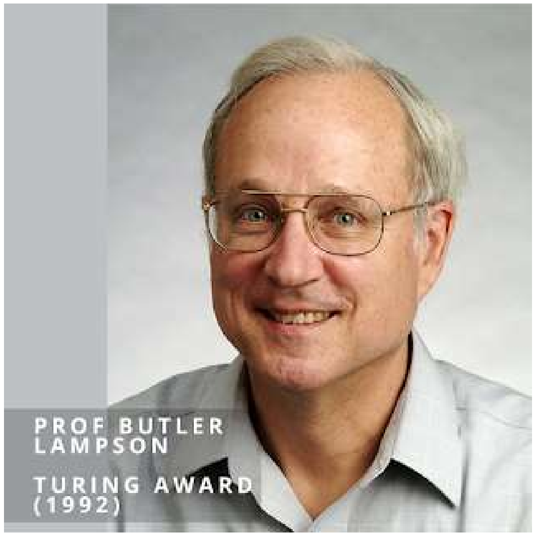
That the thing with recursion: At the end of it, you don't understand what you've done.

The ideal becomes real

- I was unsure that it all worked until **Shai Halevi** implemented it

- I was unsure it was all secure until **Zvika** and **Vinod** based FHE on LWE

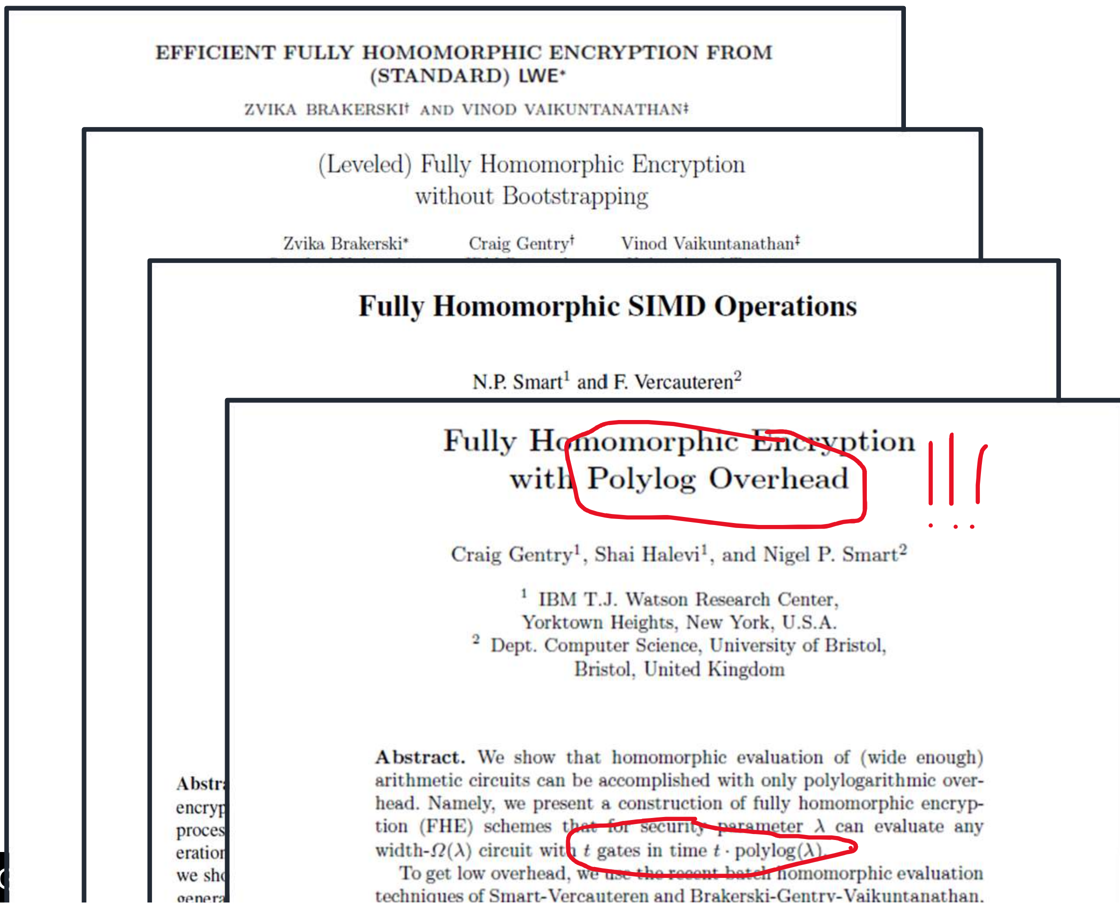
Reality check



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

He's right!

FHE: The Next Generation (BGV, BFV, GHS)



SIMD ops over 1000+ slots

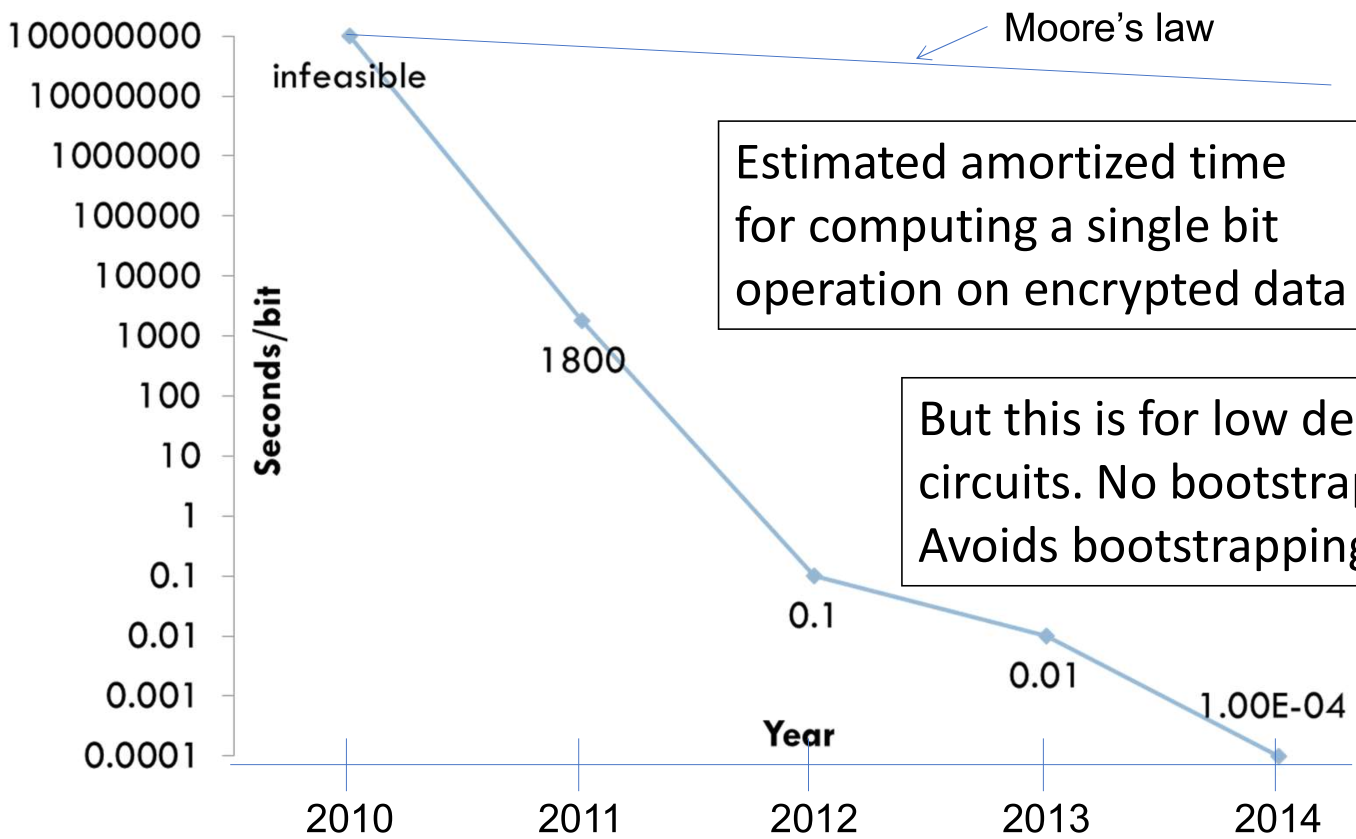
Rotation ops to align slots

Only polylog overhead!

How a theoretician thinks:

- t * poly(λ) is “efficient”
- t * polylog(λ) is AWESOME and must be PRACTICAL!

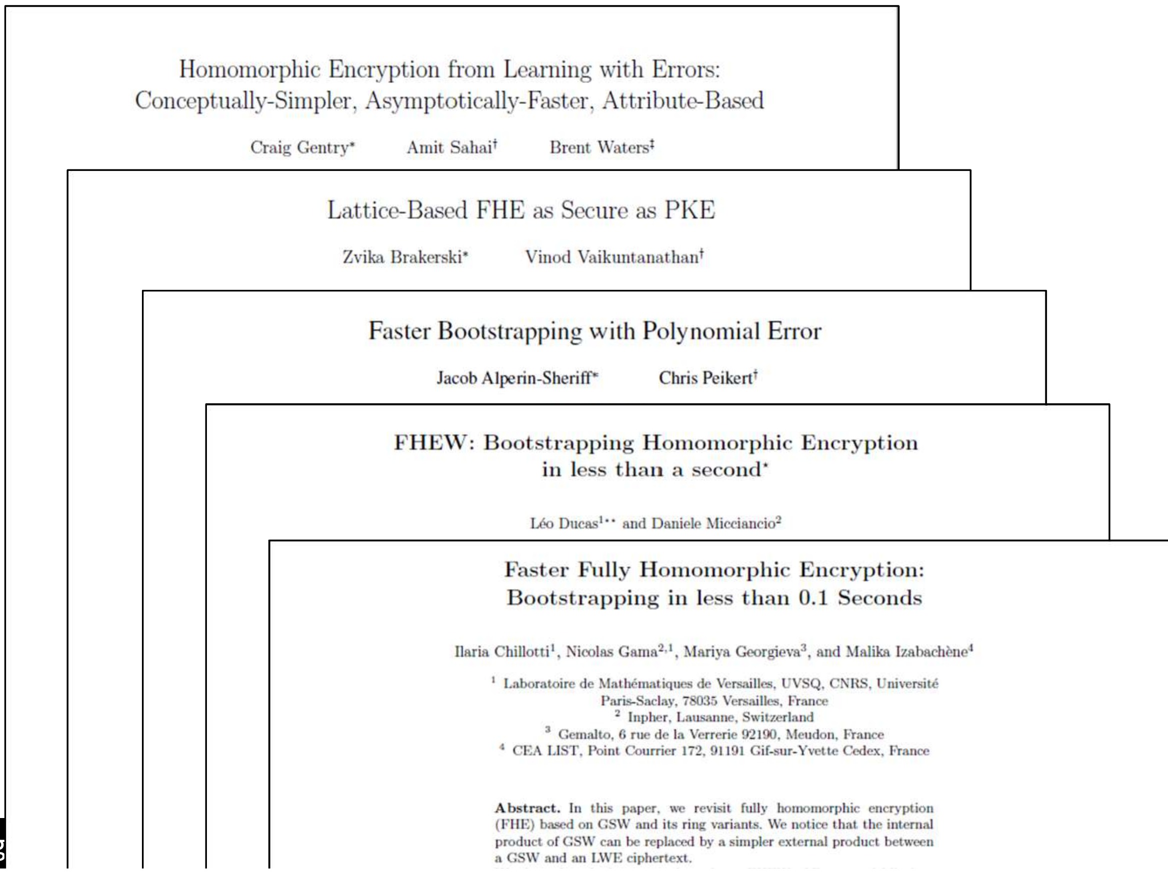
Speed of Computing on Encrypted Data on IBM's HElib Platform (1st to 2nd Gens)



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! Avoids bootstrapping like the plague.

3rd Generation FHE (FHEW, TFHE)


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!

Bootstrapping is programmable!

Stronger security!

But ciphertext packing not natively supported

4th Generation FHE (CKKS)

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-

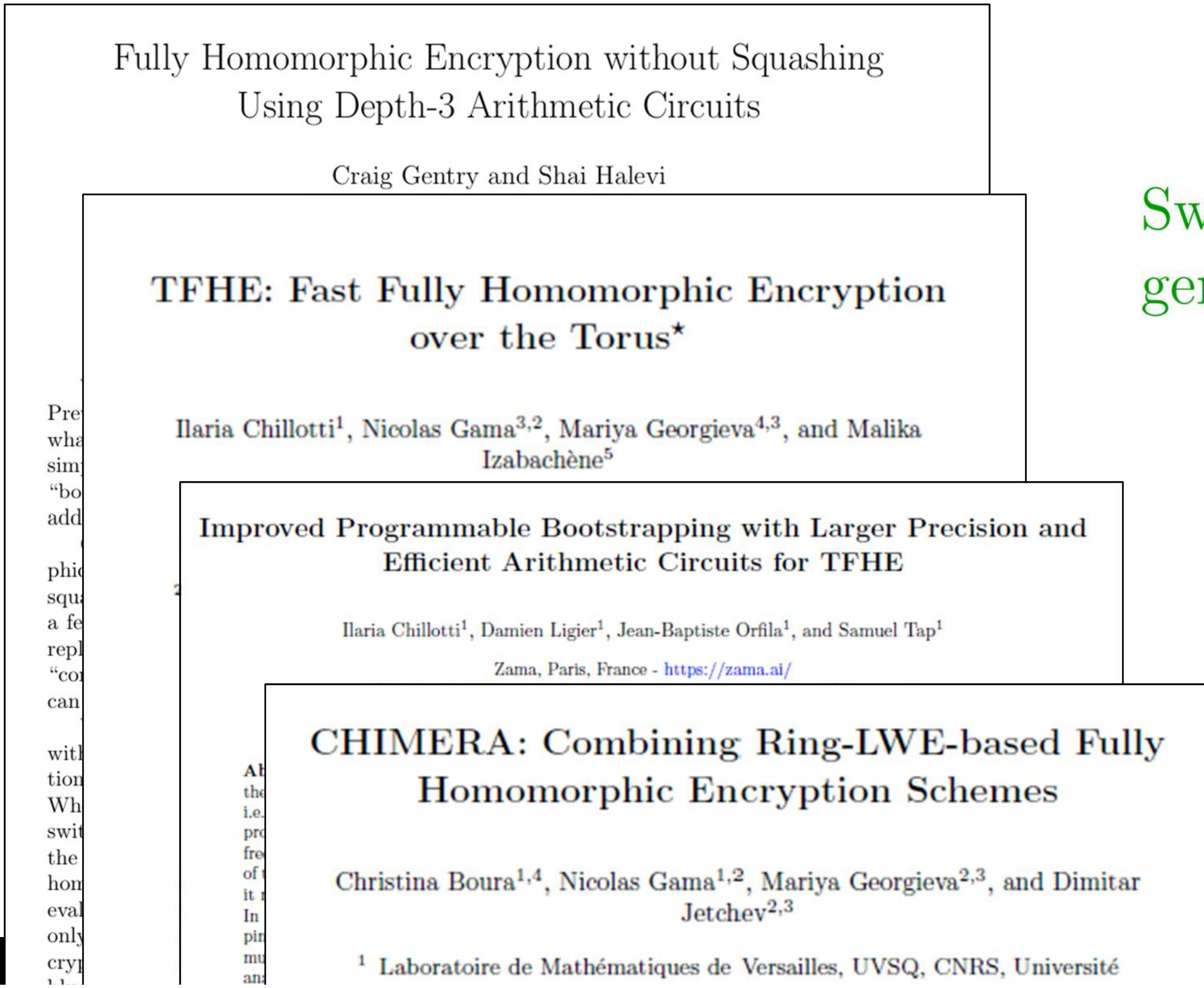
1st and 2nd Gens: mod-p numbers, arithmetic circuits

3rd Gen: bits, Boolean circuits

4th Gen: real (or complex) numbers, approximate (floating pt) arithmetic (as in neural networks)

Bootstrapping slow, but fastest when amortized over all plaintext slots

Chimeric FHE


Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits

Craig Gentry and Shai Halevi

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]

Zama, Paris, France - https://zama.ai/

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

Switch b/w 2^{nd}, 3^{rd}, and 4^{th} gen schemes as appropriate

Some lessons learned from the Past

- Be skeptical of an unproven consensus that something is impossible:
  - "Lattice-based cryptography cannot be secure"
  - "FHE is impossible"
  - "Lattice-based cryptography is post-quantum secure"
    - Yilei Chen's recent proposed quantum attack showed that we don't fully believe this.
  - "P $\neq$ NP"?
- Don't overfit on the poor performance of early schemes
  - People still mention FHE overhead of a "trillion", though now it is in the thousands
  - Lesson here about AI?
- Never underestimate the gap between theory and practice
  - Sometimes only linear is truly practical. Sometimes not even that!

Present

FHE is maturing …

FHE Libraries

- blyss - Rust FHE library specialized for private information retrieval. Includes bindings to JS & Python.
- cuFHE - CUDA-accelerated Fully Homomorphic Encryption Library.
- cuHE - GPU-accelerated HE library for NVIDIA CUDA-Enabled GPUs.
- Cupcake - Facebook's Rust library for the (additive version of the) Fan-Vercauteren scheme.
- cuYASHE - Based on leveled fully HE scheme YASHE for GPGPUs.
- fhEVM - Solidity library that enables confidential smart contracts on the Ethereum VM using FHE.
- FHEW - A Fully HE library based on *FHEW: Bootstrapping Homomorphic Encryption in less than a second*.
- FINAL - C++ FHE library based on NTRU and LWE scheme.
- FV-NFLlib - A header-only library implementing the Fan-Vercauteren scheme.
- HEAAN - Scheme with native support for fixed point approximate arithmetic.
- HEAAN-Python - Python binding for the HEANN library.
- HElib - BGV scheme with bootstrapping and the Approximate Number CKKS scheme.
- HEMat - C++ implementation of matrix computation (addition, multiplication, and transposition) using HEANN.
- krypto - C++ implementation of multivariate quadratic FHE.
- Λ ∘ λ - "Lol" Haskell library for ring-based lattice cryptography that supports FHE.
- lattigo - Go library for lattice-based crypto that implements various schemes.
- libScarab - C library implementing a FHE scheme using large integers.
- libshe - Symmetric somewhat HE library based on DGHV scheme.
- Microsoft SEAL - C++ FHE library implementing BFV and CKKS schemes.
- NFLlib - NTT-based Fast Lattice library specialized on power-of-two polynomials.
- node-seal - JavaScript/WebAssembly port of Microsoft SEAL.
- NuFHE - GPU-accelerated HE library, faster than cuFHE, that implements the tfhe algorithms.
- OpenFHE - FHE library with all features from PALISADE, merged with selected capabilities of HElib and HEAAN (all major FHE schemes).
- PALISADE - lattice encryption library (superseded by OpenFHE).
- petlib - Python library that implements a number of Privacy Enhancing Technologies.
- Pyfhel - A Python wrapper for SEAL, HElib, and PALISADE.
- python-paillier - Partially HE based on Paillier scheme.
- SEAL-python - Python binding for the Microsoft SEAL library.
- SparkFHE - Apache Spark with an add-on for FHE computations. See .
- Sunscreen - Rust compiler for the BFV fully homomorphic encryption scheme.
- TenSEAL - Library for HE operations on tensors, built on Microsoft SEAL, with a Python API.
- tfhe - Faster fully HE: Bootstrapping in less than 0.1 seconds.
- TFHE-rs - Rust implementation of the TFHE scheme for boolean and integers FHE arithmetics by Zama.

FHE Toolkits

- 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.
- Concrete - TFHE compiler for converting Python programs into FHE equivalents.
- Concrete-ML - Python-based toolkit for data scientists w/o prior FHE knowledge (using sklearn, pyTorch, XGBoost models).
- E3 - Encrypt-Everything-Everywhere framework for compiling C++ programs with encrypted operands.
- EVA - A compiler and optimizer for the CKKS scheme (targeting Microsoft SEAL).
- Google's FHE Repository - A compiler that converts a subset of C++ programs into FHE circuits implemented in various backend libraries (superseded by HEIR).
- HEIR - Google's MLIR-based toolchain for FHE compilers.
- 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.
- T2 - A cross compiler and standardized benchmarks for FHE computation that targets lattigo, HElib, PALISADE, Microsoft SEAL, and tfhe.

Source: github.com/jonaschn/awesome-he

FHE is maturing …

Communities and Standards

- fhe.org: conferences, meetups, resources around FHE
- homomorphicencryption.org: consortium of government and industry focused on standardizing FHE
- Open-source projects: OpenFHE, Palisade, Concrete, …
- iDash: hosts challenges designed to benchmark and accelerate HE for biomedical applications

Government Projects

- DPrive (DARPA): Hardware acceleration of FHE, 70M over 4 years
- NIST and EU (Prometheus) standardization of post-quantum crypto

FHE is maturing …

Some traction in blockchain?

- Zama: fheVM for confidential smart contracts
- Blyss: FHE (private information retrieval) to query the blockchain without enabling front-running

Small AI models, toward LLMs

- Zama estimates several orders of magnitude in cost reduction needed for FHE evaluation of LLMs to be reasonable: Making ChatGPT Encrypted End-to-end (zama.ai)
- Most of it to come from hardware acceleration

… But FHE is not yet mature

Wiz

- Visibility: “Single pane of glass” for security issues across cloud environments
- Compliance: Automates compliance checks / alerts
- Integration: Fits into existing workflows seamlessly, including development workflows

Datavant

- Visibility: Can *see* tokenized de-ID records
- Compliance: HIPAA expert opinion + user agreement prohibiting misuse of de-ID’d data
- Integration: Tokens and de-ID’d records allow researchers to “follow patient journey” and see patterns

Cryptography Products

- Visibility: Privacy happens “in the background”, data is hidden, EDA and troubleshooting are hard
- Compliance: Deploying crypto on sensitive data also introduces risks (TPRM)
- Integration: Big visible changes to how sensitive data is handled.

… But FHE is not yet mature

- KISS: Market for simple security / privacy solutions
  (tokenization, de-identification, federated learning, secure hardware)
  >> Market for advanced crypto solutions

- Exhaustive RAND report on securing weights of frontier models: Highlights confidential compute, HSMs, and supply chain security, but *not* FHE/SMPC, virtual HSM, or cryptographic proofs.

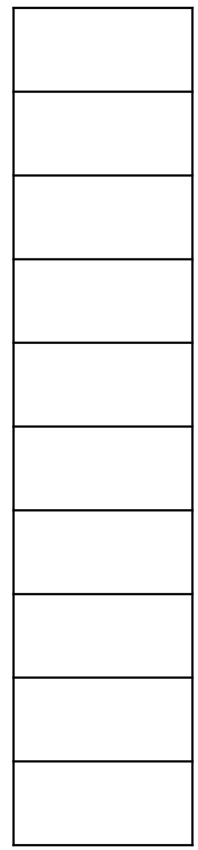
Sweet Spots for FHE? Blockchain and PIR

Private Information Retrieval

Key k, index i ∈ N

Retrieve DB[i]
without revealing i

DB ∈ $\{0,1\}^N$



Old PIR was slow

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.

Lattice-based PIR is fast

Lattice-Based Computationally-Efficient Private Information Retrieval Protocol

Carlos Aguilar-Melchor and Philippe Gaborit

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

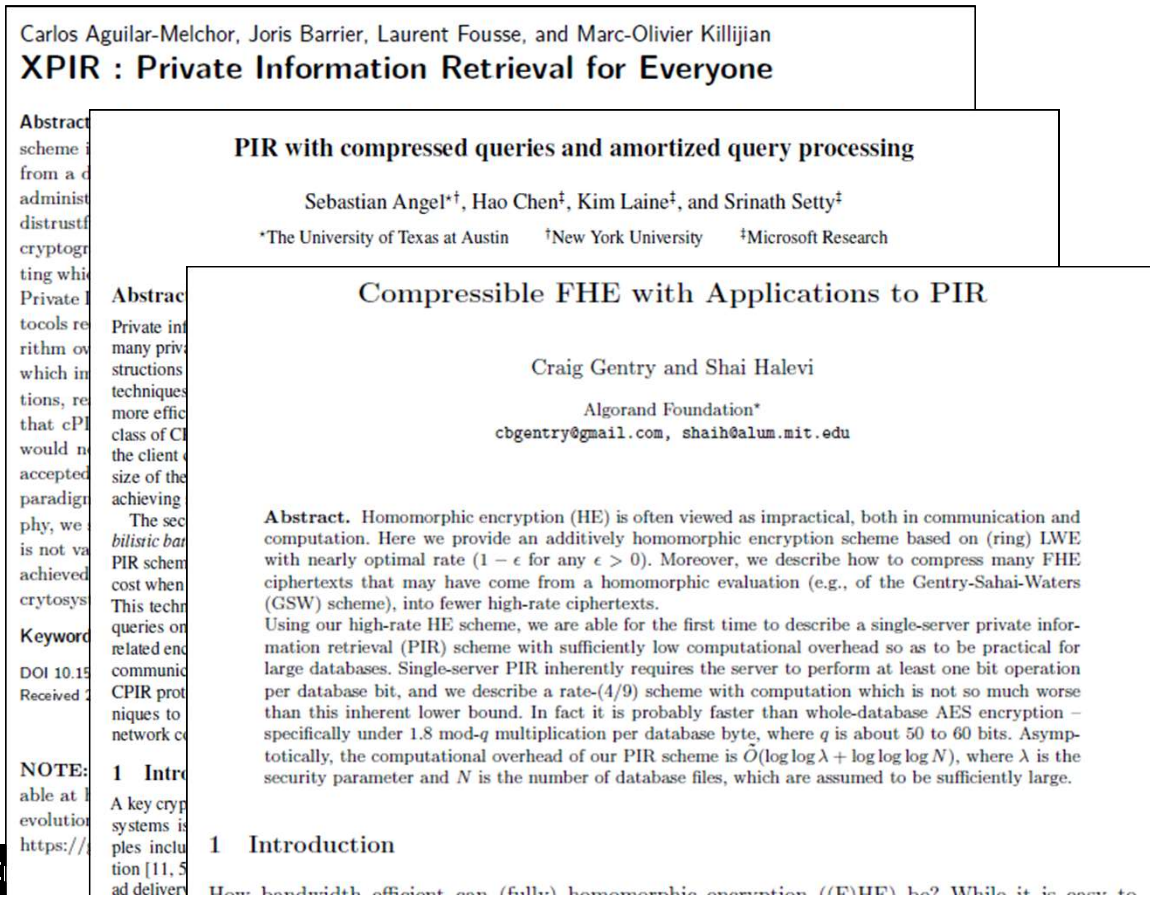
Abstract. Remote servers need search terms from the user to complete retrieval requests. However, keeping the search terms private or confidential without undermining the server's ability to retrieve the desired information is a problem that private information retrieval (PIR) schemes are designed to 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.

And faster…



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 $\approx$ AES time!

And faster…

Spiral: Fast, High-Rate Single-Server PIR via FHE Composition*

Samir Jordan Menon
Unaffiliated
menon.samir@gmail.com

David J. Wu
UT Austin
dwu4@cs.utexas.edu

Abstract

We introduce the Spiral family of single-server private information retrieval (PIR) protocols. Spiral relies on a composition of two lattice-based homomorphic encryption schemes: the Regev encryption scheme and the Gentry-Sahai-Waters encryption scheme. We introduce new ciphertext translation techniques to convert between these two schemes and in doing so, enable new trade-offs in communication and computation. Across a broad range of database configurations, the basic version of Spiral *simultaneously* achieves at least a 4.5× reduction in query size, 1.5× reduction in response size, and 2× increase in server throughput compared to previous systems. A variant of our scheme, SpiralStreamPack, is optimized for the *streaming* setting and achieves a server throughput of 1.9 GB/s for databases with over a million records (compared to 200 MB/s for previous protocols) and a rate of 0.81 (compared to 0.24 for previous protocols). For streaming large records (e.g., a private video stream), we estimate the monetary cost of SpiralStreamPack to be only 1.9× greater than that of the no-privacy baseline where the client directly downloads the desired record.

1 Introduction

A private information retrieval (PIR) [CGKS95] protocol enables a client to download an element from a public database *without* revealing to the database server which record is being requested. Beyond its direct applications to private database queries, PIR is a core building block in a wide range of privacy-preserving applications such as anonymous messaging [MOT+11, KLDF16, AS16, ACLS18], contact discovery [BDG15, DRRT18], private contact tracing [TSS+20], private navigation [FKP15, WZPM16], and safe browsing [KC21].

Private information retrieval protocols fall under two main categories: (1) multi-server protocols where the database is replicated across multiple servers [CGKS95]; and (2) single-server protocols where the database lives on a single server [KO97]. We refer to [Gas04, OS07] for excellent surveys of single-server and multi-server constructions. In many settings, multi-server constructions have reduced computational overhead compared to single-server constructions and can often achieve information-theoretic security. The drawback, however, is their reliance on having multiple *non-colluding* servers; this assumption can be challenging to realize in practice.

Conversely, single-server PIR protocols do not assume non-colluding servers. Instead, existing single-server PIR implementations have significantly higher computational costs compared to multi-server constructions. Indeed, it was believed that single-server PIR would never outperform the "trivial PIR" of simply having the client download the entire database [SC07]. While this assumption applied to earlier number-theoretic PIR schemes [KO97, CMS99, Cha04, GR05], recent lattice-based constructions [MBFK16, ACLS18, GH19, PT20, AYA+21, MCR21] have made significant strides in concrete efficiency and are much faster than the trivial PIR in many settings.

PIR over basically all of Wikipedia in a second!

See Samir Menon's fhe.org talk.

- Why PIR? No PKI, single-party product, privacy on day 1, low overhead for an HE app
- Apps: Wikipedia, private block explorers, private malware scan, …
- Real-world challenges: messy data, explainability of solution

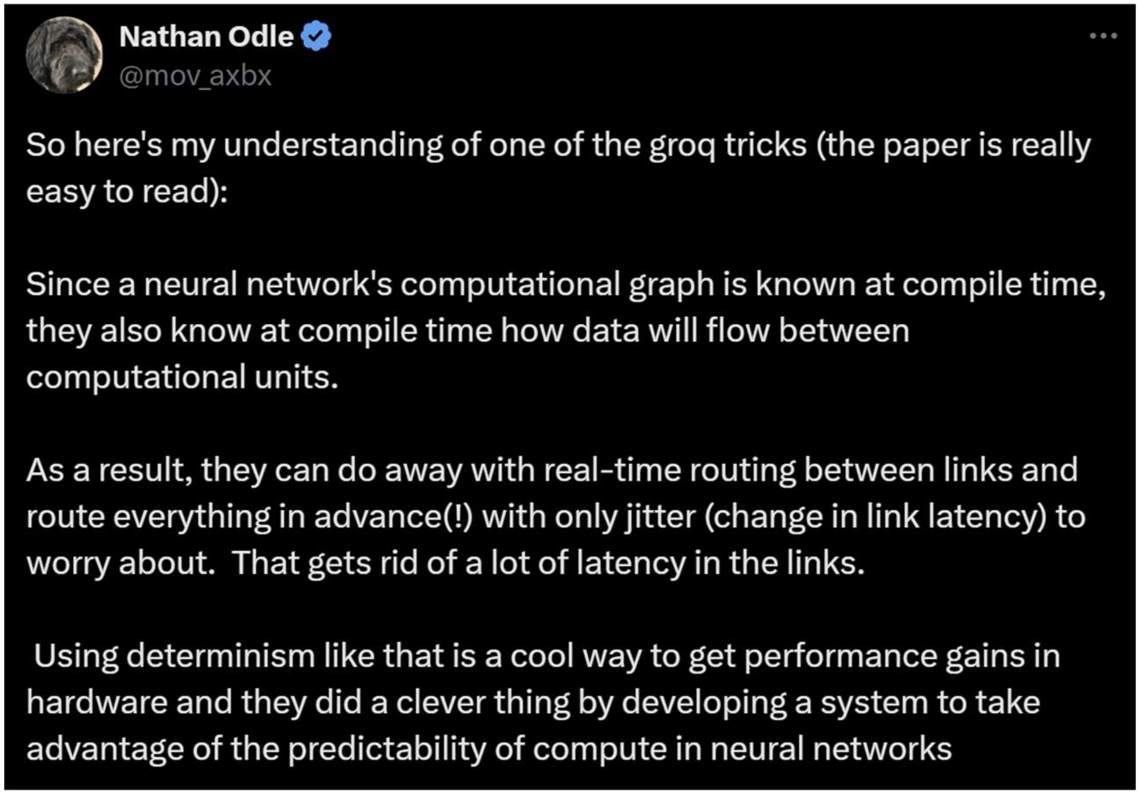
But Samir's startup Blyss now does enclaves for AI.
Draw your own conclusions!

Future

The Future of FHE

- FHE Hardware Acceleration:
  - Big boost from AI hardware, as issues are similar
- Cryptographic proofs of correct FHE Evaluation
  - Needed for FHE security and building secure AI / model ecosystem
- FHE for RAM computation
  - In the Past, I foolishly claimed it was impossible. Now it exists!

Brief Observation on Hardware Acceleration



Nathan Odle
@mov_axbx

So here's my understanding of one of the groq tricks (the paper is really easy to read):

Since a neural network's computational graph is known at compile time, they also know at compile time how data will flow between computational units.

As a result, they can do away with real-time routing between links and route everything in advance(!) with only jitter (change in link latency) to worry about. That gets rid of a lot of latency in the links.

Using determinism like that is a cool way to get performance gains in hardware and they did a clever thing by developing a system to take advantage of the predictability of compute in neural networks

FHE for RAM Computations

Private Google Queries: Possible?

Key k, query q

Enc(k, q)

Enc(k, Google(q))

About 49,200 results (0.16 seconds)

Retrieve Google(q)
without revealing q

Unproven claim: This is inherently impractical!

- FHE is in the circuit model. Google would have to take entire Internet as input for each query!
- In the real world, Google must pre-process the Internet into a data structure (inverted indices etc.) and use RAM computation to make Internet search practical.

FHE for RAM Computation is Possible

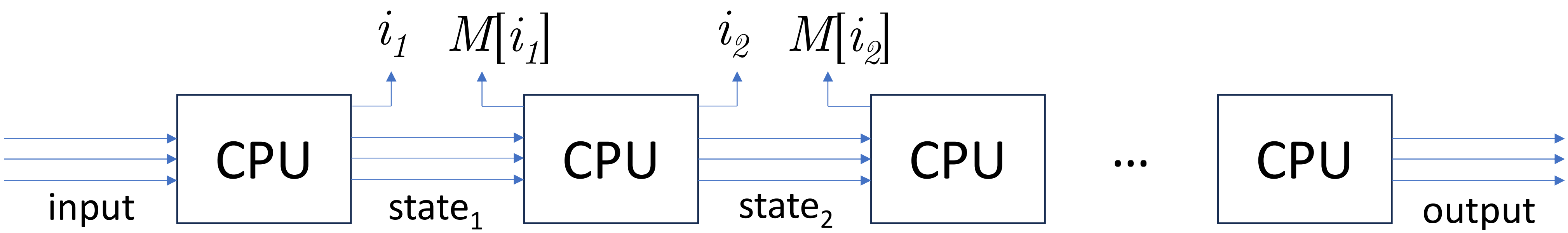
Cryptographers Solve Decades-Old Privacy Problem

7 | | SHARE

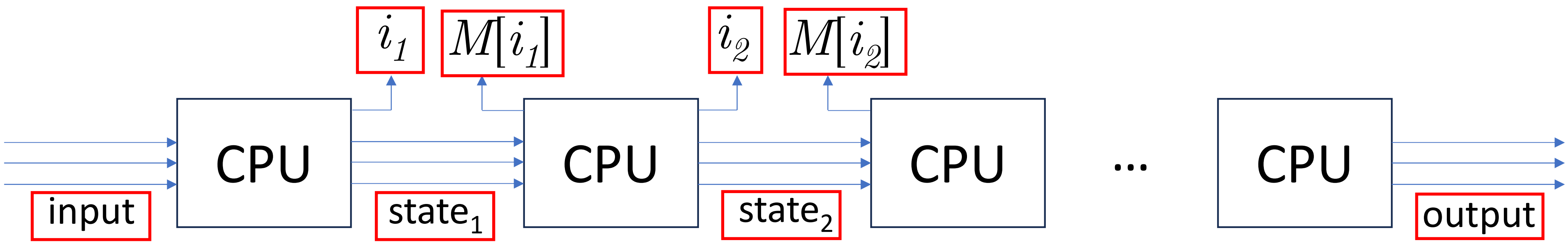
From left: Wei-Kai Lin, Ethan Mook and Daniel Wichs devised a new method for privately searching large databases.

Ian MacLellan and Khoury College of Computer Sciences/Northeastern University

Constructing RAM-FHE



Constructing RAM-FHE



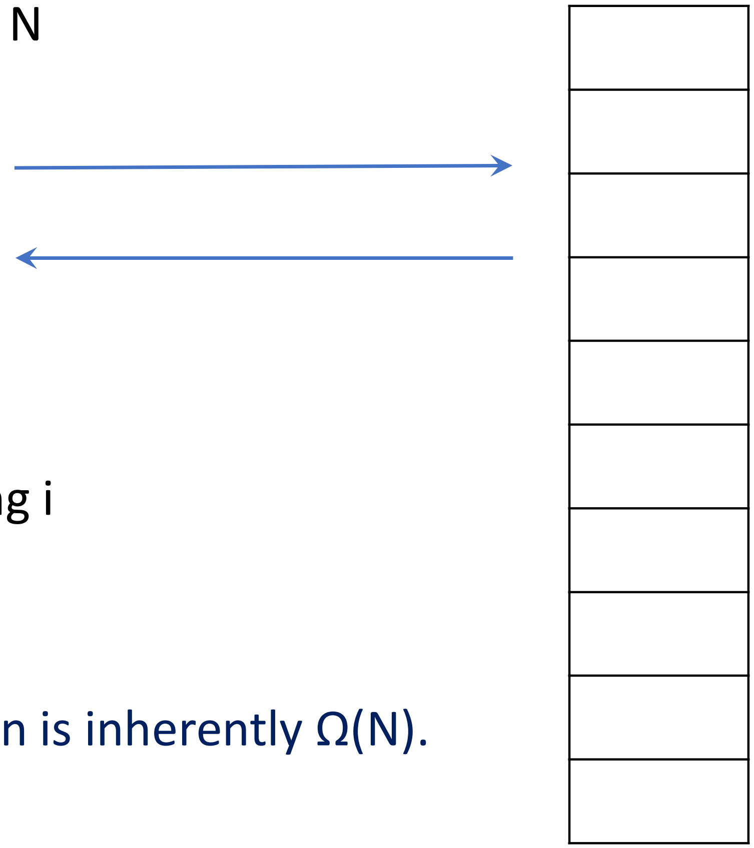
- FHE can evaluate each RAM step, encrypted
- But what about the random access from i to M[i]?
  - Private Information Retrieval! But that has complexity |M|…
  - We need RAM-PIR!

RAM-PIR: Possible?

Key k, index i ∈ N

Retrieve DB[i]
without revealing i

DB ∈ $\{0,1\}^N$



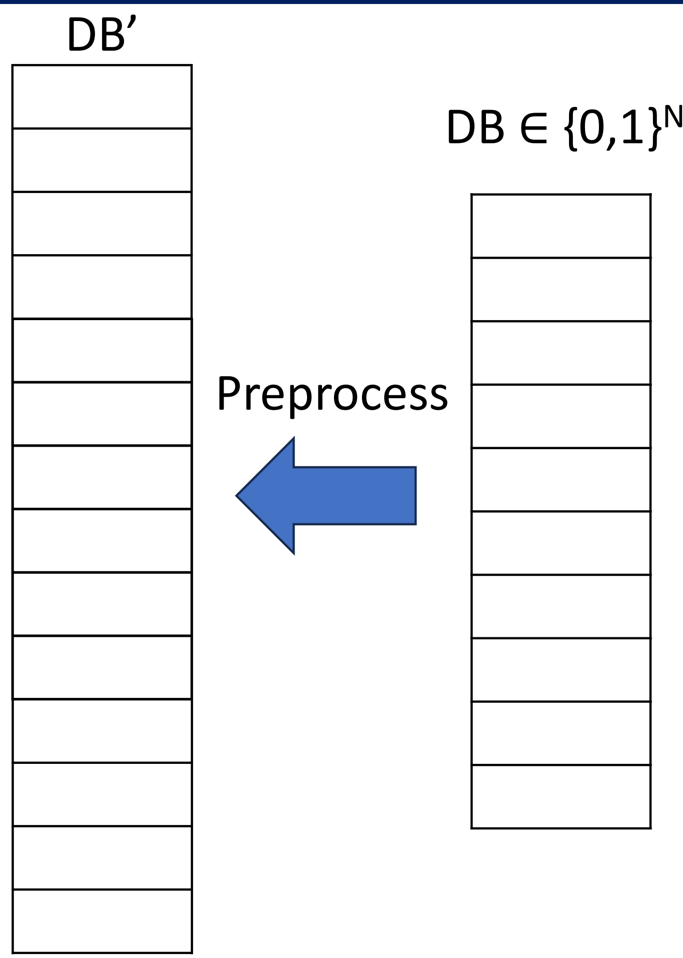
Unproven Claim:
Server computation is inherently $\Omega(N)$.

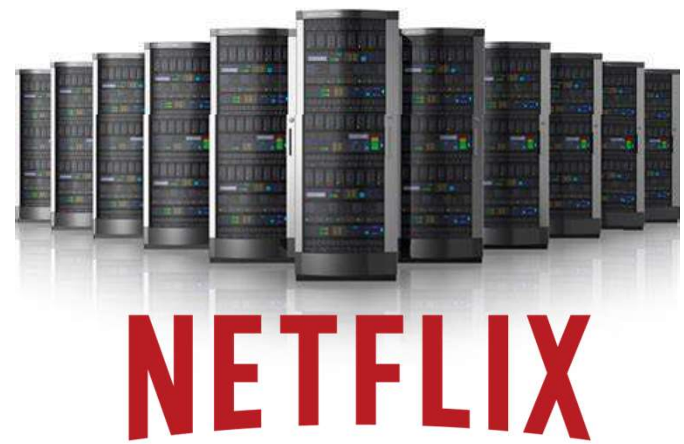
"Evidence": If server did not "read" whole DB to answer query, it would know unread part is irrelevant.

RAM-PIR Strategy

Preprocess one time, for all clients.

Preprocessing may cause expansion.

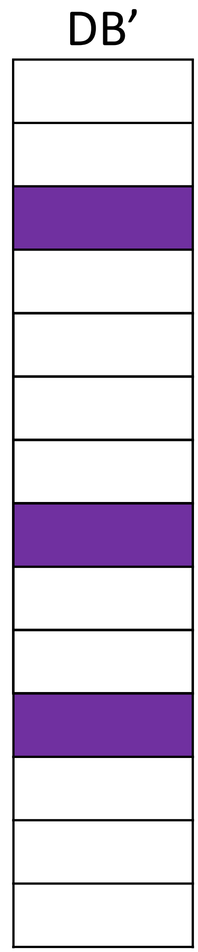




RAM-PIR Strategy

Key k, index i ∈ N

Retrieve DB[i]
without revealing i



After preprocessing, query complexity and response complexity are both low (ideally polylog(N)).

A Useful Tool? RAM Polynomial Evaluation

- Let $p(x_1, \ldots, x_m)$ be a poly of individual degree < d, hence $N = d^m$ monomials.
- How fast can we evaluate $p(x_1, \ldots, x_m)$ mod q for any x in $\{1, \ldots, q\}^m$?
  - No preprocessing of p's coefficients: In general, $\Omega(N)$ time just to "read" p.
  - Preprocessing:
    - Duh! – Just compute and store $p(x_1, \ldots, x_m)$ for all $(x_1, \ldots, x_m)$ in $\{1, \ldots, q\}^m$.
      - So, for about q^m preprocessing time, you get random access evaluations!
    - BUT: What if q is huge – e.g., $q \approx 2^N$?
    - If q is smooth – i.e., $q = q_1 \cdot q_2 \cdot \ldots \cdot q_t$ for small q_i's that are at most d – just use CRT.
    - BUT: What if q is not smooth?

Kedlaya-Umans '08: Let R be a ring of cardinality q – e.g., $R_t = Z_t[y]/(f(y))$. Let $p(x_1, \ldots, x_m) \in R[x_1, \ldots, x_m]$ be a polynomial of individual degree d. Let $N = d^m$. For any $\varepsilon > 0$, for any sufficiently large N, one can preprocess p into a data structure of size at most $N^{1+\varepsilon} \log^{1+o(1)} q$ that allows random access evaluation of $p(\alpha)$ for any $\alpha \in R^m$ in time polylog(N) $\log^{1+o(1)} q$.

RAM-PIR from RAM Polynomial Evaluation

Key k, index $i \in N$ as $(i_1, ..., i_m)$

$\{c_j = Enc(k, i_j)\}$

$c = Eval(p_{DB}, c_1, ..., c_m)$

$c = Enc(k, DB[i])$

Retrieve DB[i]
without revealing i

THE TRICK! Use old-timey algebraic SHE where $Eval(p_{DB}, c_1, ..., c_m)$ equals $p_{DB}(c_1, ..., c_m)$!!

Noise growth: Exp in total degree dm (smallish)

DB'

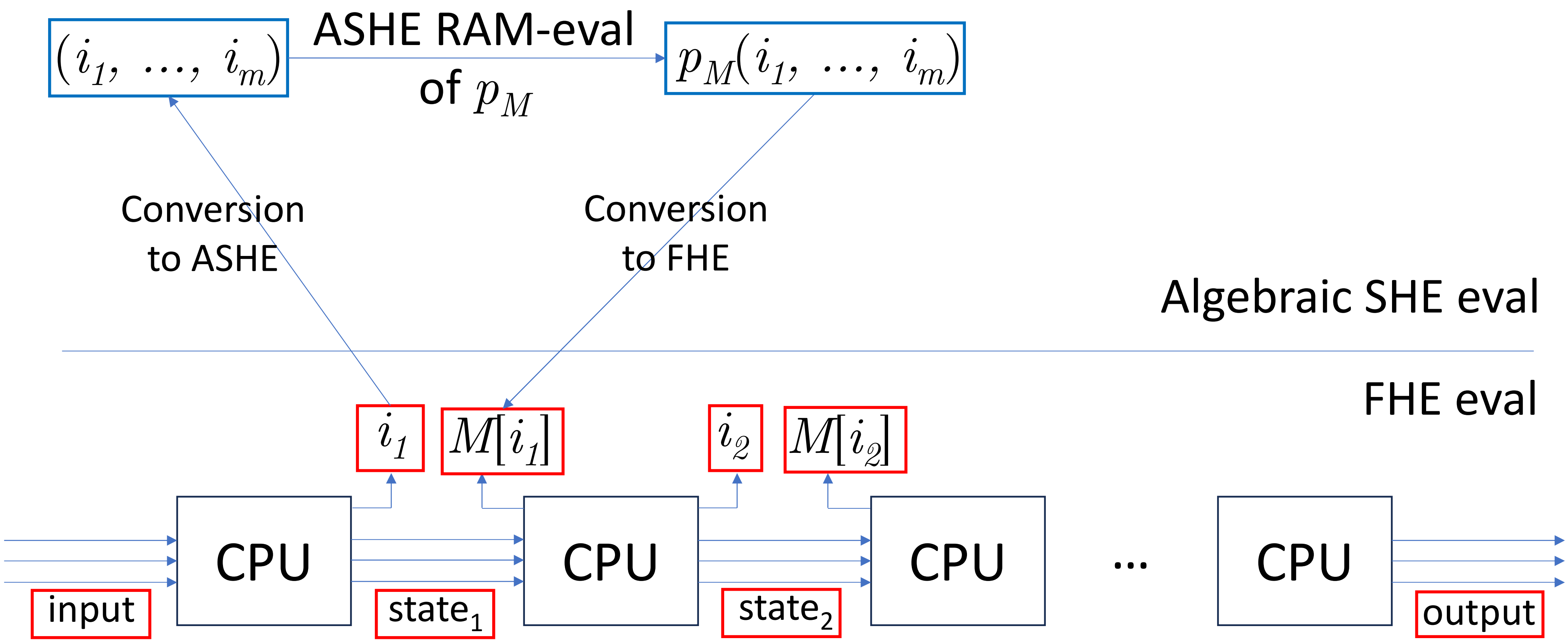
$DB \in \{0,1\}^N$

NETFLIX

$p_{DB}(x_1, ..., x_m)$ of ind. degree d with $p_{DB}[i_1, ..., i_m] = DB[i_1, ..., i_m]$ over our HE's plaintext ring.

DB' is p_{DB}'s data structure for our HE's ciphertext ring.

RAM-FHE: Putting it all together



RAM-FHE Loose Ends

- Aspects I didn't cover:
  - How Lin-Mook-Wichs handle memory updates
  - How exactly Kedlaya-Umans does RAM polynomial evaluation

- Open Questions:
  - Base security on LWE rather than RLWE
  - Improve efficiency
  - Other crypto applications of Kedlaya-Umans:
    - Polynomial commitment schemes used in SNARKs

Thank you!