Exploring Fully-Homomorphic Encryption

Alex Grabanski

4/8/2017

What is fully-homomorphic encryption?

- A way to perform computations on data without knowing what the data is.

What is fully-homomorphic encryption?

- A way to perform computations on data without knowing what the data is.
- What computations?

What is fully-homomorphic encryption?

- A way to perform computations on data without knowing what the data is.
- What computations?
- The largest possible class of computations for which we could hope to assure the security of all inputs and intermediate results.

Models of computation

- In the 1930's, Gödel, Church and Turing attempted to define computation [10]

Models of computation

- In the 1930's, Gödel, Church and Turing attempted to define computation [10]
- Result: General Recursive Functions, Lambda Calculus, and Turing Machines equivalent in power!

Models of computation

- In the 1930's, Gödel, Church and Turing attempted to define computation [10]
- Result: General Recursive Functions, Lambda Calculus, and Turing Machines equivalent in power!
- Church-Turing Thesis: There are no more powerful notions of an "effective procedure" than using one of the above

Models of computation

- In the 1930's, Gödel, Church and Turing attempted to define computation [10]
- Result: General Recursive Functions, Lambda Calculus, and Turing Machines equivalent in power!
- Church-Turing Thesis: There are no more powerful notions of an "effective procedure" than using one of the above
- Limitations of these systems: Halting Problem – determine if a program halts, given its source code

Models of computation

- In the 1930's, Gödel, Church and Turing attempted to define computation [10]
- Result: General Recursive Functions, Lambda Calculus, and Turing Machines equivalent in power!
- Church-Turing Thesis: There are no more powerful notions of an ”effective procedure” than using one of the above
- Limitations of these systems: Halting Problem – determine if a program halts, given its source code
- Undecidable!

Allowable Models of Computation: Part I

- General Computation is too powerful

Allowable Models of Computation: Part I

- General Computation is too powerful
- Vulnerability: Side-Channel Timing Attacks (an entropy leak!)

Allowable Models of Computation: Part I

- General Computation is too powerful
- Vulnerability: Side-Channel Timing Attacks (an entropy leak!)
- Impossible to avoid in general – Halting Problem!

Allowable Models of Computation: Part I

- General Computation is too powerful
- Vulnerability: Side-Channel Timing Attacks (an entropy leak!)
- Impossible to avoid in general – Halting Problem!
- So, restrict to computations which take a fixed amount of time.

Allowable Models of Computation: Part II

- If we allow arbitrary-size inputs outputs, entropy would leak from ciphertext sizes

Allowable Models of Computation: Part II

- If we allow arbitrary-size inputs outputs, entropy would leak from ciphertext sizes
- So we have fixed time, fixed I/O size operations

Allowable Models of Computation: Part II

- If we allow arbitrary-size inputs outputs, entropy would leak from ciphertext sizes
- So we have fixed time, fixed I/O size operations
- Exactly the class of functions computable by Boolean circuits!

Representing Boolean Circuits using $\mathbb{Z}_2[X_1, ...X_n]$

- Observation: If we're in the ring $\mathbb{Z}_2$:
- $a + 1$ computes "NOT a"
- $a \times b$ computes "a AND b"
- These form a *universal set* of logic gates
- Allows expressing a boolean circuit with a single bit output as a polynomial in $\mathbb{Z}_2[X_1, ...X_n]$.
- Example: $(a + 1)(b + 1) + 1 = a + b + ab$
- computes "a OR b through the *Evaluation Homomorphism* at (a, b) : $\mathbb{Z}_2[X_1, ...X_n] - > \mathbb{Z}_2$"

Cryptosystems and Homomorphic Properties

- 1978 – Rivest et. al developed RSA cryptosystem, based on impracticality of factoring large primes
- Ciphertexts are x^e for e in the public key, x the plaintext
- Homomorphic property: Multiplication of ciphertexts
- $x^e * y^e = (x * y)^e$
- Question (Rivest et. al): "[is it] possible to have a privacy homomorphism with a large set of operations which is highly secure? [8]

Cryptosystems with Homomorphic Properties

- Boneh-Goh-Nassim (BGN) cryptosystem – capable of evaluating arbitrary quadratic forms [2]
- Pallier, Benaloh cryptosystems – capable of evaluating sums [7], used for secure voting.
- Possible to securely evaluate an arbitrary number of additions, multiplications?
- Problem: Apparent three-way trade-off between ”niceness” of structures, security, and number of homomorphic properties

Gentry, 2009: Fully Homomorphic Encryption using Ideal Lattices

- Submitted as a PhD thesis under the advisement of Boneh (of the BGN cryptosystem)
- Made possible by a novel technique: Bootstrapping
- Abandon purely-algebraic approach, instead, assume an ”error signal” in ciphertexts grow over operations
- Occasionally perform a special operation on ciphertexts to reduce the ”error signal”
- Call this operation *Recrypt*.

Abstract Definition of the Cryptosystem

$$\text{KeyGen}_\epsilon : \{0,1\}^* \times \mathbb{N} \to \mathcal{K} \times \mathcal{K}$$

$$\text{Encrypt}_\epsilon : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$$

$$\text{Decrypt}_\epsilon : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$$

$$\text{Evaluate}_\epsilon : \mathcal{K} \times \mathfrak{C}_\epsilon \times \mathcal{C}^n \to \mathcal{C}$$

where $\mathcal{K}$ is the key-space, $\mathcal{C}$ is cipher-space, $\mathcal{P}$ is plaintext-space, and $\mathfrak{C}$ is the space of all ”circuits” (may be viewed as tuples of multivariate polynomials).

- Second argument to KeyGen_ϵ is λ, the *security parameter* of the scheme

Correctness Condition for Evaluation

$$\forall R \in \{0,1\}^*, \lambda \in \mathbb{N}, \quad \text{if} \quad (pk, sk) = \text{KeyGen}_\epsilon(R, \lambda),$$
$$\text{then} \quad \forall C \in \mathfrak{C}_\epsilon, \quad \pi_1, ...\pi_n \in \mathcal{P} \quad \text{with} \quad \psi_i = \text{Encrypt}_\epsilon(pk, \pi_i),$$
$$\text{Decrypt}_\epsilon(sk, \text{Evaluate}_\epsilon(pk, C, (\psi_1, ...\psi_n))) = C(\pi_1, ...\pi_n)$$

Correctness Condition for Evaluation

$$\forall R \in \{0,1\}^*, \lambda \in \mathbb{N}, \quad \text{if} \quad (pk, sk) = \text{KeyGen}_\epsilon(R, \lambda),$$

$$\text{then} \quad \forall C \in \mathfrak{C}_\epsilon, \quad \pi_1, ...\pi_n \in \mathcal{P} \quad \text{with} \quad \psi_i = \text{Encrypt}_\epsilon(pk, \pi_i),$$

$$\text{Decrypt}_\epsilon(sk, \text{Evaluate}_\epsilon(pk, C, (\psi_1, ...\psi_n))) = C(\pi_1, ...\pi_n)$$

- Fails to rule out a trivial definition of Evaluate in favor of a definition of Decrypt which performs elaborate computations!

Correctness Condition for Evaluation

$$\forall R \in \{0,1\}^*, \lambda \in \mathbb{N}, \quad \text{if} \quad (pk, sk) = \text{KeyGen}_\epsilon(R, \lambda),$$
$$\text{then} \quad \forall C \in \mathfrak{C}_\epsilon, \quad \pi_1, ...\pi_n \in \mathcal{P} \quad \text{with} \quad \psi_i = \text{Encrypt}_\epsilon(pk, \pi_i),$$
$$\text{Decrypt}_\epsilon(sk, \text{Evaluate}_\epsilon(pk, C, (\psi_1, ...\psi_n))) = C(\pi_1, ...\pi_n)$$

- Fails to rule out a trivial definition of Evaluate in favor of a definition of Decrypt which performs elaborate computations!
- Solution: Require that the decryption operation be representable as a circuit $\mathcal{D}_\epsilon$ of size polynomial in λ
- Under this requirement, the trivial definition would fail for large-enough circuits.

Secret Sauce: Recrypt_ϵ

$\text{Recrypt}_\epsilon : \mathcal{K} \times \mathfrak{C}_\epsilon \times \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, defined as:

$\text{Recrypt}_\epsilon(pk, \mathcal{D}_\epsilon, esk, \psi) = \text{Evaluate}_\epsilon(pk, \mathcal{D}_\epsilon, (esk, \text{Encrypt}_\epsilon(pk, \psi)))$

where *esk* is a ciphertext *encrypting the secret key sk*.

- *esk* is used by $\mathcal{D}_\epsilon$ to remove the inner encryption on a double-encryption of a plaintext.
- Homomorphically evaluated, so plaintext never visible to the outside world.
- Note: Requires that *esk* doesn't give us practical knowledge about *sk*!

Application of Recrypt: Proxy Re-Encryption

- Given a plaintext encrypted under $pk1$ and $esk1$, output the same plaintext encrypted under $pk2$.
- Intuitively: Allows Alice to delegate handling of a secret message addressed to her to Derek.
- Does not reveal Alice's secret key.
- Useful as a primitive in multi-agent cryptosystems.
- Possible using slightly-modified definition of Recrypt_ϵ to encrypt with $pk2$.

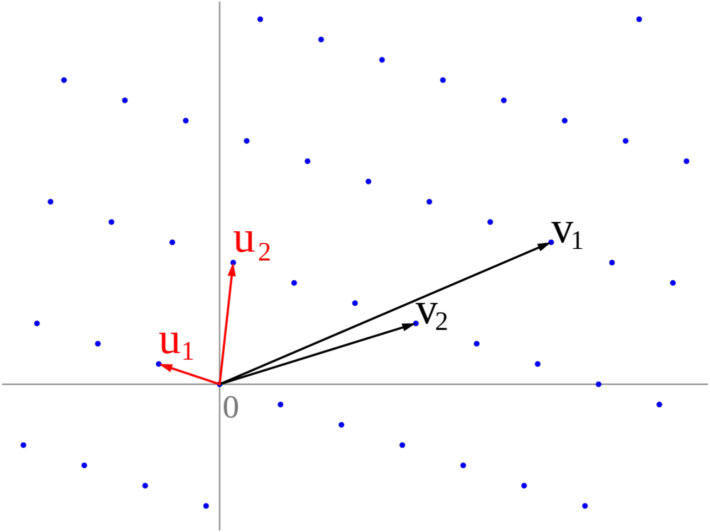
Applications of FHE

- Analysis of Genome databases without revealing participants' sequences [5]
- In general, statistical analyses on sensitive user data [6]
- Truly blind blind auctions [4]
- Search engines which *don't* know users' search queries
- Gives hope for a future of cloud computing which respects users' data privacy.

Implementing the Scheme: Lattices

- *Lattice*: a copy L of $\mathbb{Z}^n$ living in $\mathbb{R}^n$ (spanning subgroup under addition) [9]
- *Lattice Basis*: A collection of n vectors B whose span (with coefficients in $\mathbb{Z}$ is L.
- Hard problem on lattices: Given a lattice basis B for L, compute a new lattice basis B' which is also a basis for L, but with the shortest possible vectors.
- Called the *Shortest Independent Vector Problem* (SIVP), a close relative to the *Closest Vector Problem* (CVP)
- CVP known to be NP-Complete by reduction to the subset-sum problem.

Sample SIVP Instance



[0]Public Domain Image by User:Catslash on WikiMedia

Multi-dimensional modular arithmetic

- Given a lattice basis B for L, let $\mathcal{P}(B)$ be the *fundamental parallelpiped* of B.
- $\mathcal{P}(B)$ is the parallelpiped spanned by vectors in B translated to be centered on the origin.
- For any vector $v \in \mathbb{R}^n$, define $v \bmod B$ to be the vector(s) in $\{v + \sum_i a_i\vec{b}_i \,|\, \forall i \quad a_i \in \mathbb{Z} \wedge \vec{b}_i \in B\} \cap \mathcal{P}(B)$.
- Computation: $v \bmod B = v - \boldsymbol{B} * [\boldsymbol{B}^{-1} v]$, where [.] represents "round to the nearest integer vector".

Implementing the Scheme: Ideal Lattices

- Consider the ring $\mathcal{R} = \mathbb{Z}[x]/f(x)$ with $deg(f) = n$
- Polynomials of degree $< n$ with integer coefficients identifiable with vectors in $\mathbb{Z}^n$, a lattice!
- If I is an ideal of $\mathcal{R}$, by definition it's a subgroup under $+$ which is closed under multiplication by elements of $\mathcal{R}$.
- We can view I as a sub-lattice of $\mathbb{Z}^n$, called $\mathcal{L}(I)$.
- Such a lattice is called an *Ideal Lattice*.
- We restrict our attention to *Circulant Ideal Lattices*, which is an ideal lattice where $\mathcal{R} = \mathbb{Z}[x]/(x^n - 1)$

Operations in Ideal Lattices

- Represent the polynomial $a_{n-1}x^{n-1} + ...a_1x + a_0$ by the vector $(a_{n-1}...a_1a_0)^T$
- Addition of polynomials $\longleftrightarrow$ Addition of vectors
- Multiplication of polynomials?
- Billinear vector operator!
  $a * (b + c) = a * b + a * c = (b + c) * a$. General representation of multiplication: Tensors.
- Can represent "multiplication by a constant vector" as a matrix. Example (multiplication by x in $\mathbb{Z}[x]/(x^3 - 1)$):
  $$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

A Somewhat-Homomorphic Cryptosystem: Part I

- A homomorphic cryptosystem following the same format as FHE, but on a restricted class of circuits.
- In $\mathcal{R} = \mathbb{Z}[x]/(x^n - 1)$, let I and J be two relatively-prime ideals ($I + J = \mathcal{R}$)
- Public key: Two "obfuscated" bases B_I, B_J^{pk} of I and J, and a probability distribution D over I.
- Private key: A basis of short vectors B_J^{sk} for J.

A Somewhat-Homomorphic Cryptosystem: Part II

- Encryption:
- $\psi = \text{Encrypt}_{\epsilon}(pk, \pi) = (\pi + i) \bmod B_J^{pk}$
- i is a random vector in I sampled from D
- Decryption:
- $\pi = \text{Decrypt}_{\epsilon}(sk, \psi) = (\psi \bmod B_J^{sk}) \bmod B_I$

A Somewhat-Homomorphic Cryptosystem: Part II

- Encryption:
- $\psi = \text{Encrypt}_\epsilon(pk, \pi) = (\pi + i) \bmod B_J^{pk}$
- i is a random vector in I sampled from D
- Decryption:
- $\pi = \text{Decrypt}_\epsilon(sk, \psi) = (\psi \bmod B_J^{sk}) \bmod B_I$
- Works with careful choice of bases, distribution.
- View J as the "coarser" lattice, I as the "finer" lattice.

Another perspective

- Pick $\pi_j + i_j$ so that they always belong to $\mathcal{P}(B_J^{sk})$.
- Then, we are free to add/multiply ciphertexts $((\pi_j + i_j) \mod B_J^{pk})$ so long as the results stay in $\mathcal{P}(B_J^{sk})$
- Ensures that results don't wind up in a different congruence class $\mod B_I$ when we decrypt.
- $\pi_j + i_j$ is our "error signal" mentioned earlier!
- Measure size of the error by the Euclidean norm.
- Additions: $||a + b|| \leq ||a|| + ||b||$.
- (Binary) Multiplications: $||a * b|| \leq \sqrt{n}||a||||b||$.

Allowable Circuits

- If r_{DEC} is the size of the inscribed ball of $\mathcal{P}(B_J^{sk})$, we can evaluate circuits of depth (number of nested additions, multiplications) on the order of $log_2(log_2(r_{DEC}))$.
- Very slow-growing!
- Very conservative – assumes every operation is a binary multiplication.

Security of the Somewhat-Homomorphic Scheme

- Security of this scheme reduces to the hardness of the Shortest Independent Vector Problem
- Using it, an attacker could find B_J^{sk} from B_J^{pk}!
- The problem may be radically easier in Circulant Ideal Lattices, but we do not know if that's the case.

So We Have FHE, Right?

So We Have FHE, Right?

- No. That's not good enough. We need to represent the decryption circuit.

So We Have FHE, Right?

- No. That's not good enough. We need to represent the decryption circuit.
- Representing the decryption operation $(\psi \bmod B_J^{sk}) \bmod B_I$ requires evaluating $\psi - B_J^{sk}[(B_J^{sk})^{-1}\psi]$
- Hard to do with a small circuit because $(B_J^{sk})^{-1}\psi$ lives in $\mathbb{Q}^n$, not $\mathbb{Z}^n$

So We Have FHE, Right?

- No. That's not good enough. We need to represent the decryption circuit.
- Representing the decryption operation $(\psi \bmod B_J^{sk}) \bmod B_I$ requires evaluating $\psi - B_J^{sk}[(B_J^{sk})^{-1}\psi]$
- Hard to do with a small circuit because $(B_J^{sk})^{-1}\psi$ lives in $\mathbb{Q}^n$, not $\mathbb{Z}^n$
- Need to represent rationals or a decimal approximation of intermediate computational values!

So We Have FHE, Right?

- No. That's not good enough. We need to represent the decryption circuit.
- Representing the decryption operation $(\psi \bmod B_J^{sk}) \bmod B_I$ requires evaluating $\psi - B_J^{sk}[(B_J^{sk})^{-1}\psi]$
- Hard to do with a small circuit because $(B_J^{sk})^{-1}\psi$ lives in $\mathbb{Q}^n$, not $\mathbb{Z}^n$
- Need to represent rationals or a decimal approximation of intermediate computational values!
- Without some radical modification, makes the decryption circuit $\mathcal{D}_\epsilon$ always too deep to evaluate.

Gentry's Fixes

- 1. Use up only half of the available room for error.
- Result: coordinates of $(B_J^{sk})^{-1}\psi$ are each at most $\frac{1}{4}$ away from an integer.
- Less precision needed!

Gentry's Fixes

- 2. Have the *encrypter* help compute $(B_J^{sk})^{-1}\psi$!
- How? And how could that possibly be secure?
- Use subset-sum! Generate a large collection of matrices $A_1, ...A_m$, some (small) subset of which sums to $(B_J^{sk})^{-1}$, say $A_{s_1} + ...A_{s_n} = (B_J^{sk})^{-1}$.
- We can force this to have a unique solution.
- All $A_1, ...A_m$ are public knowledge.
- Someone (doesn't matter who!) publically computes $A_1\psi, ...A_m\psi$
- Include the indices $s_1, ...s_n$ in the secret key
- Evaluator uses the secret key's indices and the result from the encrypter to compute $A_{s_1}\psi + ...A_{s_n}\psi = (B_J^{sk})^{-1}\psi$

Is this scheme fully-homomorphic?

Yes!

But is it a secure form of encryption?

Security of the FHE Scheme

- For an attacker to obtain the secret key, they need to solve two hard problems:
- 1. Shortest Independent Vector Problem (B_J^{sk} from B_J^{pk})
- 2. Sparse Subset Sum Problem (subset of A_i's from B_J^{sk}.)
- Note: Solving 1 is enough to decrypt a ciphertext
- Best algorithms for each take exponential time in the worst-case, no efficient quantum algorithms are known.

Developments since 2009

- FHE is too resource-intensive for practical usage right now.
- Gentry et al. demonstrated a version of FHE which does not require bootstrapping [3], but the performance benefits if it uses bootstrapping on deep circuits.
- Gentry et al. also demonstrated a FHE scheme over the integers. [11].
- Even though circuit evaluation is very slow, evaluation is massively-parallel! [1].

Concluding Remarks

Questions?

Nathanael Black.
Homomorphic encryption and the approximate gcd problem.
2014.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim.
Evaluating 2-dnf formulas on ciphertexts.
In *Theory of Cryptography Conference*, pages 325–341. Springer, 2005.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan.
(leveled) fully homomorphic encryption without bootstrapping.
ACM Transactions on Computation Theory (TOCT), 6(3):13, 2014.

Jong-Hyuk Im, Taek-Young Youn, and Mun-Kyu Lee.
Privacy-preserving blind auction protocol using fully homomorphic encryption.
Advanced Science Letters, 22(9):2598–2600, 2016.

Yu Ishimaki, Kana Shimizu, Koji Nuida, and Hayato Yamana.
Poster: Privacy-preserving string search for genome sequences using fully homomorphic encryption.
Bioinformatics, 2016.

Wen-jie Lu, Shohei Kawasaki, and Jun Sakuma.
Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data.
2017.

Pascal Paillier.
Public-key cryptosystems based on composite degree residuosity classes.
In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

Ronald L Rivest, Len Adleman, and Michael L Dertouzos.
On data banks and privacy homomorphisms.
1978.

- Joseph H Silverman.
  An introduction to the theory of lattices and applications to cryptography.
  2006.

- Robert I Soare.
  Turing oracle machines, online computing, and three displacements in computability theory.
  Annals of Pure and Applied Logic, 160(3):368–399, 2009.

- Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan.
  Fully homomorphic encryption over the integers.
  Cryptology ePrint Archive, Report 2009/616, 2009.
  `http://eprint.iacr.org/2009/616`.