Fully Homomorphic Encryption

Zvika Brakerski
Weizmann Institute of Science
What Are You Searching For?

Medical information, navigation, email, business information, other personal information...

Want privacy!
Outsourcing Computation

What if $x$ is private?
How to Keep Private From the Cloud

We promise we won't look at your data. Honest!

We want real protection.
Outsourcing Computation

Homomorphic Evaluation function:

\[ f, \text{Enc}(x) \rightarrow \text{Enc}(f(x)) \]

Learns nothing about \( x \).

\[ y = \text{Eval}(f, \text{Enc}(x)) \]

\[ \text{Dec}(y) = f(x) \]

Fully Homomorphic = Homomorphism for any efficient \( f \)

Homomorphic \( f, \text{Enc}(x) \)

Goal: \text{Eval} for universal set of gates (NAND(\( x, y \))=1-xy)
Some Applications

In the cloud:

- Private outsourcing of computation.
- Near-optimal private outsourcing of storage (single-server PIR). [G09,BV11b]
- Verifiable outsourcing (delegation). [GGP11,CKV11,KRR13,KRR15]
- Private machine learning in the cloud. [GLN12,HW13]

Secure multiparty computation:

- Low-communication multiparty computation. [AJLTVW12,LTV12]
- More efficient MPC. [BDOZ11,DPSZ12,DKLPSS12]

Primitives:

- Succinct argument systems. [GLR11,DFH11,BCCT11,BC12,BCCT12,BCGT13,…]
- General functional encryption. [GKPVZ12]
- Indistinguishability obfuscation for all circuits. [GGHRSW13]
Making Crypto History

30 years of hardly scratching the surface:

- Addition [RSA78, R79, GM82, G84, P99, R05].
- Addition + 1 multiplication [BGN05, GHV10].
- Other variants [SYY99, IP07, MPH10].

... is it even possible?
FHE Challenges

Understanding.

Security.
- Cryptographic assumptions.
- Security notions.

Efficiency.
- Size of keys/ciphertexts.
- Time overhead for Eval.
- Computational model.
Constructing (Somewhat) Homomorphic Encryption

**Basic Idea:** Find scheme s.t. \( c \approx m + 2e \)

Add/multiply ciphertexts \( \Rightarrow \) Add/multiply messages

Noise grows with homomorphic evaluation – must not grow “too much”!

In the example above: \( |e_{\text{mult}}| \approx |e_{\text{in}}|^2 \)
Noise in Homomorphic Evaluation

Noise grows during homomorphic evaluation

\[ |e_{out}| \leq E^{2^d} \]

\[ |e_{i+1}| \leq |e_i|^2 \]

\[ |e_{in}| \leq E \]
Some of the Progress Since 2009

• From ad-hoc assumption to worst-case lattice assumption [BV11b,BGV12,BV14].
  – As secure as any other encryption scheme.

• Noise is down to $|e_{mult}| \approx k \cdot |e_{in}|$ [BGV12,B12,GSW13,BV14].
  – $|e_{out}| \leq k^d \cdot E$ (instead of $E^{2d}$).
  – “Leveled” FHE.

• Using polynomial rings to improve efficiency [G09,SV10,BV11a,BGV12,GHS12a,GHS12b,GHS12c,GHPS13,AP13].

• “Batching” many messages in single ciphertext [SV10,BGV12,GHS12a,GHS12b,GHS12c,HS15].

• But still need “bootstrapping” to get full homomorphism...
Bootstrapping [G09]

Given scheme with bounded $d_{hom}$
How to extend its homomorphic capability?

**Idea:** Do a few operations, then “switch” to a new instance

Switch keys

“cost” in homomorphism

$(pk_3, sk_3)$

$(pk_2, sk_2)$

$(pk_1, sk_1)$
**How to Switch Keys**

Decryption circuit:

\[ \text{Dec}_{sk}(\cdot) \]

\[ x \]

\[ c \]

Dual view:

\[ \text{Dec}_{(\cdot)}(c) \equiv h_c(\cdot) \]

\[ x \]

\[ sk \]

\[ h_c(sk) = \text{Dec}_{sk}(c) = x \]

given \( c \), server can compute circuit for \( h_c(\cdot) \)

Apply \( h_c(\cdot) \) homomorphically on \( sk \) !

\[ \text{aux} = \text{Enc}_{pk',(sk)} \]

\[ \text{Eval}_{pk'}(h_c, aux) = \text{Eval}_{pk'}(h_c, \text{Enc}_{pk'}(sk)) \]

\[ = \text{Enc}_{pk'}(h_c(sk)) = \text{Enc}_{pk'}(\text{Dec}_{sk}(c)) \]

\[ = \text{Enc}_{pk'}(x) \]

**hom. capacity of output:**

\[ d_{\text{hom}} - d_{h_c} = d_{\text{hom}} - d_{\text{dec}} \]
Bootstrapping [G09]

Given scheme with bounded $d_{hom}$.

How to extend its homomorphic capability?

Idea: Do a few operations, then “switch” to a new instance

Switch keys: $aux_{1→2} = Enc_{pk_2}(sk_1), aux_{2→3} = Enc_{pk_3}(sk_2)$

⇒ Bootstrapping if $d_{hom} \geq d_{dec} + 1$

Downside: Need to generate many keys...
Bootstrapping [G09]

Given scheme with bounded $d_{hom}$. How to extend its homomorphic capability?

**Idea:** Do a few operations, then "switch" to a new instance.

$$\text{aux} = Enc_{pk}(sk)$$

Switch from key to itself!

Functionality of switching works.

Circular security required.
(Some) Public Implementations of FHE

- **HElib (IBM/NYU)**
  - Ring-LWE (ideal-lattice) scheme of [BGV12], optimizations of [GHS12a]
  - [https://github.com/shaih/HElib](https://github.com/shaih/HElib)

- **“Stanford FHE”**
  - LWE scheme of [B12] with optimizations

- **FHEW (UCSD)**
  - Ring-LWE scheme of [DM14], built upon approximate eigenvector approach of [GSW13,BV14,AP14]
  - No batching but very fast bootstrapping
  - [https://github.com/lducas/FHEW](https://github.com/lducas/FHEW)
So Where is That Homomorphic Google Search?

• Circuit model = huge overhead.
  – Inherent? Need to touch all elements to not leak.

• Bootstrapping is expensive.
  – No known alternative for deep computations.

• Memory requirements are huge (GBs).
  – Large ciphertexts, long keys.
  – Can “batch” to reduce overhead.
Thank You!