A review of fully homomorphic encryption

Zhenfei Zhang

Oct 18 2019

Introduction

Privacy Homomorphism

- Raised in 1978 by Rivest, Adleman and Dertouzos
- To evaluate arbitrary number of ciphertext, without knowing corresponding plaintext.

$$\textsc{Decrypt}(\mathbf{sk}, \textsc{Eval}^n(\mathbf{pk}, \mathcal{C}^n, c_1, c_2, c_3, ..., c_n)) = \mathcal{C}^n(m_1, m_2, m_3, ..., m_n)$$

- Fully homomorphic encryption allows circuit $\mathcal{C}$ with any depth
- Leveled homomorphic encryption allows a limited number of circuit depth

Introduction

Constructions

- (Principal) Ideal lattice based schemes
  - Finding short generator for principal ideal lattice is easy
- Integer based schemes
  - Based on Approximate GCD problem.
- (Ring-)Learning with error based schemes
  - 2011, Brakerski-Vaikuntanathan, based on (Ring-)LWE
  - 2013, Brakerski-Gentry-Vaikuntanathan, modulus switching
  - 2013, Gentry-Sahai-Waters, approximate eigenvector method
  - 2014, Brakerski et al. leveled HE without bootstrapping
  - ...

Introduction

Some random thoughts

- Lattice based crypto invented 1996 (NTRU, GGH); matured 2006 (LWE); start standardization 2016 (NIST)
- FHE invented 2008; matured ???; start standardization 2017

How mature is FHE?

- Theoretician starts to build new toys
  - attribute based encryption, multi-linear map, program obfuscation
- Practitioner starts to write efficient implementation
  - HElib, SEAL, TFHE, cuHE

Background

Gentry's Framework

- Construct a somewhat homomorphic encryption scheme that enables bitwise homomorphism;

$\times$	0	1
0	0	0
1	0	1

$\otimes$	Enc(0)	Enc(1)
Enc(0)	Enc(0)	Enc(0)
Enc(1)	Enc(0)	Enc(1)

- Use bootstrap technique to enable unlimited homomorphic circuit depth;

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

- Is semantic secure assuming *Approx-GCD* is hard;
- Can be turned into a public key system using the *subset sum problem*;

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

- Is indeed Homomorphic

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

- Add(c_1, c_2)
  - $c_1 = a_1 s + 2r_1 + m_1$
  - $c_2 = a_2 s + 2r_2 + m_2$
  - $c_1 + c_2 = (a_1 + a_2)s + 2(r_1 + r_2) + (m_1 + m_2)$
  - Decrypt($c_1 + c_2$) $= m_1 + m_2 \bmod 2$

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

- Mul(c_1, c_2)
  - $c_1 = a_1 s + 2r_1 + m_1$
  - $c_2 = a_2 s + 2r_2 + m_2$
  - $c_1 \times c_2 = (\dots)s + 2(r_2 m_1 + r_1 m_2 + 2r_1 r_2) + (m_1 \times m_2)$
  - Decrypt($c_1 + c_2$) $= m_1 \times m_2 \bmod 2$ if $r_2 m_1 + r_1 m_2 + 2r_1 r_2 < s/2$

A toy example

A symmetric system

- The secret key: an odd integer s;
- Encrypt(m):
  - Message $m \in \{0, 1\}$
  - Sample $r \ll s$ and a, random integers;
  - Return $c = as + 2r + m$
- Decrypt(c)
  - Output $m' = c \bmod s \bmod 2$
  - $m' = m$ as long as $r < s/2$;

- Decrypt($c_1 \times c_2$) $= m_1 \times m_2 \bmod 2$ if $r_2 m_1 + r_1 m_2 + 2 r_1 r_2 < s/2$
- Noise grows quadratic in circuit depth
- Capability $\tau = O(\log_r s)$

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2 r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

Want to achieve

- Homomorphic for any circuit

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

Let's try to evaluate Decrypt circuit

- Suppose the decrypt circuit depth is t
- Can be evaluated homomorphically if $t < \tau$

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

What happens if we evaluate the decryption circuit

- Encrypt ciphertexts ($Enc(c)$) and secret keys ($Enc(\mathbf{sk})$);
- Evaluate the decryption circuit homomorphically over $Enc(c)$ and $Enc(\mathbf{sk})$:

$$\textsc{Eval}(\mathbf{pk}, \mathcal{C}_D, Enc(c), Enc(\mathbf{sk})) = Enc(m)$$

- $\mathcal{C}_D$ is the decryption circuit;
- The formula is correct so long as $t < \tau$;
- $Enc(m)$ is a new ciphertext with minimum noise - can be evaluated again;

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2 r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

Achieving fully homomorphic encryption

- If $\tau > t + 1$, then the scheme is fully homomorphic
- For any input circuit, evaluate it gate by gate
- Bootstrap after every evaluation to refresh the noise

Example extended

What we have

- Noise $r_2 m_1 + r_1 m_2 + 2r_1 r_2$ grows quadratic during Mul.
- Can set $s \gg r$ to allow for a circuit depth $\tau = O(\log_r s)$;
- Decrypt circuit depth denoted by t.

Caveat

- Efficiency improvement – use lattice
- circular security – open problem

Questions?

Warm up

Diffie-Hellman KEX

Alice		Bob
$A = s \cdot a$		
	Alice sends A to Bob	
		$\mathrm{B} = r \cdot a$
	Bob sends B to Alice	
$C = (sr) \cdot a$		$C = (sr) \cdot a$

- a: group generator
- s, r: scalar
- $\cdot$: group multiplication

Warm up

Diffie-Hellman KEX, on a different setting

Alice		Bob
$A = s \cdot a + e_1$		
	Alice sends A to Bob	
		$B = r \cdot a + e_2$
	Bob sends B to Alice	
$C = (sr) \cdot a + e_1 b$		$C = (sr) \cdot a + e_2 a$

- G: a public, large polynomial
- s, r: small polynomials
- $\cdot$: polynomial multiplication

Warm up, continued

Setup

- Work over a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$
- May view polynomials as vectors

Definition (Ring-LWE)

Let s be a secret, *uniform* polynomial over $\mathcal{R}_q$. Let $a_1, \ldots, a_t$ polynomials sampled uniformly from $\mathcal{R}_q$. Let $e_1, \ldots, e_t$ be error polynomials whose norm are bounded by $\beta \ll q$. The LWE is given pairs $\{(a_i, b_i := a_i s + e_i)\}_{i=1}^{t}$, find s.

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \ldots, 0 \rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \ldots, 0 \rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

correctness

$$\begin{aligned} m' &= c_2 - c_1 s \\ &= asr + 2er + 2e_2 + \langle m, 0, \ldots, 0 \rangle - asr - 2e_1 s \\ &= 2er + 2e_2 + \langle m, 0, \ldots, 0 \rangle - 2e_1 s \\ &\equiv \langle m, 0, \ldots, 0 \rangle \bmod 2 \end{aligned}$$

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \dots, 0 \rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

- The *dual Regev* cryptosystem
- Semantic secure assuming (Ring-)LWE is hard
- Most lattice based NIST PQC submissions follow this framework

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \ldots, 0 \rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

- Add$(c_1 = (c_{1,1}, c_{1,2}),\ c_2 = (c_{2,1}, c_{2,2}))$
  - $c_{+,1} = a(r_1 + r_2) + 2(e_{1,1} + e_{2,1})$
  - $c_{+,2} = b(r_1 + r_2) + 2(e_{1,2} + e_{2,2}) + \langle m_1 + m_2, 0, \ldots, 0 \rangle$
  - Decrypt$(c_{+,1}, c_{+,2}) = \langle m_1 + m_2, 0, \ldots, 0 \rangle \bmod 2$

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \dots, 0\rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

- Mul$(c_1 = (c_{1,1}, c_{1,2}),\ c' = (c_{2,1}, c_{2,2}))$
  - $c_{\times,1} = c_{1,1}c_{2,1} = a^2 r_1 r_2 + 2(ar_1 e_{2,1} + ar_2 e_{1,1} + 2e_{1,1}e_{2,1})$
  - $c_{\times,2} = c_{1,2}c_{2,1} + c_{1,1}c_{2,2} = 2a^2 sr_1 r_2 + 4aer_1 r_2 + 2ase_{1,1}r_2 + 2asr_1 e_{2,1} + 4ee_{1,1}r_2 + 2ae_{1,2}r_2 + am_1 r_2 + 4er_1 e_{2,1} + 2ar_1 e_{2,2} + ar_1 m_2 + 4e_{1,2}e_{2,1} + 2m_1 e_{2,1} + 4e_{1,1}e_{2,2} + 2e_{1,1}m_2$
  - $c_{\times,3} = a^2 s^2 r_1 r_2 + 4aser_1 r_2 + 4e^2 r_1 r_2 + 2ase_{1,2}r_2 + asm_1 r_2 + 2asr_1 e_{2,2} + asr_1 m_2 + 4ee_{1,2}r_2 + 2em_1 r_2 + 4er_1 e_{2,2} + 2er_1 m_2 + 4e_{1,2}e_{2,2} + 2m_1 e_{2,2} + 2e_{1,2} * m_2 + \langle m_1 \times m_2, 0, \dots, 0\rangle$
- $m' = c_{\times,3} - c_{\times,2}s - c_{\times,1}s^2 \bmod 2$

Lattice based construction

- Public key $(a, b = as + 2e)$; secret key s.
- Encrypt(m)
  - $c_1 = ar + 2e_1$
  - $c_2 = br + 2e_2 + \langle m, 0, \ldots, 0\rangle$
- Decrypt$((c_1, c_2))$
  - $m' = c_2 - c_1 s \bmod 2$

$$
\begin{aligned}
c_{\times,3} - c_{\times,2}s - c_{\times,1}s^2 \\
&\equiv 4e^2 r_1 r_2 + 4ee_{1,2}r_2 + 2em_1 r_2 + 4er_1 e_{2,2} + 2er_1 m_2 + \\
&\quad 4e_{1,2}e_{2,2} + 2m_1 e_{2,2} + 2m_2 e_{1,2} + m_1 m_2 \pmod{s}
\end{aligned}
$$

Noise grows quadratic

- Red part $< |s|_\infty/2$

Achieving FHE

- Set appropriate parameters so that $\tau > t + 1$

Comparing to Integer based solution

- More efficient - decryption circuit is shallower
- Post-quantum secure
- SIMD
- Further optimization: re-linearlization; modulus switching

FHE in practice

- Only Leveled HE is being used;
- Analyze the use case to obtain its maximum circuit depth t
- Set parameters for the system so that $\tau > t$

FHE in practice

- Only Leveled HE is being used;
- Analyze the use case to obtain its maximum circuit depth t
- Set parameters for the system so that $\tau > t$

How to derive parameters

- Based on best known attacks
- Primal attack and dual attack

Thank you!