Homomorphic Encryption

Jay Daigle

Occidental College

November 7, 2019

1. I want to search a medical database for information about an STD without letting anyone know I have one.

1. I want to search a medical database for information about an STD without letting anyone know I have one.
2. I want to search for politically dangerous information in an authoritarian regime.

1. I want to search a medical database for information about an STD without letting anyone know I have one.
2. I want to search for politically dangerous information in an authoritarian regime.
3. I want to store and search my data in the cloud without giving any internet companies access to it.

1. I want to search a medical database for information about an STD without letting anyone know I have one.
2. I want to search for politically dangerous information in an authoritarian regime.
3. I want to store and search my data in the cloud without giving any internet companies access to it.
4. I want to make a database of genetics information available to researchers without allowing them to identify any specific person.

1. I want to search a medical database for information about an STD without letting anyone know I have one.
2. I want to search for politically dangerous information in an authoritarian regime.
3. I want to store and search my data in the cloud without giving any internet companies access to it.
4. I want to make a database of genetics information available to researchers without allowing them to identify any specific person.

Definition

Let R, S be rings. We say a function $f : R \to S$ is a homomorphism if $f(x+y) = f(x) + f(y)$ and $f(xy) = f(x)f(y)$.

Key generation:

1. Alice generates a random small polynomial $s(x)$; this is the shared, symmetric key.

Key generation:

1. Alice generates a random small polynomial $s(x)$; this is the shared, symmetric key.

Encryption:

1. Alice generates a random polynomial $a(x)$ from the entire ring, and a random small polynomial $e(x)$.
2. Her message is a string of bits, which she can think about as a polynomial $m(x)$ with coefficients either 0 or 1.

Key generation:

1. Alice generates a random small polynomial $s(x)$; this is the shared, symmetric key.

Encryption:

1. Alice generates a random polynomial $a(x)$ from the entire ring, and a random small polynomial $e(x)$.
2. Her message is a string of bits, which she can think about as a polynomial $m(x)$ with coefficients either 0 or 1.
3. Alice computes $c_1(x) = -a(x)$ and $c_0(x) = a(x)s(x) + 2e(x) + m(x)$.
4. She transmits the ciphertext $(c_0(x), c_1(x))$.

Key generation:

1. Alice generates a random small polynomial $s(x)$; this is the shared, symmetric key.

Encryption:

1. Alice generates a random polynomial $a(x)$ from the entire ring, and a random small polynomial $e(x)$.
2. Her message is a string of bits, which she can think about as a polynomial $m(x)$ with coefficients either 0 or 1.
3. Alice computes $c_1(x) = -a(x)$ and $c_0(x) = a(x)s(x) + 2e(x) + m(x)$.
4. She transmits the ciphertext $(c_0(x), c_1(x))$.

Decryption:

1. Bob receives $(c_0(x), c_1(x))$.
2. He computes $c_0(x) + c_1(x)s(x)$.
3. He reduces modulo 2, and gets the message $m(x)$.

Keygen:

1. Alice generates a random polynomial a_0 and random small polynomials s and e_0.
2. Alice computes $b_0 = as + 2e_0$. Her public key is (a_0, b_0).

Keygen:

1. Alice generates a random polynomial a_0 and random small polynomials s and e_0.
2. Alice computes $b_0 = as + 2e_0$. Her public key is (a_0, b_0).

Encryption:

1. Bob generates random small polynomials v, e_1, e_2.
2. He computes $a_1 = a_0 v + 2e_1, b_1 = b_0 v + 2e_2$.

Keygen:

1. Alice generates a random polynomial a_0 and random small polynomials s and e_0.
2. Alice computes $b_0 = as + 2e_0$. Her public key is (a_0, b_0).

Encryption:

1. Bob generates random small polynomials v, e_1, e_2.
2. He computes $a_1 = a_0 v + 2e_1, b_1 = b_0 v + 2e_2$.
3. He computes $c_0 = b_1 + m$ and $c_1 = -a_1$.
4. The ciphertext is (c_0, c_1).

Keygen:

1. Alice generates a random polynomial a_0 and random small polynomials s and e_0.
2. Alice computes $b_0 = as + 2e_0$. Her public key is (a_0, b_0).

Encryption:

1. Bob generates random small polynomials v, e_1, e_2.
2. He computes $a_1 = a_0 v + 2e_1, b_1 = b_0 v + 2e_2$.
3. He computes $c_0 = b_1 + m$ and $c_1 = -a_1$.
4. The ciphertext is (c_0, c_1).

Decryption:

1. Alice receives (c_0, c_1).
2. Alice computes $M = c_0 + sc_1$.
3. Alice reduces M mod 2 and gets the message m.

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.
Add ciphertexts pointwise. Then we have

$$\mathbf{c} + \mathbf{c}' = (c_0, \ldots, c_d) + (c_0', \ldots, c_d') = (c_0 + c_0', \ldots, c_d + c_d').$$

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.
Add ciphertexts pointwise. Then we have

$$\mathbf{c} + \mathbf{c}' = (c_0, \ldots, c_d) + (c_0', \ldots, c_d') = (c_0 + c_0', \ldots, c_d + c_d').$$

Multiplication: introduce a new variable v, and write:

$$\mathbf{c} = \sum_{i=0}^{d} c_i v^i = c_0 + c_1 v + \cdots + c_d v^d \in R_q[d].$$

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.
Add ciphertexts pointwise. Then we have

$$\mathbf{c} + \mathbf{c}' = (c_0, \ldots, c_d) + (c_0', \ldots, c_d') = (c_0 + c_0', \ldots, c_d + c_d').$$

Multiplication: introduce a new variable v, and write:

$$\mathbf{c} = \sum_{i=0}^{d} c_i v^i = c_0 + c_1 v + \cdots + c_d v^d \in R_q[d].$$

$$\mathbf{c} \times \mathbf{c}' = (\hat{c}_0, \ldots, \hat{c}_{d+d'})$$
$$\left(\sum_{i=0}^{d} c_i v^i\right)\left(\sum_{i=0}^{d'} c_i' v^i\right) = \sum_{i=0}^{d+d'} \hat{c}_i v^i.$$

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.
Add ciphertexts pointwise. Then we have

$$\mathbf{c} + \mathbf{c}' = (c_0, \ldots, c_d) + (c_0', \ldots, c_d') = (c_0 + c_0', \ldots, c_d + c_d').$$

Multiplication: introduce a new variable v, and write:

$$\mathbf{c} = \sum_{i=0}^{d} c_i v^i = c_0 + c_1 v + \cdots + c_d v^d \in R_q[d].$$

$$\mathbf{c} \times \mathbf{c}' = (\hat{c}_0, \ldots, \hat{c}_{d+d'})$$
$$\left(\sum_{i=0}^{d} c_i v^i\right)\left(\sum_{i=0}^{d'} c_i' v^i\right) = \sum_{i=0}^{d+d'} \hat{c}_i v^i.$$

Decryption: compute $\mathbf{s} = (1, s, \ldots, s^D)$, and

$$\langle \mathbf{c}, \mathbf{s} \rangle = \sum_{i=0}^{D} c_i s^i.$$

Ciphertext: a sequence $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$.
Add ciphertexts pointwise. Then we have

$$\mathbf{c} + \mathbf{c}' = (c_0, \ldots, c_d) + (c_0', \ldots, c_d') = (c_0 + c_0', \ldots, c_d + c_d').$$

Multiplication: introduce a new variable v, and write:

$$\mathbf{c} = \sum_{i=0}^{d} c_i v^i = c_0 + c_1 v + \cdots + c_d v^d \in R_q[d].$$

$$\mathbf{c} \times \mathbf{c}' = (\hat{c}_0, \ldots, \hat{c}_{d+d'})$$
$$\left(\sum_{i=0}^{d} c_i v^i\right)\left(\sum_{i=0}^{d'} c_i' v^i\right) = \sum_{i=0}^{d+d'} \hat{c}_i v^i.$$

Decryption: compute $\mathbf{s} = (1, s, \ldots, s^D)$, and

$$\langle \mathbf{c}, \mathbf{s} \rangle = \sum_{i=0}^{D} c_i s^i.$$