Learning with Errors and GSW's Homomorphic Encryption

July 4, 2019

Intuition of LWE

$$14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$$

$$6s_1 + 10s_2 + 13s_3 + s_4 \approx 3 \pmod{17}$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 \approx 12 \pmod{17}$$

$$9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$$

$$3s_1 + 6s_2 + 4s_3 + 5s_4 \approx 16 \pmod{17}$$

$$\vdots$$

$$6s_1 + 7s_2 + 16s_3 + 2s_4 \approx 3 \pmod{17}$$

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:
  - pick coefficients $\vec{a} \in \mathbb{Z}_q^n$ at random

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:
  - pick coefficients $\vec{a} \in \mathbb{Z}_q^n$ at random
  - Sample error $e \leftarrow \chi$

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:
  - pick coefficients $\vec{a} \in \mathbb{Z}_q^n$ at random
  - Sample error $e \leftarrow \chi$
  - Output $(\vec{a} \cdot \vec{s} + e, \vec{a}) \in \mathbb{Z}_q^{1+n}$

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:
  - pick coefficients $\vec{a} \in \mathbb{Z}_q^n$ at random
  - Sample error $e \leftarrow \chi$
  - Output $(\vec{a} \cdot \vec{s} + e, \vec{a}) \in \mathbb{Z}_q^{1+n}$
- Output: $\vec{s}$

Definition (LWE Problem)

- Pick uniformly random $\vec{s} \in \mathbb{Z}_q^n$
- Take χ to be a discrete Gaussian distribution with some variance to be specified
- Input: $poly(n)$ equations sampled as below:
  - pick coefficients $\vec{a} \in \mathbb{Z}_q^n$ at random
  - Sample error $e \leftarrow \chi$
  - Output $(\vec{a} \cdot \vec{s} + e, \vec{a}) \in \mathbb{Z}_q^{1+n}$
- Output: $\vec{s}$
- Note that $(\vec{a} \cdot \vec{s} + e, \vec{a}) \cdot (1, -\vec{s}) = e$

Homomorphic encryption

Allows secure communication by using the tools below:

Homomorphic encryption

Allows secure communication by using the tools below:

- pk: "public key", used for encryption

Homomorphic encryption

Allows secure communication by using the tools below:

- *pk*: "public key", used for encryption
- $\text{Enc}_{pk} : \Sigma^* \times \mathcal{R} \to U$

Homomorphic encryption

Allows secure communication by using the tools below:

- pk: "public key", used for encryption
- $\mathrm{Enc}_{pk} : \Sigma^* \times \mathcal{R} \to U$
- sk: "secret key", used for decryption

Homomorphic encryption

Allows secure communication by using the tools below:

- pk: "public key", used for encryption
- $\mathrm{Enc}_{pk} : \Sigma^* \times \mathcal{R} \to U$
- sk: "secret key", used for decryption
- $\mathrm{Dec}_{sk} : U \to \Sigma^*$.

Homomorphic encryption

Allows secure communication by using the tools below:

- *pk*: "public key", used for encryption
- $\mathrm{Enc}_{pk} : \Sigma^* \times \mathcal{R} \to U$
- *sk*: "secret key", used for decryption
- $\mathrm{Dec}_{sk} : U \to \Sigma^*$.
- *evk*: "evaluation key"

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector
- ciphertexts are matrices with $\vec{v}$ approximately as an eigenvector

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector
- ciphertexts are matrices with $\vec{v}$ approximately as an eigenvector
- plaintexts are corresponding approximate eigenvalues

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector
- ciphertexts are matrices with $\vec{v}$ approximately as an eigenvector
- plaintexts are corresponding approximate eigenvalues
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector
- ciphertexts are matrices with $\vec{v}$ approximately as an eigenvector
- plaintexts are corresponding approximate eigenvalues
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$
- $(C_1 C_2)\vec{v} \approx (\lambda_1 \lambda_2)\vec{v}$

Idea of GSW's homomorphic encryption

Intuitive idea as follows:

- private key $\vec{v}$ is a vector
- ciphertexts are matrices with $\vec{v}$ approximately as an eigenvector
- plaintexts are corresponding approximate eigenvalues
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$
- $(C_1 C_2)\vec{v} \approx (\lambda_1 \lambda_2)\vec{v}$
- When plaintexts are booleans, $I_N - C_1 C_2$ encodes NAND.

Idea of GSW's homomorphic encryption cont.

- Take pk to be a matrix where each row is a tuple

Idea of GSW's homomorphic encryption cont.

- Take pk to be a matrix where each row is a tuple
- Take $sk = (1, -\vec{s}$

Idea of GSW's homomorphic encryption cont.

- Take pk to be a matrix where each row is a tuple
- Take $sk = (1, -\vec{s}$
- Take $C = \mu I_n +$

GSW's tools - terminology

- For the sake of illustration, fix $q = 2^l = 2^4$.

GSW's tools - terminology

- For the sake of illustration, fix $q = 2^l = 2^4$.
- Let $a = \underline{a_3 a_2 a_1 a_0} = \sum 2^i a_i$ represent the base-2 digits of a

GSW's tools - terminology

- For the sake of illustration, fix $q = 2^l = 2^4$.
- Let $a = \underline{a_3 a_2 a_1 a_0} = \sum 2^i a_i$ represent the base-2 digits of a
- Similarly, denote base-2 numbers like $\underline{1011} = 8 + 2 + 1 = 11$

GSW's tools - Powersof2

- $\text{Powersof2}(a) = (2^3 a, 2^2 a, 2a, a)$

GSW's tools - Powersof2

- Powersof2$(a) = (2^3 a, 2^2 a, 2a, a)$
- $= \begin{pmatrix} 2^3 a \\ 2^2 a \\ 2a \\ a \end{pmatrix}$

GSW's tools - Powersof2

- $\text{Powersof2}(a) = (2^3 a, 2^2 a, 2a, a)$
- $= \begin{pmatrix} 2^3 a \\ 2^2 a \\ 2a \\ a \end{pmatrix}$
- $\text{Powersof2}(\underline{11}) = (\underline{1000}, \underline{1100}, \underline{110}, \underline{11})$

GSW's tools - Powersof2

- $\text{Powersof2}(a) = (2^3 a, 2^2 a, 2a, a)$
- $= \begin{pmatrix} 2^3 a \\ 2^2 a \\ 2a \\ a \end{pmatrix}$
- $\text{Powersof2}(\underline{11}) = (\underline{1000}, \underline{1100}, \underline{110}, \underline{11})$
- $\text{Powersof2}(a, b) = (2^3 a, 2^2 a, 2a, a, 2^3 b, 2^2 b, 2b, b)$

GSW's tools - BitDecomp

- $\text{BitDecomp}(\underline{a_3 a_2 a_1 a_0}) = \begin{pmatrix} a_3 & a_2 & a_1 & a_0 \end{pmatrix}$

GSW's tools - BitDecomp

- $\text{BitDecomp}(\underline{a_3 a_2 a_1 a_0}) = \begin{pmatrix} a_3 & a_2 & a_1 & a_0 \end{pmatrix}$
- $\text{BitDecomp}\begin{pmatrix} \underline{a_3 a_2 a_1 a_0} & \underline{b_3 b_2 b_1 b_0} \\ \underline{c_3 c_2 c_1 c_0} & \underline{d_3 d_2 d_1 d_0} \end{pmatrix} =$
$\begin{pmatrix} a_3 & a_2 & a_1 & a_0 & b_3 & b_2 & b_1 & b_0 \\ c_3 & c_2 & c_1 & c_0 & d_3 & d_2 & d_1 & d_0 \end{pmatrix}$

GSW's tools - BitDecomp

- $\text{BitDecomp}(\underline{a_3 a_2 a_1 a_0}) = \begin{pmatrix} a_3 & a_2 & a_1 & a_0 \end{pmatrix}$
- $\text{BitDecomp}\begin{pmatrix} \underline{a_3 a_2 a_1 a_0} & \underline{b_3 b_2 b_1 b_0} \\ \underline{c_3 c_2 c_1 c_0} & \underline{d_3 d_2 d_1 d_0} \end{pmatrix} =$
$\begin{pmatrix} a_3 & a_2 & a_1 & a_0 & b_3 & b_2 & b_1 & b_0 \\ c_3 & c_2 & c_1 & c_0 & d_3 & d_2 & d_1 & d_0 \end{pmatrix}$
- $\text{BitDecomp}(A)\text{Powersof2}(\vec{b}) = A\vec{b}$

GSW's tools - BitDecomp^{-1}

- $\text{BitDecomp}^{-1}\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix} = \underline{1001}$

GSW's tools - BitDecomp^{-1}

- $\text{BitDecomp}^{-1}\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix} = \underline{1001}$
- $\text{BitDecomp}^{-1}\begin{pmatrix}0 & 0 & \underline{10} & 0\end{pmatrix} = \underline{0100}$

GSW's tools - BitDecomp^{-1}

- $\text{BitDecomp}^{-1}\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix} = \underline{1001}$
- $\text{BitDecomp}^{-1}\begin{pmatrix}0 & 0 & \underline{10} & 0\end{pmatrix} = \underline{0100}$
- $A\text{Powersof2}(\vec{b}) = \text{BitDecomp}^{-1}(A)\vec{b}$

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten}\left(\underline{110} \quad \underline{101} \quad 1 \quad \underline{11}\right)$

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$
- $= \text{BitDecomp} \circ \text{BitDecomp}^{-1}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$
- $= \text{BitDecomp} \circ \text{BitDecomp}^{-1}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$
- $= \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}$

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten} \begin{pmatrix} \underline{110} & \underline{101} & 1 & \underline{11} \end{pmatrix}$
- $= \text{BitDecomp} \circ \text{BitDecomp}^{-1} \begin{pmatrix} \underline{110} & \underline{101} & 1 & \underline{11} \end{pmatrix}$
- $= \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$
- $\text{Flatten}(A)$ has only $0, 1$ entries

GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$
- $= \text{BitDecomp} \circ \text{BitDecomp}^{-1}\begin{pmatrix}\underline{110} & \underline{101} & 1 & \underline{11}\end{pmatrix}$
- $= \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}$
- $\text{Flatten}(A)$ has only $0, 1$ entries
- $\text{Flatten}(A)\text{Powersof2}(\vec{b}) = A\text{Powersof2}(\vec{b})$

GSW's Construction - Setup

Choose the following parameters:

- Modulus $q = 2^l$ (to simplify some proofs)

GSW's Construction - Setup

Choose the following parameters:

- Modulus $q = 2^l$ (to simplify some proofs)
- Lattice dimension n

GSW's Construction - Setup

Choose the following parameters:

- Modulus $q = 2^l$ (to simplify some proofs)
- Lattice dimension n
- Error distribution $\chi(\lambda, L)$

GSW's Construction - Setup

Choose the following parameters:

- Modulus $q = 2^l$ (to simplify some proofs)
- Lattice dimension n
- Error distribution $\chi(\lambda, L)$
- $m \in O(n \log q)$

GSW's Construction - Secret Keygen

1. Sample $\vec{s} \leftarrow \mathbb{Z}_q^n$ uniformly. This represents the solution of the LWE system of equations.

GSW's Construction - Secret Keygen

1. Sample $\vec{s} \leftarrow \mathbb{Z}_q^n$ uniformly. This represents the solution of the LWE system of equations.
2. Output sk as 1 on the first coordinate, followed by $-\vec{s}$.

GSW's Construction - Public Keygen

1. Generate $B \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly

GSW's Construction - Public Keygen

1. Generate $B \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly
2. Sample $\vec{e} \leftarrow \chi^m$

GSW's Construction - Public Keygen

1. Generate $B \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly
2. Sample $\vec{e} \leftarrow \chi^m$
3. Set pk as $B\vec{s} + \vec{e}$ on the first column, followed by the columns of B.

GSW's Construction - Public Keygen

1. Generate $B \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly
2. Sample $\vec{e} \leftarrow \chi^m$
3. Set pk as $B\vec{s} + \vec{e}$ on the first column, followed by the columns of B.
4. Observe that $pk \cdot sk = \vec{e}$

GSW's Construction - Encryption

Input: μ

1. Sample $R \in \{0,1\}^{N \times m}$

GSW's Construction - Encryption

Input: μ

1. Sample $R \in \{0,1\}^{N \times m}$
2. Output $\text{Flatten}(\mu \cdot I + \text{BitDecomp}(R \cdot pk))$

GSW's Construction - Decryption

$$\text{Flatten}(\mu \cdot I + \text{BitDecomp}(R \cdot pk)) \cdot \text{Powersof2}(sk)$$

GSW's Construction - Decryption

$$\text{Flatten}(\mu \cdot I + \text{BitDecomp}(R \cdot pk)) \cdot \text{Powersof2}(sk)$$
$$= (\mu \cdot I + \text{BitDecomp}(R \cdot pk)) \cdot \text{Powersof2}(sk)$$

GSW's Construction - Decryption

$$
\begin{aligned}
&\mathrm{Flatten}(\mu \cdot I + \mathrm{BitDecomp}(R \cdot pk)) \cdot \mathrm{Powersof2}(sk) \\
&= (\mu \cdot I + \mathrm{BitDecomp}(R \cdot pk)) \cdot \mathrm{Powersof2}(sk) \\
&= \mu \mathrm{Powersof2}(sk) + R \cdot pk \cdot sk
\end{aligned}
$$

GSW's Construction - Decryption cont.

$\mu\text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.

GSW's Construction - Decryption cont.

$\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of sk is 1.

GSW's Construction - Decryption cont.

$\mu\text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of *sk* is 1.
- $\Rightarrow$ The first coordinates of the first term are $2^{l-1}\mu, 2^{l-2}\mu, \ldots, \mu$

GSW's Construction - Decryption cont.

$\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of sk is 1.
- $\Rightarrow$ The first coordinates of the first term are $2^{l-1}\mu, 2^{l-2}\mu, \ldots, \mu$
- Recover μ's least significant bit by $LSB(\mu) = 2^{l-1}\mu$

GSW's Construction - Decryption cont.

$\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of *sk* is 1.
- $\Rightarrow$ The first coordinates of the first term are $2^{l-1}\mu, 2^{l-2}\mu, \ldots, \mu$
- Recover μ's least significant bit by $LSB(\mu) = 2^{l-1}\mu$
- Recover μ's next bit by $2^{l-2}(\mu - LSB(\mu))$

GSW's Construction - Decryption cont.

$\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of sk is 1.
- $\Rightarrow$ The first coordinates of the first term are $2^{l-1}\mu, 2^{l-2}\mu, \ldots, \mu$
- Recover μ's least significant bit by $LSB(\mu) = 2^{l-1}\mu$
- Recover μ's next bit by $2^{l-2}(\mu - LSB(\mu))$
- Similar for all other bits of μ.

GSW's Construction - Decryption cont.

$\mu\text{Powersof2}(sk) + R \cdot pk \cdot sk$

- The second term is small.
- The first coordinate of sk is 1.
- $\Rightarrow$ The first coordinates of the first term are $2^{l-1}\mu, 2^{l-2}\mu, \ldots, \mu$
- Recover μ's least significant bit by $LSB(\mu) = 2^{l-1}\mu$
- Recover μ's next bit by $2^{l-2}(\mu - LSB(\mu))$
- Similar for all other bits of μ.
- Decryption breaks down when the error reaches $q/4$.

GSW's Construction - Security

- $\text{BitDecomp}^{-1}(C) = \mu \cdot \text{BitDecomp}^{-1}(I) + R \cdot A$

GSW's Construction - Security

- $\text{BitDecomp}^{-1}(C) = \mu \cdot \text{BitDecomp}^{-1}(I) + R \cdot A$
- Fact: The joint distribution $(A, R \cdot A)$ is indistinguishable from uniform, if $m > 2nl$

GSW's Construction - Security

- $\mathrm{BitDecomp}^{-1}(C) = \mu \cdot \mathrm{BitDecomp}^{-1}(I) + R \cdot A$
- Fact: The joint distribution $(A, R \cdot A)$ is indistinguishable from uniform, if $m > 2nl$
- $\Rightarrow \mathrm{BitDecomp}^{-1}(C)$ hides μ

GSW's Construction - Security

- $\text{BitDecomp}^{-1}(C) = \mu \cdot \text{BitDecomp}^{-1}(I) + R \cdot A$
- Fact: The joint distribution $(A, R \cdot A)$ is indistinguishable from uniform, if $m > 2nl$
- $\Rightarrow \text{BitDecomp}^{-1}(C)$ hides μ
- $C = \text{Flatten}(C) = \text{BitDecomp} \circ \text{BitDecomp}^{-1}(C)$ hides μ

Security

- What does it mean for an encryption scheme to be secure?

Security

- What does it mean for an encryption scheme to be secure?
- Chosen plaintext attack (CPA): The "intuitive" definition. An (efficient) adversary who's able to encrypt anything shouldn't be able to decrypt anything.

Security

- What does it mean for an encryption scheme to be secure?
- Chosen plaintext attack (CPA): The "intuitive" definition. An (efficient) adversary who's able to encrypt anything shouldn't be able to decrypt anything.
- Ciphertext indistinguishability

GSW's Construction - NAND

- $(I - C_1 \cdot C_2)\vec{v} = (1 - \mu_1\mu_2)\vec{v} - \mu_2\vec{e}_1 - C_1\vec{e}_2$

GSW's Construction - NAND

- $(I - C_1 \cdot C_2)\vec{v} = (1 - \mu_1\mu_2)\vec{v} - \mu_2\vec{e}_1 - C_1\vec{e}_2$
- Error increased by a factor of $N + 1$.

GSW's Construction - NAND

- $(I - C_1 \cdot C_2)\vec{v} = (1 - \mu_1\mu_2)\vec{v} - \mu_2\vec{e}_1 - C_1\vec{e}_2$
- Error increased by a factor of $N + 1$.
- Final error increase by a factor of $(N + 1)^L$

GSW's Construction - Multiplying by constant

Input: C, α

- Set $M_\alpha = \text{Flatten}(\alpha I)$

GSW's Construction - Multiplying by constant

Input: C, α

- Set $M_\alpha = \text{Flatten}(\alpha I)$
- Output $\text{Flatten}(M_\alpha \cdot C)$

GSW's Construction - Multiplying by constant

Input: C, α

- Set $M_\alpha = \text{Flatten}(\alpha I)$
- Output $\text{Flatten}(M_\alpha \cdot C)$
- Observe $M_\alpha \cdot C\vec{v} = M_\alpha \cdot (\mu\vec{v} + \vec{e}) = \alpha\mu\vec{v} + M_\alpha \cdot e$

GSW's Construction - Multiplying by constant

Input: C, α

- Set $M_\alpha = \text{Flatten}(\alpha I)$
- Output $\text{Flatten}(M_\alpha \cdot C)$
- Observe $M_\alpha \cdot C\vec{v} = M_\alpha \cdot (\mu\vec{v} + \vec{e}) = \alpha\mu\vec{v} + M_\alpha \cdot e$
- Error increases by a factor of N

GSW's Construction - Addition

- Output Flatten($C_1 + C_2$)

GSW's Construction - Addition

- Output Flatten($C_1 + C_2$)
- Error increases by a factor of 2

GSW's Construction - Multiplication

- Output Flatten($C_1 C_2$)

GSW's Construction - Multiplication

- Output Flatten($C_1 C_2$)
- $C_1 \cdot C_2 \vec{v} = C_1(\mu_2 \vec{v} + \vec{e}_2) = \mu_1 \mu_2 \vec{v} + \mu_2 \vec{e}_1 + C_1 \vec{e}_2$

GSW's Construction - Multiplication

- Output Flatten($C_1 C_2$)
- $C_1 \cdot C_2 \vec{v} = C_1(\mu_2 \vec{v} + \vec{e}_2) = \mu_1 \mu_2 \vec{v} + \mu_2 \vec{e}_1 + C_1 \vec{e}_2$
- Error increase depends on what's encrypted

GSW's Construction - Multiplication

- Output Flatten($C_1 C_2$)
- $C_1 \cdot C_2 \vec{v} = C_1(\mu_2 \vec{v} + \vec{e}_2) = \mu_1 \mu_2 \vec{v} + \mu_2 \vec{e}_1 + C_1 \vec{e}_2$
- Error increase depends on what's encrypted
- May need to assume bounds on the values being computed

Choosing parameters

- $\frac{q}{B} > 4(N+1)^L$ for NANDs

Choosing parameters

- $\frac{q}{B} > 4(N+1)^L$ for NANDs
- $\frac{q}{B} > 4(N+T)^L$ for arithmetic circuit, where T is the upper bound on plaintexts

Choosing parameters

- $\frac{q}{B} > 4(N+1)^L$ for NANDs
- $\frac{q}{B} > 4(N+T)^L$ for arithmetic circuit, where T is the upper bound on plaintexts
- N increases linearly with $\log \frac{q}{B}$ for LWE's security

Choosing parameters

- $\frac{q}{B} > 4(N+1)^L$ for NANDs
- $\frac{q}{B} > 4(N+T)^L$ for arithmetic circuit, where T is the upper bound on plaintexts
- N increases linearly with $\log \frac{q}{B}$ for LWE's security
- Pick (q, B, N) accordingly

References

- O. Regev. The Learning with Errors Problem.
- C. Gentry, A. Sahai, B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based.