

# 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  $\chi$  to be some discrete Gaussian distribution

# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:

# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row 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  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row 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  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:
  - pick coefficients  $\vec{a} \in \mathbb{Z}_q^n$  at random
  - Sample error  $e \leftarrow \chi$
  - Set  $b = \vec{a} \cdot \vec{s} + e$



# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:
  - pick coefficients  $\vec{a} \in \mathbb{Z}_q^n$  at random
  - Sample error  $e \leftarrow \chi$
  - Set  $b = \vec{a} \cdot \vec{s} + e$
  - Note that  $\begin{pmatrix} b & -\vec{a}^T \end{pmatrix} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} = e$

# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:
  - pick coefficients  $\vec{a} \in \mathbb{Z}_q^n$  at random
  - Sample error  $e \leftarrow \chi$
  - Set  $b = \vec{a} \cdot \vec{s} + e$
  - Note that  $(b \ -\vec{a}^T) \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} = e$
  - Output  $(b \ -\vec{a}^T)$

# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:
  - pick coefficients  $\vec{a} \in \mathbb{Z}_q^n$  at random
  - Sample error  $e \leftarrow \chi$
  - Set  $b = \vec{a} \cdot \vec{s} + e$
  - Note that  $(b \ -\vec{a}^T) \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} = e$
  - Output  $(b \ -\vec{a}^T)$
- Output:  $\vec{s}$

# Definition (LWE Problem)

- Pick uniformly random  $\vec{s} \in \mathbb{Z}_q^n$
- Take  $\chi$  to be some discrete Gaussian distribution
- Input:  $A \in \mathbb{Z}_q^{m \times (n+1)}$ ; sample each row as below:
  - pick coefficients  $\vec{a} \in \mathbb{Z}_q^n$  at random
  - Sample error  $e \leftarrow \chi$
  - Set  $b = \vec{a} \cdot \vec{s} + e$
  - Note that  $(b \quad -\vec{a}^T) \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} = e$
  - Output  $(b \quad -\vec{a}^T)$
- Output:  $\vec{s}$
- Note that  $A \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} = \vec{e}$

# Homomorphic encryption

Allows secure communication by using the tools below:

# Homomorphic encryption

Allows secure communication by using the tools below:

- $pk$ : “public key” for encryption  $\text{Enc}_{pk}$

# Homomorphic encryption

Allows secure communication by using the tools below:

- $pk$ : “public key” for encryption  $\text{Enc}_{pk}$
- $sk$ : “secret key” for decryption  $\text{Dec}_{sk}$

# Homomorphic encryption

Allows secure communication by using the tools below:

- $pk$ : “public key” for encryption  $\text{Enc}_{pk}$
- $sk$ : “secret key” for decryption  $\text{Dec}_{sk}$
- $evk$ : “evaluation key”



# Homomorphic encryption

Allows secure communication by using the tools below:

- $pk$ : “public key” for encryption  $\text{Enc}_{pk}$
- $sk$ : “secret key” for decryption  $\text{Dec}_{sk}$
- $evk$ : “evaluation key”
- $\text{Eval}_{evk}(f, \text{Enc}(\mu_1), \dots, \text{Enc}(\mu_n)) = \text{Enc}(f(\mu_1, \dots, \mu_n))$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$
- $sk = \vec{v}$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$
- $sk = \vec{v}$
- $C = \text{Enc}(\mu)$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$
- $sk = \vec{v}$
- $C = \text{Enc}(\mu)$
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$
- $sk = \vec{v}$
- $C = \text{Enc}(\mu)$
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$
- $(C_1 C_2)\vec{v} \approx (\lambda_1 \lambda_2)\vec{v}$

# Idea of GSW's homomorphic encryption

- $C\vec{v} \approx \mu\vec{v}$
- $sk = \vec{v}$
- $C = \text{Enc}(\mu)$
- $(C_1 + C_2)\vec{v} \approx (\lambda_1 + \lambda_2)\vec{v}$
- $(C_1 C_2)\vec{v} \approx (\lambda_1 \lambda_2)\vec{v}$
- $\text{Enc}(\neg(\mu_0 \wedge \mu_1)) = I - C_1 C_2.$

# GSW's homomorphic encryption - naive approach

- Take  $A\vec{s} = \vec{e}$  as given by LWE.



# GSW's homomorphic encryption - naive approach

- Take  $A\vec{s} = \vec{e}$  as given by LWE.
- Sample  $R \in \{0, 1\}^{(n+1) \times m}$  randomly

# GSW's homomorphic encryption - naive approach

- Take  $A\vec{s} = \vec{e}$  as given by LWE.
- Sample  $R \in \{0, 1\}^{(n+1) \times m}$  randomly
- Observe  $(\mu I_{n+1} + RA)\vec{s} \approx \mu\vec{s}$

# GSW's homomorphic encryption - naive approach

- Take  $A\vec{s} = \vec{e}$  as given by LWE.
- Sample  $R \in \{0, 1\}^{(n+1) \times m}$  randomly
- Observe  $(\mu I_{n+1} + RA)\vec{s} \approx \mu\vec{s}$
- Challenge: Error grows too quickly

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

# Terminology

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

# Terminology

- For the sake of illustration, set  $q = 2^4$ .
- 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  $\underline{1011} = 8 + 2 + 1 = 11$

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

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



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

- $\text{BitDecomp}(\underline{a_3 a_2 a_1 a_0}) = (a_3 \ a_2 \ a_1 \ a_0)$

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

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

- $\text{BitDecomp}^{-1}(1 \ 0 \ 0 \ 1) = \underline{1001}$

# GSW's tools - BitDecomp<sup>-1</sup>

- $\text{BitDecomp}^{-1}(1 \ 0 \ 0 \ 1) = \underline{1001}$
- $\text{BitDecomp}^{-1}(a_3 \ a_2 \ a_1 \ a_0) = \sum 2^i a_i$

# GSW's tools - $\text{BitDecomp}^{-1}$

- $\text{BitDecomp}^{-1} (1 \ 0 \ 0 \ 1) = \underline{1001}$
- $\text{BitDecomp}^{-1} (a_3 \ a_2 \ a_1 \ a_0) = \sum 2^i a_i$
- $\text{BitDecomp}^{-1} (0 \ 0 \ \underline{10} \ 0) = \underline{0100}$

# GSW's tools - $\text{BitDecomp}^{-1}$

- $\text{BitDecomp}^{-1} (1 \ 0 \ 0 \ 1) = \underline{1001}$
- $\text{BitDecomp}^{-1} (a_3 \ a_2 \ a_1 \ a_0) = \sum 2^i a_i$
- $\text{BitDecomp}^{-1} (0 \ 0 \ \underline{10} \ 0) = \underline{0100}$
- $A\text{Powersof2}(\vec{b}) = \text{BitDecomp}^{-1}(A)\vec{b}$



# GSW's tools (Flatten)

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

# GSW's tools (Flatten)

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

# GSW's tools (Flatten)

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

# GSW's tools (Flatten)

- $\text{Flatten} = \text{BitDecomp} \circ \text{BitDecomp}^{-1}$
- $\text{Flatten}(\underline{110} \quad \underline{101} \quad 1 \quad \underline{11})$
- $= \text{BitDecomp} \circ \text{BitDecomp}^{-1}(\underline{110} \quad \underline{101} \quad 1 \quad \underline{11})$
- $= (1 \quad 0 \quad 0 \quad 1)$

# GSW's tools (Flatten)

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

# GSW's tools (Flatten)

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

# GSW's Construction - Overview

$$\text{Flatten}(\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix}$$

# GSW's Construction - Overview

$$\begin{aligned} & \text{Flatten}(\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= (\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \end{aligned}$$



# GSW's Construction - Overview

$$\begin{aligned} & \text{Flatten}(\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= (\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} + \text{BitDecomp}(RA) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \end{aligned}$$

# GSW's Construction - Overview

$$\begin{aligned} & \text{Flatten}(\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= (\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} + \text{BitDecomp}(RA) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} + RA \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \end{aligned}$$

# GSW's Construction - Overview

$$\begin{aligned} & \text{Flatten}(\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= (\mu I_N + \text{BitDecomp}(RA)) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} + \text{BitDecomp}(RA) \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &= \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} + RA \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \\ &\approx \mu \text{Powersof2} \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix} \end{aligned}$$

# 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 Set  $sk = \begin{pmatrix} 1 \\ \vec{s} \end{pmatrix}$

# GSW's Construction - Public Keygen

- 1 Generate  $A \leftarrow \mathbb{Z}_q^{m \times n}$  uniformly

# GSW's Construction - Public Keygen

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

# GSW's Construction - Public Keygen

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

# GSW's Construction - Public Keygen

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

# GSW's Construction - Encryption

Input:  $\mu$

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

# GSW's Construction - Encryption

Input:  $\mu$

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

# GSW's Construction - Decryption

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



# GSW's Construction - Decryption

- We have  $\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.

# GSW's Construction - Decryption

- We have  $\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.

# GSW's Construction - Decryption

- We have  $\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.
- $\Rightarrow$  The first components of the first term are  $2^{l-1}\mu, 2^{l-2}\mu, \dots, \mu$

# GSW's Construction - Decryption

- We have  $\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.
- $\Rightarrow$  The first components of the first term are  $2^{l-1}\mu, 2^{l-2}\mu, \dots, \mu$
- Recover  $\mu$ 's least significant bit by  $LSB(\mu) = 2^{l-1}\mu$

# GSW's Construction - Decryption

- We have  $\mu \text{Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.
- $\Rightarrow$  The first components of the first term are  $2^{l-1}\mu, 2^{l-2}\mu, \dots, \mu$
- Recover  $\mu$ 's least significant bit by  $LSB(\mu) = 2^{l-1}\mu$
- Recover  $\mu$ 's next bit by  $2^{l-2}(\mu - LSB(\mu))$

# GSW's Construction - Decryption

- We have  $\mu \text{ Powers of } 2(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.
- $\Rightarrow$  The first components of the first term are  $2^{l-1}\mu, 2^{l-2}\mu, \dots, \mu$
- Recover  $\mu$ 's least significant bit by  $LSB(\mu) = 2^{l-1}\mu$
- Recover  $\mu$ 's next bit by  $2^{l-2}(\mu - LSB(\mu))$
- Similar for all other bits of  $\mu$ .

# GSW's Construction - Decryption

- We have  $\mu \text{ Powersof2}(sk) + R \cdot pk \cdot sk$
- The second term is small.
- The first coordinate of  $sk$  is 1.
- $\Rightarrow$  The first components of the first term are  $2^{l-1}\mu, 2^{l-2}\mu, \dots, \mu$
- Recover  $\mu$ 's least significant bit by  $LSB(\mu) = 2^{l-1}\mu$
- Recover  $\mu$ 's next bit by  $2^{l-2}(\mu - LSB(\mu))$
- Similar for all other bits of  $\mu$ .
- 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: If  $m$  is big enough,  $(A, R \cdot A)$  is computationally indistinguishable from uniform

# GSW's Construction - Security

- $\text{BitDecomp}^{-1}(C) = \mu \cdot \text{BitDecomp}^{-1}(I) + R \cdot A$
- Fact: If  $m$  is big enough,  $(A, R \cdot A)$  is computationally indistinguishable from uniform
- $\Rightarrow \text{BitDecomp}^{-1}(C)$  hides  $\mu$

# GSW's Construction - Security

- $\text{BitDecomp}^{-1}(C) = \mu \cdot \text{BitDecomp}^{-1}(I) + R \cdot A$
- Fact: If  $m$  is big enough,  $(A, R \cdot A)$  is computationally indistinguishable from uniform
- $\Rightarrow \text{BitDecomp}^{-1}(C)$  hides  $\mu$
- $\Rightarrow C = \text{Flatten}(C) = \text{BitDecomp} \circ \text{BitDecomp}^{-1}(C)$  hides  $\mu$

# GSW's Construction - Multiplying by constant

Input:  $C, \alpha$

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

# GSW's Construction - Multiplying by constant

Input:  $C, \alpha$

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

# GSW's Construction - Multiplying by constant

Input:  $C, \alpha$

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

# GSW's Construction - Multiplying by constant

Input:  $C, \alpha$

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

# GSW's Construction - NAND

- $\text{Enc}(\neg(\mu_1 \wedge \mu_2)) = \text{Flatten}(I_N - C_1 \cdot C_2)\vec{v}$

# GSW's Construction - NAND

- $\text{Enc}(\neg(\mu_1 \wedge \mu_2)) = \text{Flatten}(I_N - C_1 \cdot C_2)\vec{v}$
- Error increased by a factor of  $N + 1$ .

# GSW's Construction - NAND

- $\text{Enc}(\neg(\mu_1 \wedge \mu_2)) = \text{Flatten}(I_N - C_1 \cdot C_2)\vec{v}$
- Error increased by a factor of  $N + 1$ .
- In boolean circuits, final error increases by a factor of  $(N + 1)^L$



# 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

- 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.