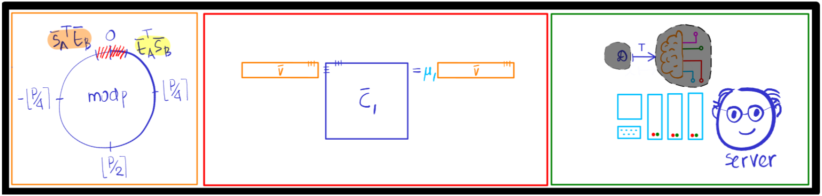


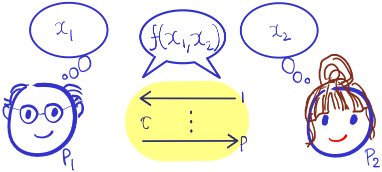
CS783: Theoretical Foundations of Cryptography

Lecture 19 (15/Oct/24)

Instructor: Chethan Kamath

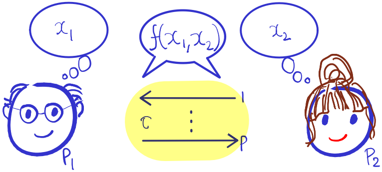
Recall from Last Two Lectures

- Task 6: Private computation of functions



Recall from Last Two Lectures

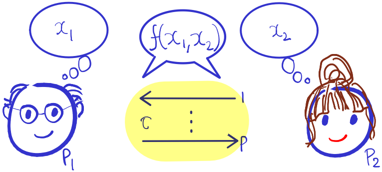
- Task 6: Private computation of functions



  - Perfectly-private MPC for *linear* functions over $\mathbb{F}_p$
  - Perfectly-private 2PC for $\wedge$ is impossible!
  - Computationally-private 2PC for general functions over $\mathbb{F}_2$

Recall from Last Two Lectures

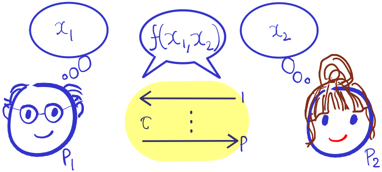
- Task 6: Private computation of functions



  - Perfectly-private MPC for *linear* functions over $\mathbb{F}_p$
  - Perfectly-private 2PC for $\wedge$ is impossible!
  - Computationally-private 2PC for general functions over $\mathbb{F}_2$
- Key tools:
  - Threshold secret sharing (TSS): privately computes $\oplus/+$
    - Construction: Shamir's TSS
    - Linearity: "sum of shares $\rightarrow$ shares of sum"
  - Oblivious transfer (OT): privately computes $\wedge/\cdot$
    - Trapdoor permutation (TDP) $\rightarrow$ OT

Recall from Last Two Lectures

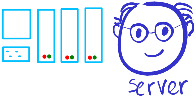
- Task 6: Private computation of functions



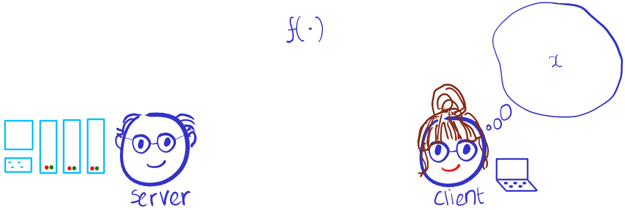
  - Perfectly-private MPC for *linear* functions over $\mathbb{F}_p$
  - Perfectly-private 2PC for $\wedge$ is impossible!
  - Computationally-private 2PC for general functions over $\mathbb{F}_2$
- Key tools:
  - Threshold secret sharing (TSS): privately computes $\oplus/+$
    - Construction: Shamir's TSS
    - Linearity: "sum of shares $\rightarrow$ shares of sum"
  - Oblivious transfer (OT): privately computes $\wedge/\cdot$
    - Trapdoor permutation (TDP) $\rightarrow$ OT
- Key idea: "computing over secret shares"

Plan for Today's Lecture...

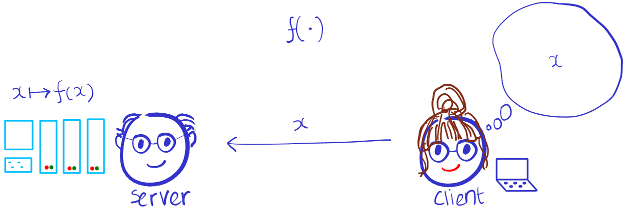
- Task 7: secure outsourcing in the client-server setting



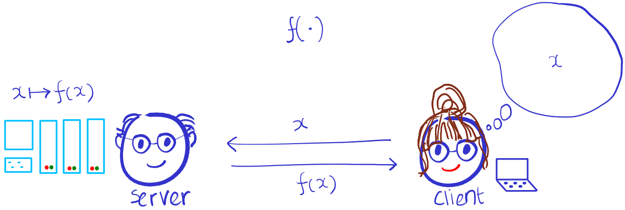
- Task 7: secure outsourcing in the client-server setting



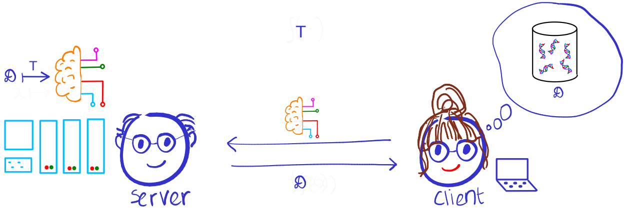
- Task 7: secure outsourcing in the client-server setting



- Task 7: secure outsourcing in the client-server setting

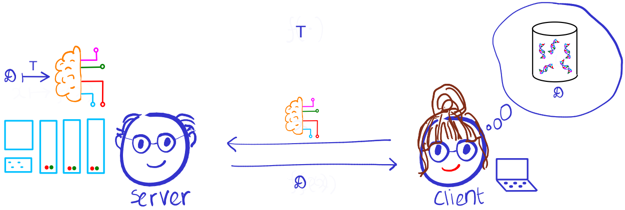


- Task 7: secure outsourcing in the client-server setting

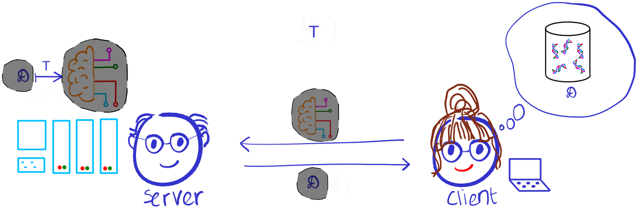


Plan for Today's Lecture...

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting

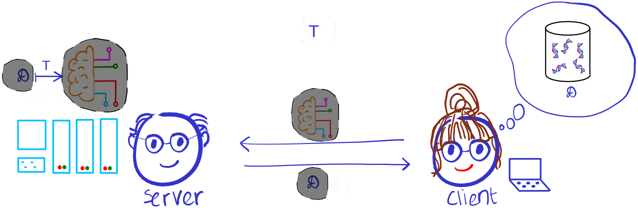


- Task 7: secure outsourcing in the client-server setting
    - Task 7.a: *private* outsourcing in the client-server setting



Plan for Today's Lecture...

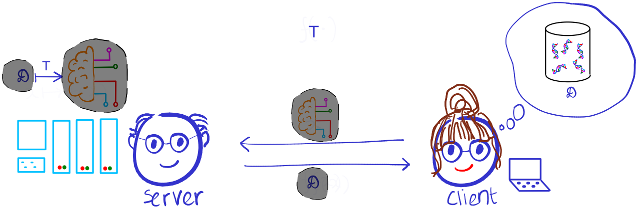
- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting



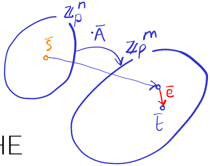
- Key tool: homomorphic (public-key) encryption
  - Operation on ciphertext $\implies$ operation on plaintext
  - Fully homomorphic encryption (FHE)
  - Private outsourcing of computation using FHE

Plan for Today's Lecture...

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting



- Key tool: homomorphic (public-key) encryption
  - Operation on ciphertext $\implies$ operation on plaintext
  - Fully homomorphic encryption (FHE)
  - Private outsourcing of computation using FHE
- FHE from learning with errors (LWE) assumption
  - Recall LWE and Regev's encryption
  - Gentry-Sahai-Waters construction of (levelled) FHE



Plan for Today's Lecture...

General *template*:   Task 7.a: private outsourcing

1. Identify the task
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

Plan for Today's Lecture...

General *template*:

Task 7.a: private outsourcing

1. Identify the task
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
     Honest-but-curious server
   - Security Goal: What does it mean to be secure?
     computational privacy
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

Plan for Today's Lecture...

General *template*:

1. Identify the task → Task 7.a: private outsourcing
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities? → Honest-but-curious server
   - Security Goal: What does it mean to be secure? → computational privacy
3. Construct a scheme Π FHE-based protocol
4. Formally prove that Π in secure in model M

   ↳ FHE IND-CPA secure $\Rightarrow$ Π private

Plan for Today's Lecture

1. Private Outsourcing of Computation
2. Fully-Homomorphic Encryption (FHE)
3. Gentry-Sahai-Waters FHE from Learning with Errors

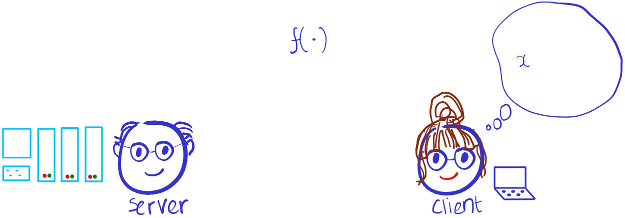
Plan for Today's Lecture

1. Private Outsourcing of Computation

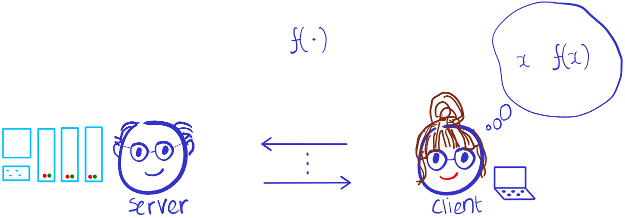
2. Fully-Homomorphic Encryption (FHE)

3. Gentry-Sahai-Waters FHE from Learning with Errors

Private Outsourcing: Setting and Security

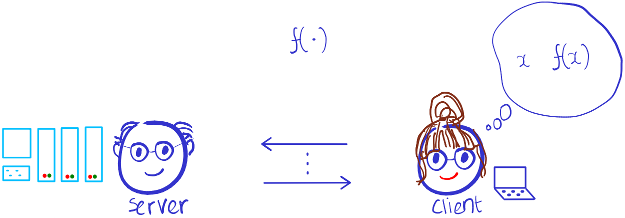


- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x



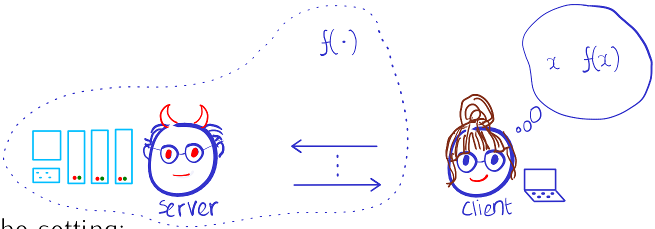
- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x
  - Server and client interact; in the end client locally outputs f(x)

Private Outsourcing: Setting and Security

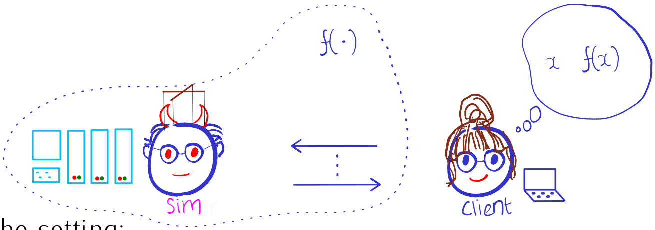


- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x
  - Server and client interact; in the end client locally outputs f(x)
- Security model: privacy against honest-but-curious server
  - Computational privacy of client's input: there exists a simulator **Sim** for (honest) server's view

Private Outsourcing: Setting and Security

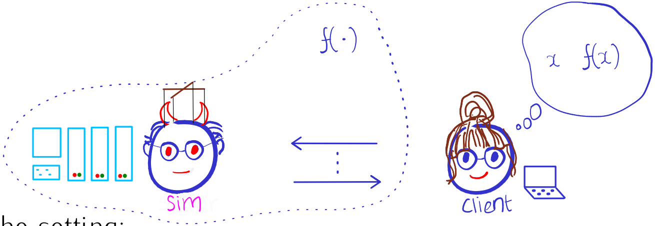


- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x
  - Server and client interact; in the end client locally outputs f(x)
- Security model: privacy against honest-but-curious server
  - Computational privacy of client's input: there exists a simulator **Sim** for (honest) server's view



- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x
  - Server and client interact; in the end client locally outputs f(x)
- Security model: privacy against honest-but-curious server
  - Computational privacy of client's input: there exists a simulator **Sim** for (honest) server's view

Private Outsourcing: Setting and Security



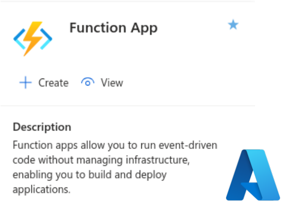
- The setting:
  - Client is resource constrained and server is powerful
  - Function f known to both client and server
    - Alternatively: f is known only to server (=2PC)
  - Client's local input is x
  - Server and client interact; in the end client locally outputs f(x)
- Security model: privacy against honest-but-curious server
  - Computational privacy of client's input: there exists a simulator **Sim** for (honest) server's view

Exercise 1

Why is private outsourcing trivial from a 2PC perspective?

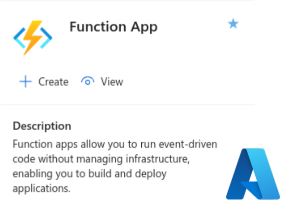
Why is it Useful?

- Compute as a service:



Why is it Useful?

- Compute as a service:



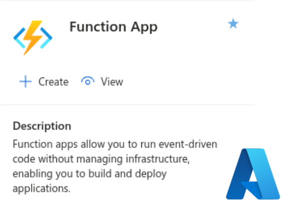
- We want: *private/confidential* compute as a service
  - Current solutions: some form of trusted hardware (TPM/HSM)

CONFIDENTIAL COMPUTING CONSORTIUM

Confidential Computing: Hardware-Based Trusted Execution for Applications and Data

Why is it Useful?

- Compute as a service:



- We want: *private/confidential* compute as a service
  - Current solutions: some form of trusted hardware (TPM/HSM)

CONFIDENTIAL COMPUTING CONSORTIUM

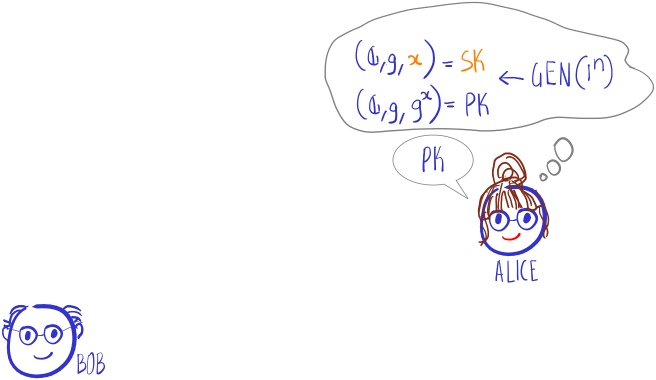
Confidential Computing: Hardware-Based Trusted Execution for Applications and Data

  - Private outsourcing avoids trusted hardware

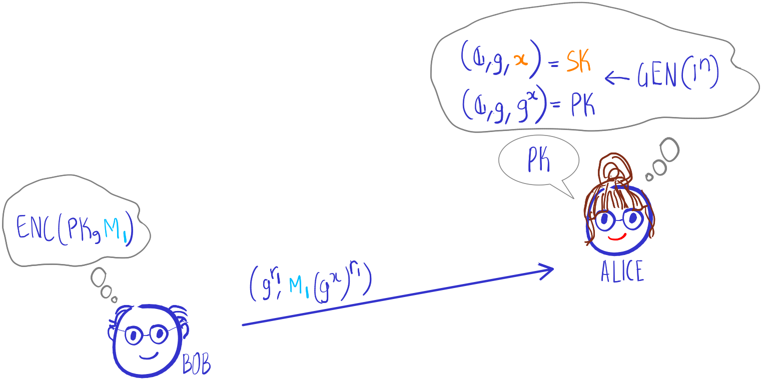
Plan for Today's Lecture

Recall the PKEs We Saw in Lecture 9...

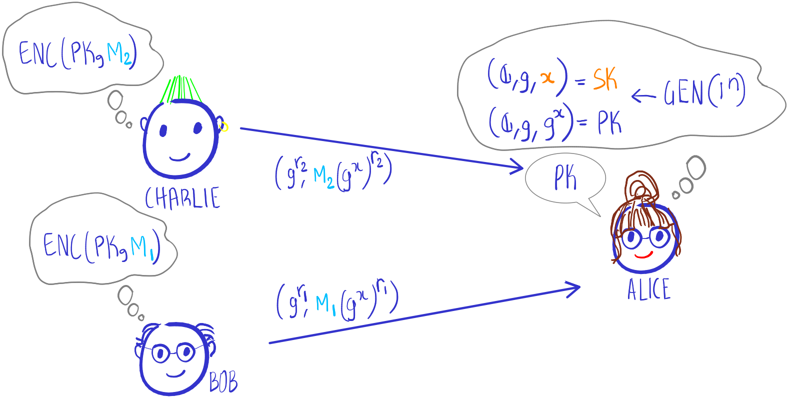
- PKE 1: Elgamal encryption



- PKE 1: Elgamal encryption



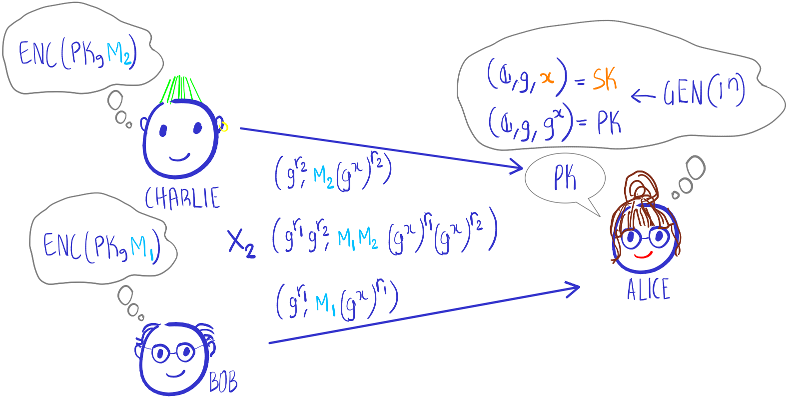
- PKE 1: Elgamal encryption



What happens when you multiply two ciphertexts?

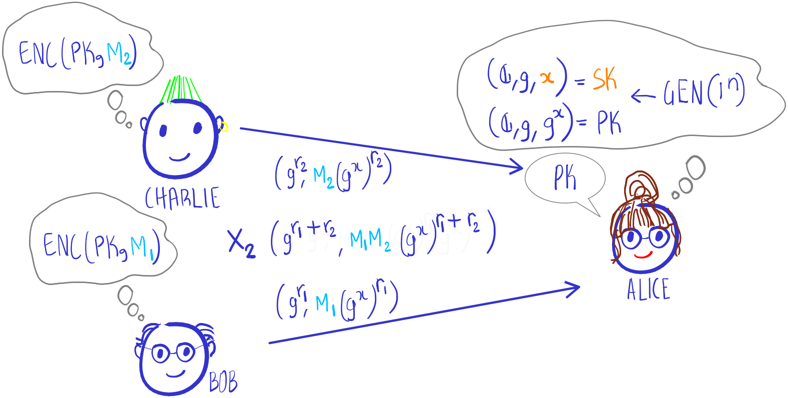
Recall the PKEs We Saw in Lecture 9...

- PKE 1: Elgamal encryption



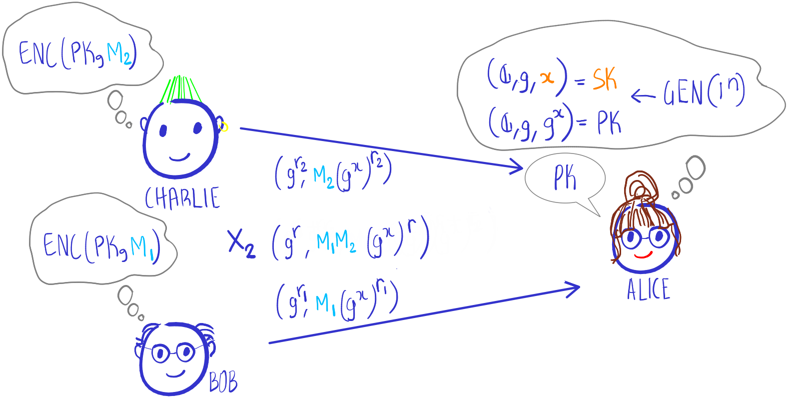
? What happens when you multiply two ciphertexts?

- PKE 1: Elgamal encryption



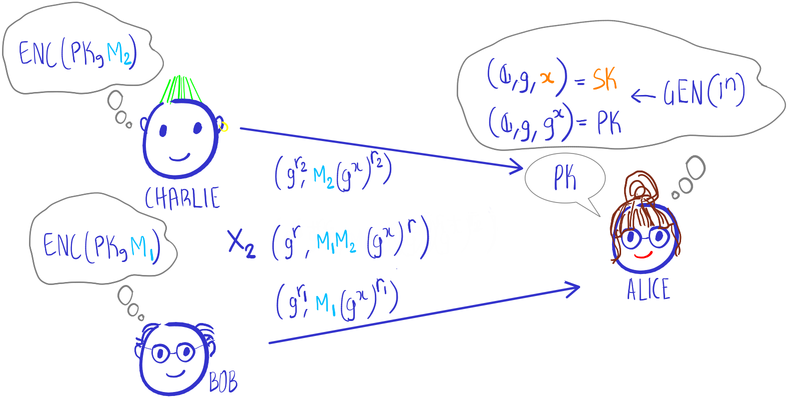
What happens when you multiply two ciphertexts?

- PKE 1: Elgamal encryption



What happens when you multiply two ciphertexts?

- PKE 1: Elgamal encryption

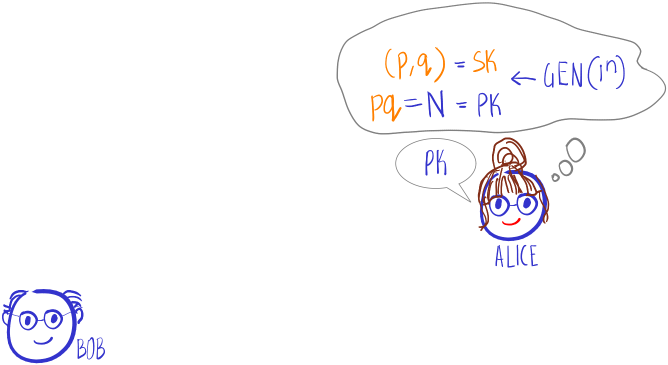


What happens when you multiply two ciphertexts?

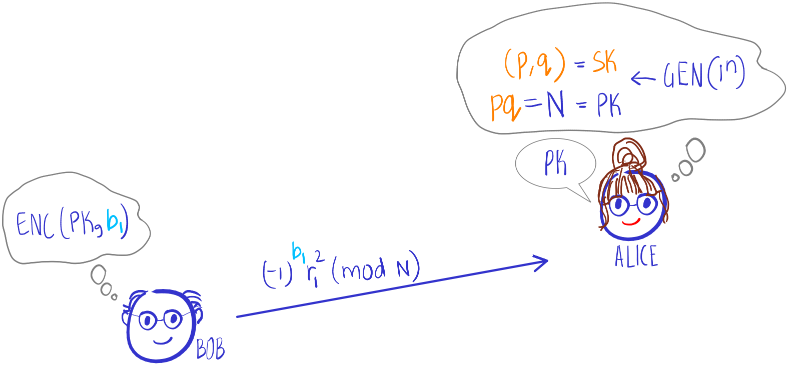
Is it possible to compute sum of plaintexts modulo p?

Recall the PKEs We Saw in Lecture 9...

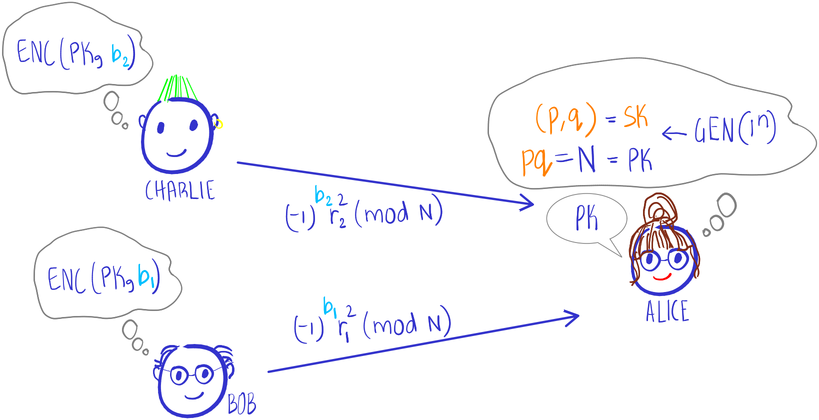
- PKE 2: Goldwasser-Micali bit encryption



- PKE 2: Goldwasser-Micali bit encryption

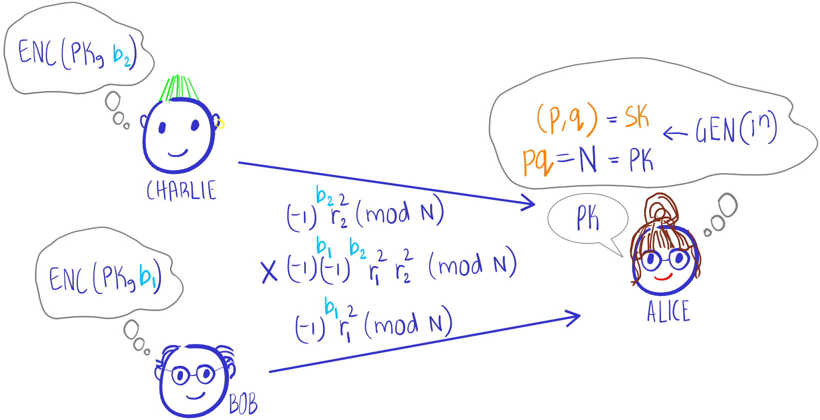


- PKE 2: Goldwasser-Micali bit encryption



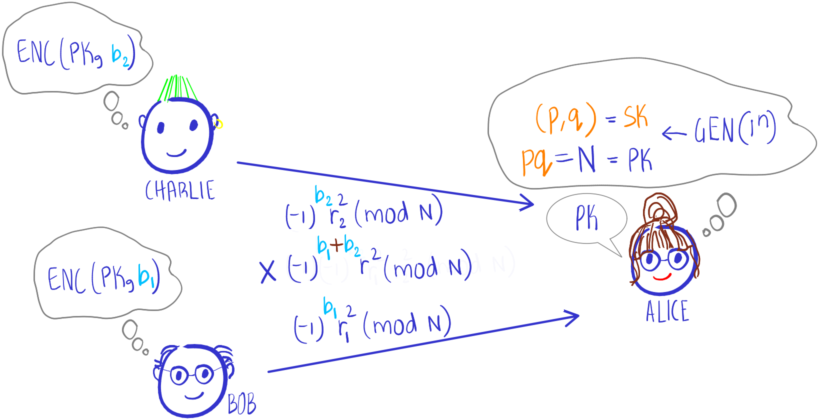
What happens when you multiply ciphertexts?

- PKE 2: Goldwasser-Micali bit encryption



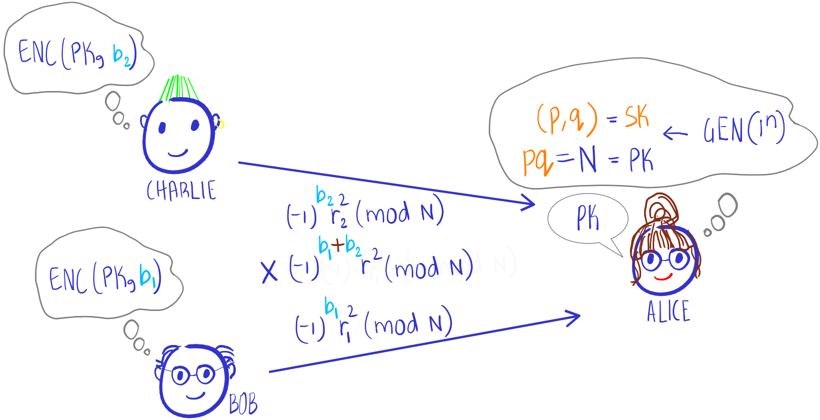
What happens when you multiply ciphertexts?

- PKE 2: Goldwasser–Micali bit encryption



What happens when you multiply ciphertexts?

- PKE 2: Goldwasser–Micali bit encryption



What happens when you multiply ciphertexts?

Is it possible compute $\wedge$ of plaintexts?

Let's Define Homomorphic Encryption...

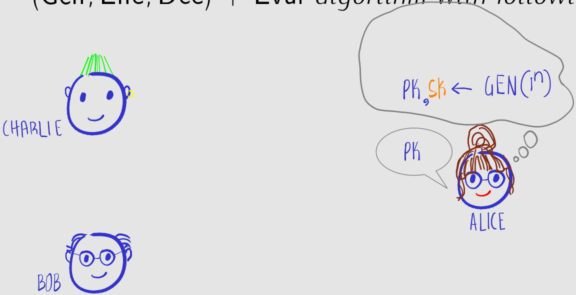
- Public-key encryption + public *evaluation* algorithm

Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*

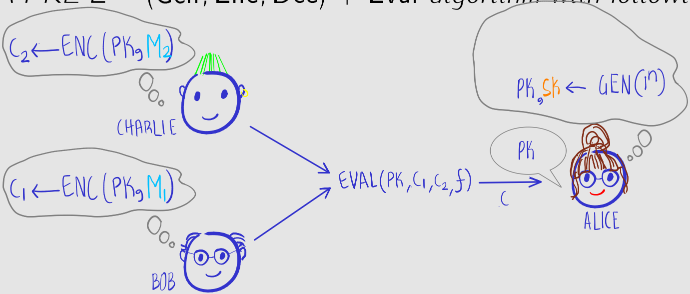


Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*

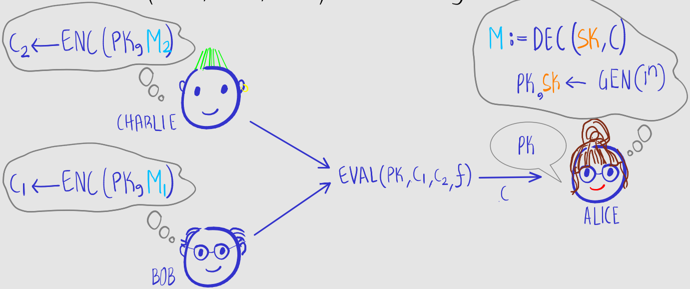


Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*

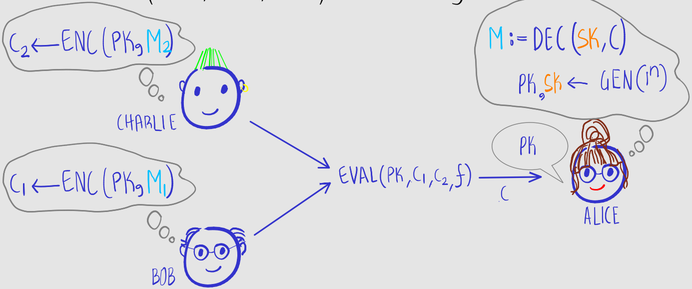


Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*



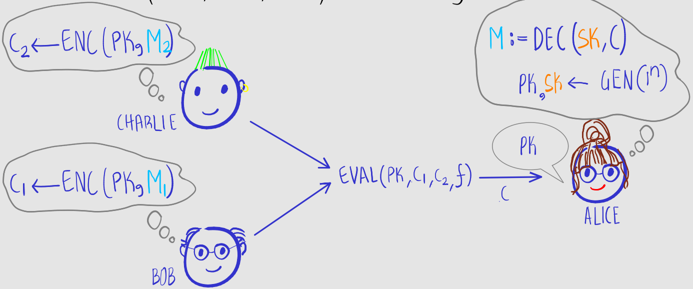
- *Compactness of evaluation:* $|c|$ *obtained from* Eval *independent of* $|f|$
- *Correctness of evaluation*

Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*



- *Compactness of evaluation:* $|c|$ *obtained from* Eval *independent of* $|f|$
- *Correctness of evaluation*

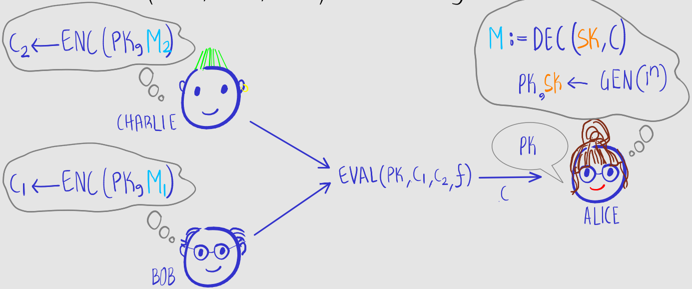
- *Fully* HE: $\mathcal{F}$=functions computable by poly.-sized circuits
  - We will represent f using a Boolean circuit of NAND gates

Let's Define Homomorphic Encryption...

- Public-key encryption + public *evaluation* algorithm

Defintion 1 (Homomorphic encryption (HE) for function class $\mathcal{F}$)

A PKE $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) + \mathsf{Eval}$ *algorithm with following syntax*



- *Compactness of evaluation:* $|c|$ *obtained from* Eval *independent of* $|f|$
- *Correctness of evaluation*

- *Fully* HE: $\mathcal{F}$=functions computable by poly.-sized circuits
  - We will represent f using a Boolean circuit of NAND gates
- Levelled FHE: $\mathcal{F}$=functions computable by depth L circuits

Let's Define Homomorphic Encryption...

- Security model: same as PKE (IND-CPA)

Let's Define Homomorphic Encryption...

- Security model: same as PKE (IND-CPA)

Defintion 2 (CPA Secrecy for FHE)

An FHE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is CPA-secret if for every PPT eavesdropper Eve, the following is negligible:*

$$\delta(n) := \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_1)}}[\mathsf{Eve}(c)=0]$$

Let's Define Homomorphic Encryption...

- Security model: same as PKE (IND-CPA)

Defintion 2 (CPA Secrecy for FHE)

An FHE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is CPA-secret if for every PPT eavesdropper Eve, the following is negligible:*

$$\delta(n) := \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_1)}}[\mathsf{Eve}(c)=0]$$

"Left world" "Right world"

Let's Define Homomorphic Encryption...

- Security model: same as PKE (IND-CPA)

Defintion 2 (CPA Secrecy for FHE)

An FHE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is CPA-secret if for every PPT eavesdropper Eve, the following is negligible:

$$\delta(n) := \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(pk,sk)\leftarrow\mathsf{Gen}(1^n)\\(m_0,m_1)\leftarrow\mathsf{Eve}(pk)\\c\leftarrow\mathsf{Enc}(pk,m_1)}}[\mathsf{Eve}(c)=0]$$

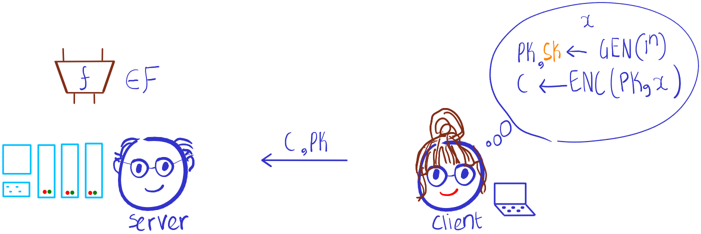
"Left world" "Right world"

Exercise 2 (Recall: IND-CCA=IND-CPA+decryption oracle)

Can FHE be IND-CCA secure?

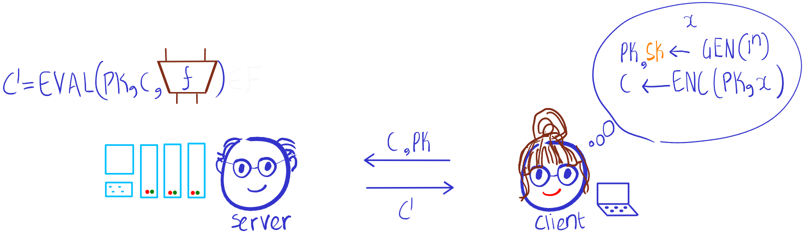
How to Privately Outsource using FHE?

How to Privately Outsource using FHE?



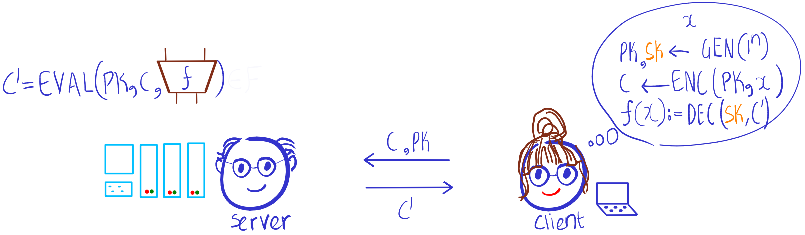
1. Client:
   1. Generate FHE public-secret key-pair $(\boldsymbol{pk}, \boldsymbol{sk})$
   2. Encrypt input x using $\boldsymbol{pk}$ to get ciphertext $\boldsymbol{c}$; send it to server

How to Privately Outsource using FHE?



1. Client:
   1. Generate FHE public-secret key-pair (pk, sk)
   2. Encrypt input x using pk to get ciphertext c; send it to server
2. Server:
   1. Use **Eval** to run f on c and obtain encrypted output c'
   2. Send c' to client

How to Privately Outsource using FHE?



1. Client:
   1. Generate FHE public-secret key-pair (pk, sk)
   2. Encrypt input x using pk to get ciphertext c; send it to server
2. Server:
   1. Use **Eval** to run f on c and obtain encrypted output c'
   2. Send c' to client
3. Client: decrypt c' using sk to retrieve output f(x)

Exercise 3

Prove that the above protocol is private if FHE is IND-CPA secure

Plan for this Session

1 Private Outsourcing of Computation

2 Fully-Homomorphic Encryption (FHE)

3 Gentry-Sahai-Waters FHE from Learning with Errors

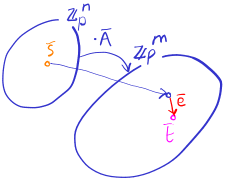
Let's Recall LWE from Lecture 10

- Solving "noisy" linear equations over $(\mathbb{Z}_p, +, \cdot)$ is hard

Let's Recall LWE from Lecture 10

- Solving "noisy" linear equations over $(\mathbb{Z}_p, +, \cdot)$ is hard
  - Input $(\bar{A}, \bar{t})$, where $\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\bar{s} \leftarrow \mathbb{Z}_p^n$, $\bar{e} \leftarrow E^m$ and

$$\bar{t}^\top := \bar{s}^\top \bar{A} + \bar{e}^\top \bmod p$$

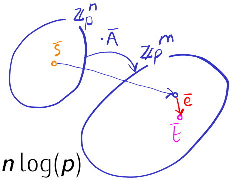


  - Solution: $\bar{s}$

Let's Recall LWE from Lecture 10

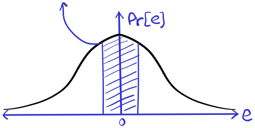
- Solving "noisy" linear equations over $(\mathbb{Z}_p, +, \cdot)$ is hard
  - Input $(\bar{A}, \bar{t})$, where $\bar{A} \leftarrow \mathbb{Z}_p^{n\times m}$, $\bar{s} \leftarrow \mathbb{Z}_p^n$, $\bar{e} \leftarrow \mathrm{E}^m$ and

$$\bar{t}^\top := \bar{s}^\top \bar{A} + \bar{e}^\top \bmod p$$



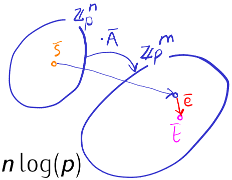
  - Solution: $\bar{s}$
- Usual parameters:
  - n=security parameter, $p = \mathsf{poly}(n)$ and $m \approx n\log(p)$
  - Noise distribution $\mathrm{E} = \mathrm{E}_\alpha$, the *discrete Gaussian distribution* over $\mathbb{Z}$
    - Centred at 0; parameter $\alpha < 1$ determines s.d. $\sigma := \alpha p \approx n$

$$\Pr[e] = \frac{1}{\sqrt{2\pi}\sigma \cdot \exp e^2/2\sigma^2}$$

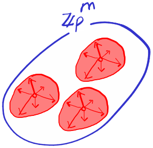


- Solving "noisy" linear equations over $(\mathbb{Z}_p, +, \cdot)$ is hard
    - Input $(\bar{A}, \bar{t})$, where $\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\bar{s} \leftarrow \mathbb{Z}_p^n$, $\bar{e} \leftarrow \mathrm{E}^m$ and

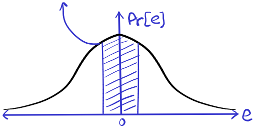
$$\bar{t}^\top := \bar{s}^\top \bar{A} + \bar{e}^\top \bmod p$$



- Solution: $\bar{s}$
- Usual parameters:
    - n=security parameter, $p = \mathrm{poly}(n)$ and $m \approx n \log(p)$
    - Noise distribution $\mathrm{E} = \mathrm{E}_\alpha$, the *discrete Gaussian distribution* over $\mathbb{Z}$
        - Centred at 0; parameter $\alpha < 1$ determines s.d. $\sigma := \alpha p \approx n$



$$\Pr[e] = \frac{1}{\sqrt{2\pi}\sigma \cdot \exp e^2/2\sigma^2}$$

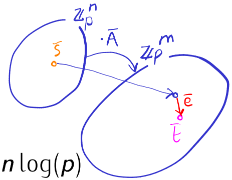


- $\bar{t}$ "determines" $\bar{s}$, but efficient algorithm to recover $\bar{s}$ not known

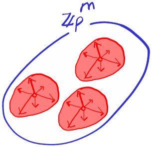
Let's Recall LWE from Lecture 10

- Solving "noisy" linear equations over $(\mathbb{Z}_p, +, \cdot)$ is hard
  - Input $(\bar{A}, \bar{t})$, where $\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\bar{s} \leftarrow \mathbb{Z}_p^n$, $\bar{e} \leftarrow \mathrm{E}^m$ and

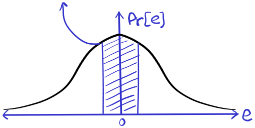
$$\bar{t}^\top := \bar{s}^\top \bar{A} + \bar{e}^\top \bmod p$$



  - Solution: $\bar{s}$
- Usual parameters:
  - n=security parameter, $p = \mathbf{poly}(n)$ and $m \approx n \log(p)$
  - Noise distribution $\mathrm{E} = \mathrm{E}_\alpha$, the *discrete Gaussian distribution* over $\mathbb{Z}$
    - Centred at 0; parameter $\alpha < 1$ determines s.d. $\sigma := \alpha p \approx n$



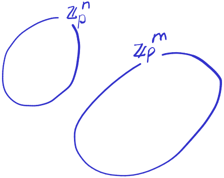
$$Pr[e] = \frac{1}{\sqrt{2\pi}\sigma \cdot \exp e^2/2\sigma^2}$$

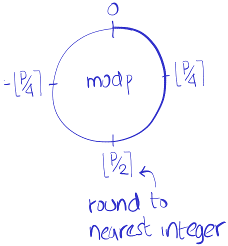


  - $\bar{t}$ "determines" $\bar{s}$, but efficient algorithm to recover $\bar{s}$ not known
- Decision LWE (DLWE): $(\bar{A}, \bar{t}) \approx (\bar{A}, \bar{r})$, where $\bar{r} \leftarrow \mathbb{Z}_p^m$

Let's Recall Regev's PKE from Lecture 10...

- "Noisy/approximate" 1-bit key-exchange based on DLWE:



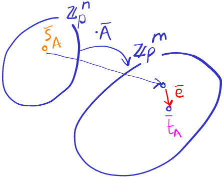


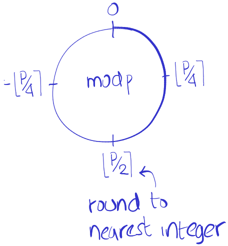
- "Noisy/approximate" 1-bit key-exchange based on DLWE:

LWE instance

$\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\bar{s}_A \leftarrow \mathbb{Z}_p^n$

$\bar{t}_A^T := \bar{s}_A^T \bar{A} + \bar{e}^T$





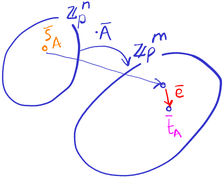
Let's Recall Regev's PKE from Lecture 10...

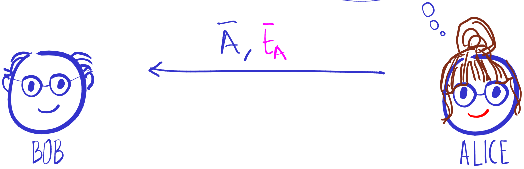
- "Noisy/approximate" 1-bit key-exchange based on DLWE:

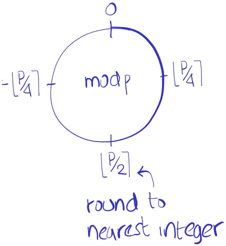
LWE instance

$$\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}, \quad \bar{s}_A \leftarrow \mathbb{Z}_p^{n}$$

$$\bar{t}_A^T := \bar{s}_A^T \bar{A} + \bar{e}^T$$







Let's Recall Regev's PKE from Lecture 10...

- "Noisy/approximate" **1**-bit key-exchange based on DLWE:

$\bar{s}_B \leftarrow \{0,1\}^m$

$\bar{t}_B := \bar{A}\bar{s}_B$, $b_B \leftarrow \{0,1\}$

$c := \bar{t}_A \bar{s}_B + b_B \lfloor p/2 \rceil$

"noisy key"

exchanged bit

LWE instance

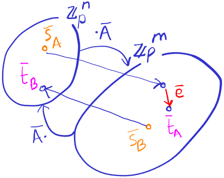
$\bar{A} \leftarrow \mathbb{Z}_p^{n \times m}$, $\bar{s}_A \leftarrow \mathbb{Z}_p^n$

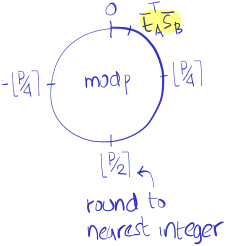
$\bar{t}_A^T := \bar{s}_A^T \bar{A} + \bar{e}^T$

BOB

$\bar{A}, \bar{t}_A$ $\longleftarrow$

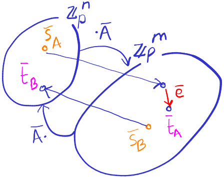
ALICE





Let's Recall Regev's PKE from Lecture 10...

- "Noisy/approximate" **1**-bit key-exchange based on DLWE:



LWE instance

$\bar{A} \leftarrow \mathbb{Z}_p^{n\times m}$, $\bar{s}_A \leftarrow \mathbb{Z}_p^{n}$

$\bar{t}_A^T := \bar{s}_A^T \bar{A} + \bar{e}^T$

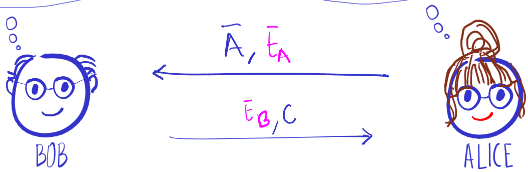
$\bar{s}_B \leftarrow \{0,1\}^m$

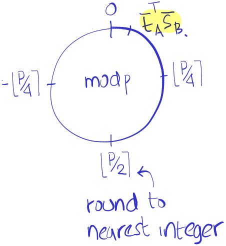
$\bar{t}_B := \bar{A}\bar{s}_B$, $b_B \leftarrow \{0,1\}$

$c := \bar{t}_A^T\bar{s}_B + b_B \lfloor p/2 \rceil$

"noisy key"

exchanged bit





- "Noisy/approximate" **1**-bit key-exchange based on DLWE:

$\bar{s}_B \leftarrow \{0,1\}^m$

$\bar{t}_B := \bar{A}\bar{s}_B$, $b_B \leftarrow \{0,1\}$

$c := \bar{t}_A\bar{s}_B + b_B \lfloor p/2 \rceil$

"noisy key"

exchanged bit

LWE instance

$\bar{A} \leftarrow \mathbb{Z}_p^{n\times m}$, $\bar{s}_A \leftarrow \mathbb{Z}_p^{n}$

$\bar{t}_A^T := \bar{s}_A^T \bar{A} + \bar{e}^T$

$b_A := \lfloor c - \bar{s}_A^T \bar{t}_B \rceil_2$

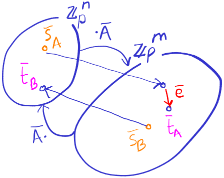
"noisy key"

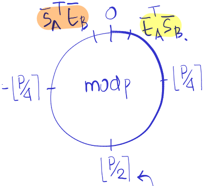
BOB

$\bar{A}, \bar{t}_A$

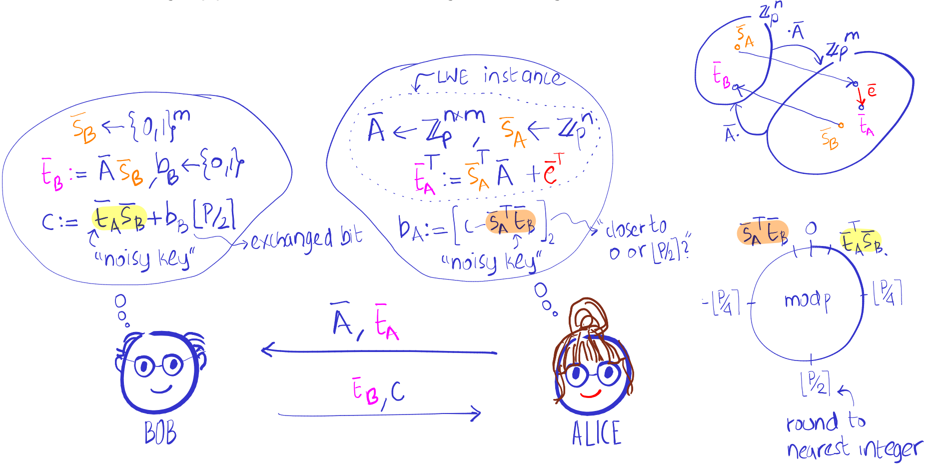
$\bar{t}_B, c$

ALICE

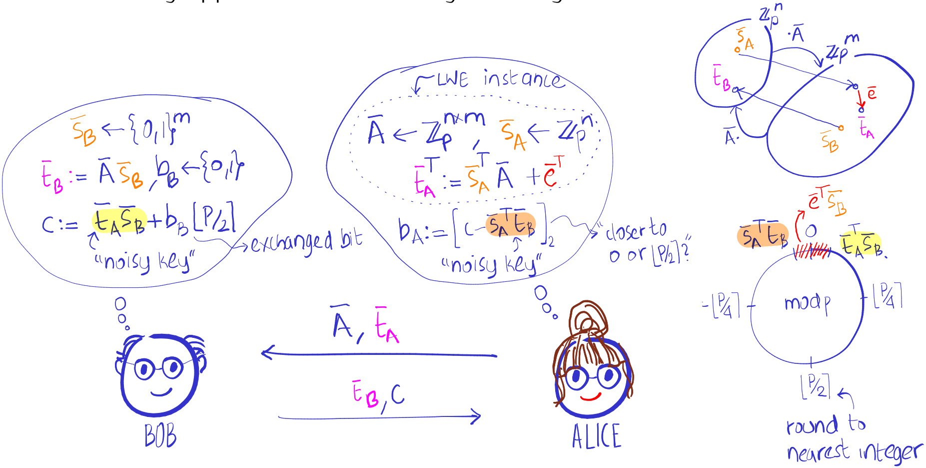




- "Noisy/approximate" **1**-bit key-exchange based on DLWE:

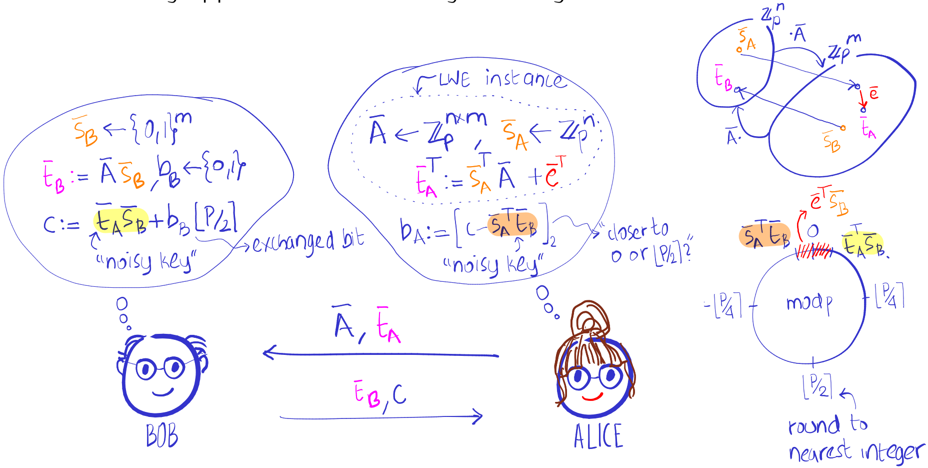


- "Noisy/approximate" **1**-bit key-exchange based on DLWE:



Let's Recall Regev's PKE from Lecture 10...

- "Noisy/approximate" 1-bit key-exchange based on DLWE:



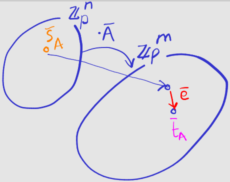
- Regev's PKE obtained by generic transformation

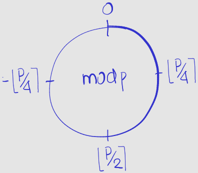
Construction 1 (Regev's PKE for parameters n, m, p and E_α)

- *Key generation* $\mathsf{Gen}(1^n; \bar{A}, \bar{s}_A, \bar{e})$*:*

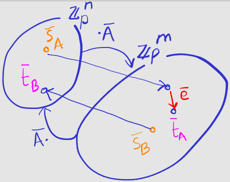
$$pk := \begin{pmatrix} \bar{A} \\ \bar{t}_A^\top := \bar{s}_A^\top \bar{A} + \bar{e}^\top \bmod p \end{pmatrix} \qquad sk := \bar{s}_A$$

LWE instance





Construction 1 (Regev's PKE for parameters n, m, p and E_α)



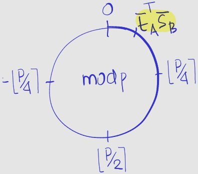
- *Key generation* $\mathsf{Gen}(1^n; \bar{A}, \bar{s}_A, \bar{e})$:

$$pk := \begin{pmatrix} \bar{A} \\ \bar{t}_A^\top := \bar{s}_A^\top \bar{A} + \bar{e}^\top \bmod p \end{pmatrix} \qquad sk := \bar{s}_A$$

LWE instance

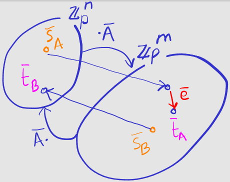
- *Encryption* $\mathsf{Enc}(pk, b; \bar{s}_B)$:

$$\bar{c} := pk\,\bar{s}_B + \begin{pmatrix} 0^n \\ b \cdot \lfloor p/2 \rceil \end{pmatrix} = \begin{pmatrix} \bar{t}_B := \bar{A}\bar{s}_B \\ \bar{t}_A^\top \bar{s}_B + b \cdot \lfloor p/2 \rceil \end{pmatrix} \bmod p$$



Let's Recall Regev's PKE from Lecture 10...

Construction 1 (Regev's PKE for parameters n, m, p and E_α)



- *Key generation* $\mathsf{Gen}(1^n; \bar{A}, \bar{s}_A, \bar{e})$:

$$pk := \begin{pmatrix} \bar{A} \\ \bar{t}_A^\top := \bar{s}_A^\top \bar{A} + \bar{e}^\top \bmod p \end{pmatrix} \qquad sk := \bar{s}_A$$

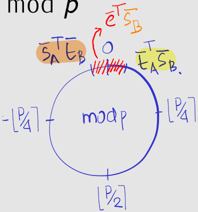
↖LWE instance

- *Encryption* $\mathsf{Enc}(pk, b; \bar{s}_B)$:

$$\bar{c} := pk\bar{s}_B + \begin{pmatrix} 0^n \\ b \cdot \lfloor p/2 \rceil \end{pmatrix} = \begin{pmatrix} \bar{t}_B := \bar{A}\bar{s}_B \\ \bar{t}_A^\top \bar{s}_B + b \cdot \lfloor p/2 \rceil \end{pmatrix} \bmod p$$

- *Decryption* $\mathsf{Dec}(sk, \bar{c})$:

$$b' := [(-\bar{s}_A^\top, 1)\bar{c} = \bar{e}^\top \bar{s}_B + b \cdot \lfloor p/2 \rceil \bmod p]_2$$



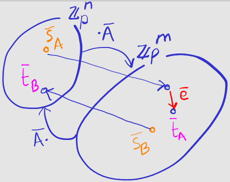
Let's Recall Regev's PKE from Lecture 10...

Construction 1 (Regev's PKE for parameters n, m, p and E_α)

- *Key generation* $\mathsf{Gen}(1^n; \bar{A}, \bar{s}_A, \bar{e})$:

$$pk := \begin{pmatrix} \bar{A} \\ \bar{t}_A^\top := \bar{s}_A^\top \bar{A} + \bar{e}^\top \bmod p \end{pmatrix} \qquad sk := \bar{s}_A$$

↖LWE instance

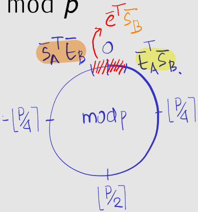


- *Encryption* $\mathsf{Enc}(pk, b; \bar{s}_B)$:

$$\bar{c} := pk\bar{s}_B + \begin{pmatrix} 0^n \\ b \cdot \lfloor p/2 \rceil \end{pmatrix} = \begin{pmatrix} \bar{t}_B := \bar{A}\bar{s}_B \\ \bar{t}_A^\top \bar{s}_B + b \cdot \lfloor p/2 \rceil \end{pmatrix} \bmod p$$

- *Decryption* $\mathsf{Dec}(sk, \bar{c})$:

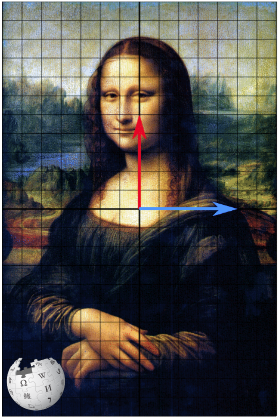
$$b' := [(-\bar{s}_A^\top, 1)\bar{c} = \bar{e}^\top \bar{s}_B + b \cdot \lfloor p/2 \rceil \bmod p]_2$$

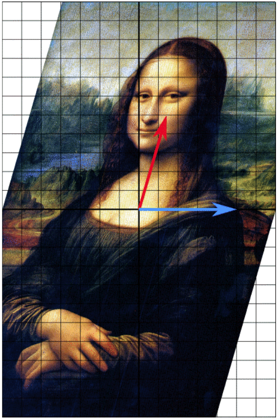


❓ What happens when you add two ciphertexts?

First Attempt: "Eigenvector" Encryption...

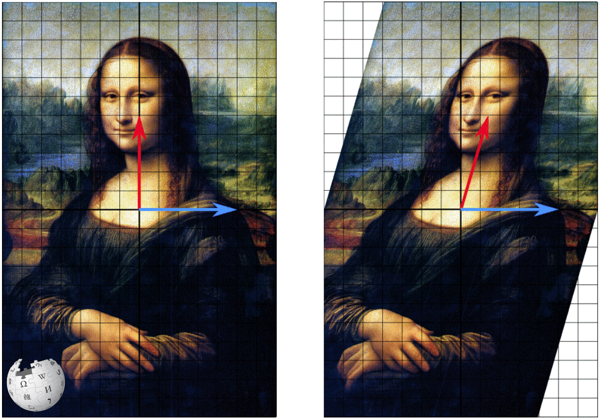
- Let's recall eigenvectors





First Attempt: "Eigenvector" Encryption...

- Let's recall eigenvectors

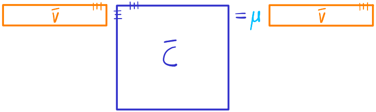


Definition 1 (Eigenvectors for matrices over $\mathbb{F}_p$)

A (left) eigenvector of a square matrix $\bar{C}$ is a vector $\bar{v}$ such that $\bar{v}\,\bar{C} = \mu\bar{v}$ for some scalar μ, which is the eigenvalue.

First Attempt: "Eigenvector" Encryption...

- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



First Attempt: "Eigenvector" Encryption...

- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$

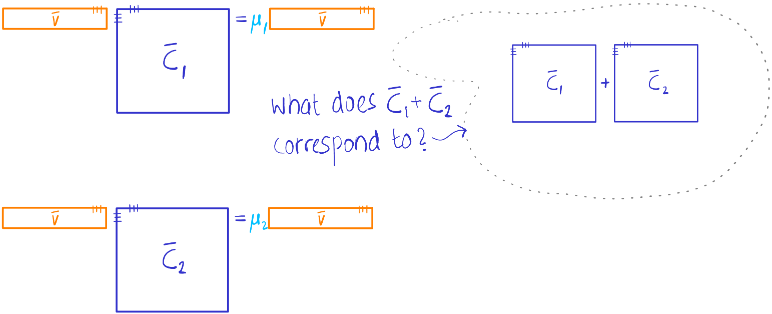
$$\bar{v}\,\bar{C}_1 = \mu_1 \bar{v}$$

what does $\bar{C}_1 + \bar{C}_2$ correspond to?

$$\bar{v}\,\bar{C}_2 = \mu_2 \bar{v}$$

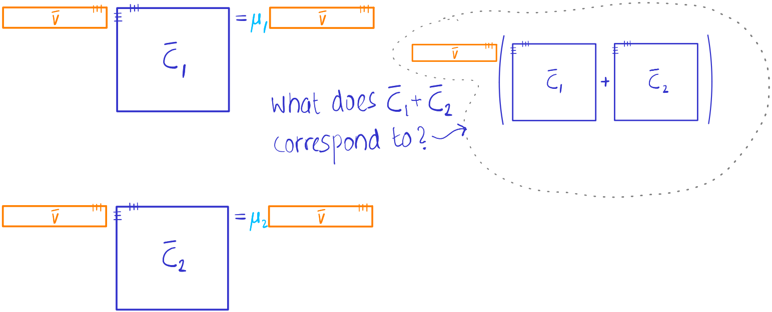
First Attempt: “Eigenvector” Encryption...

- Invariant: $n \times n$ “ciphertext” matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



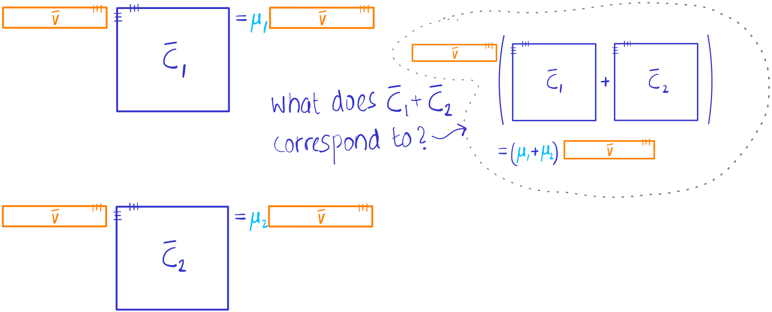
First Attempt: “Eigenvector” Encryption...

- Invariant: $n \times n$ “ciphertext” matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



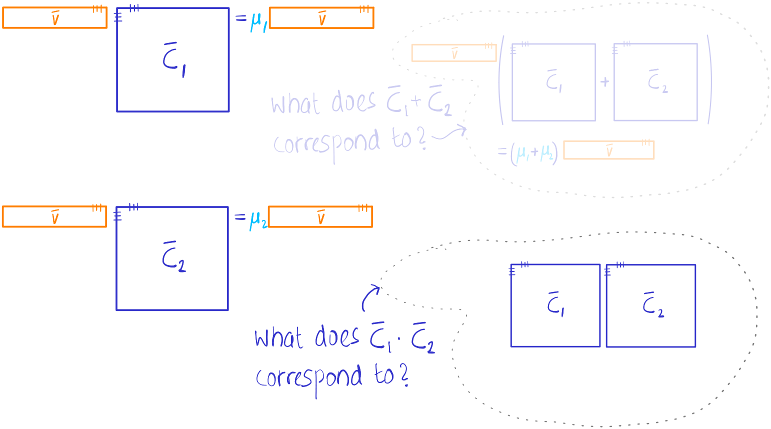
First Attempt: "Eigenvector" Encryption...

- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$

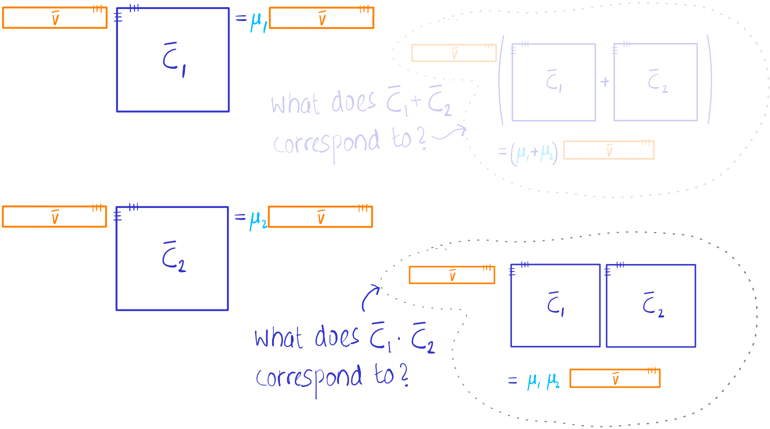


First Attempt: "Eigenvector" Encryption...

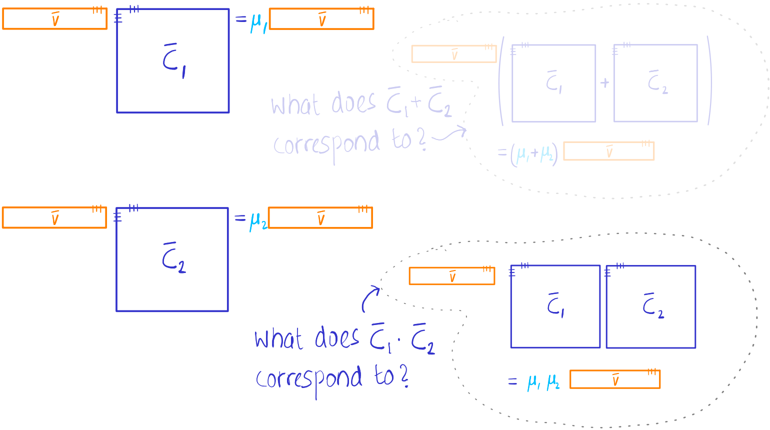
- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$

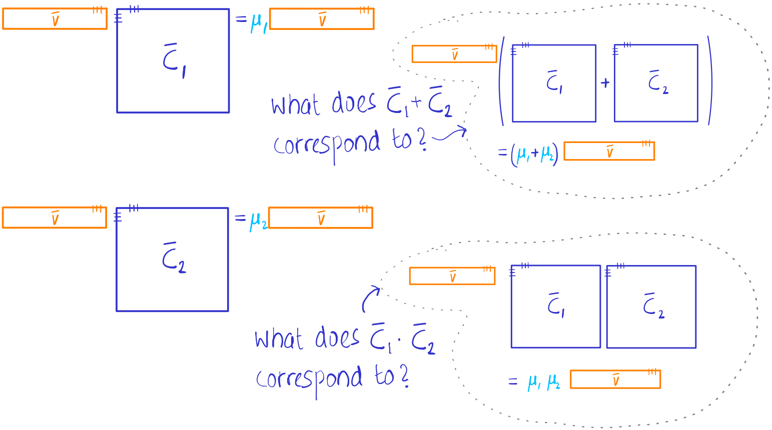


- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



First Attempt: "Eigenvector" Encryption...

- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$



Do we have an FHE?

First Attempt: "Eigenvector" Encryption...

- Invariant: $n \times n$ "ciphertext" matrix $\bar{C}$ encrypts bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} = \mu\bar{v}$

$\bar{v}\,\bar{C}_1 = \mu_1 \bar{v}$

$\bar{v}\,\bar{C}_2 = \mu_2 \bar{v}$

what does $\bar{C}_1 + \bar{C}_2$ correspond to?

$\bar{v}\left(\bar{C}_1 + \bar{C}_2\right) = (\mu_1 + \mu_2)\,\bar{v}$

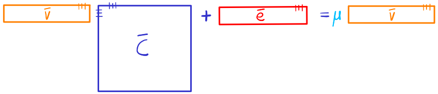
what does $\bar{C}_1 \cdot \bar{C}_2$ correspond to?

$\bar{v}\,\bar{C}_1\,\bar{C}_2 = \mu_1 \mu_2\,\bar{v}$

Do we have an FHE? No, can break by Gaussian elimination

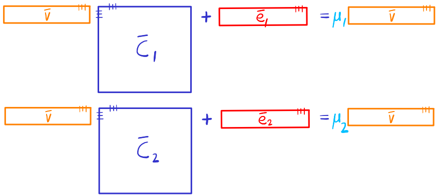
Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



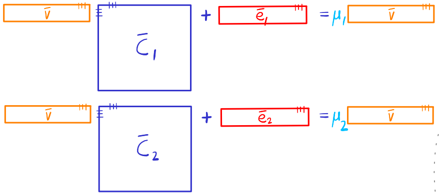
Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)

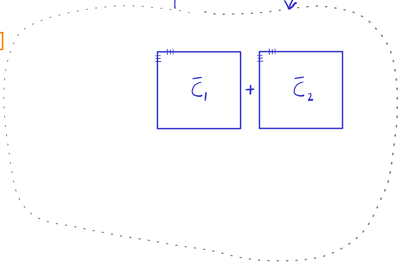


Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)

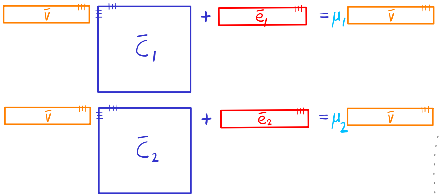


what does $\bar{C}_1 + \bar{C}_2$ correspond to?

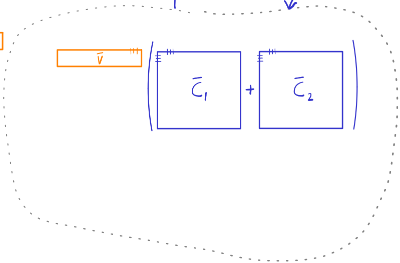


Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)

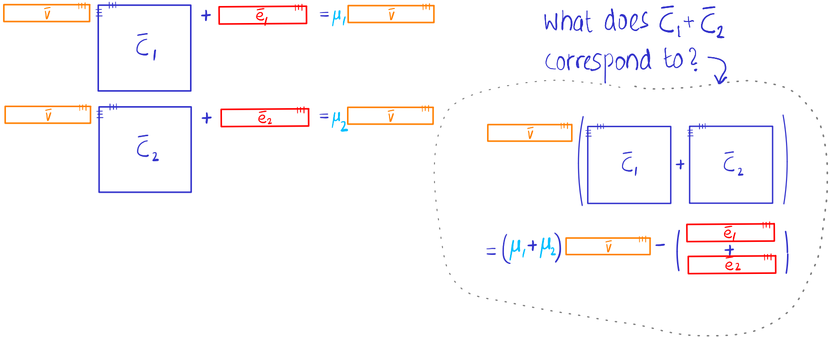


what does $\bar{C}_1 + \bar{C}_2$ correspond to?



Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)

$$\bar{v}\,\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}$$

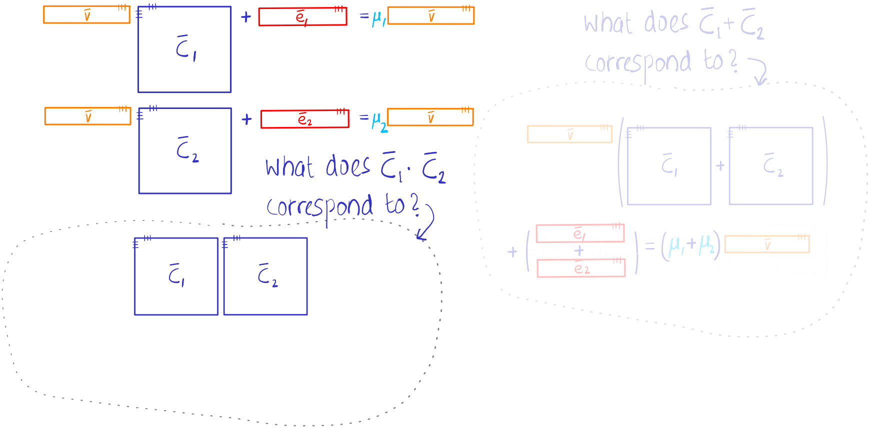
$$\bar{v}\,\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}$$

what does $\bar{C}_1 + \bar{C}_2$ correspond to?

$$\bar{v}\left(\bar{C}_1 + \bar{C}_2\right) + \left(\bar{e}_1 + \bar{e}_2\right) = (\mu_1 + \mu_2)\,\bar{v}$$

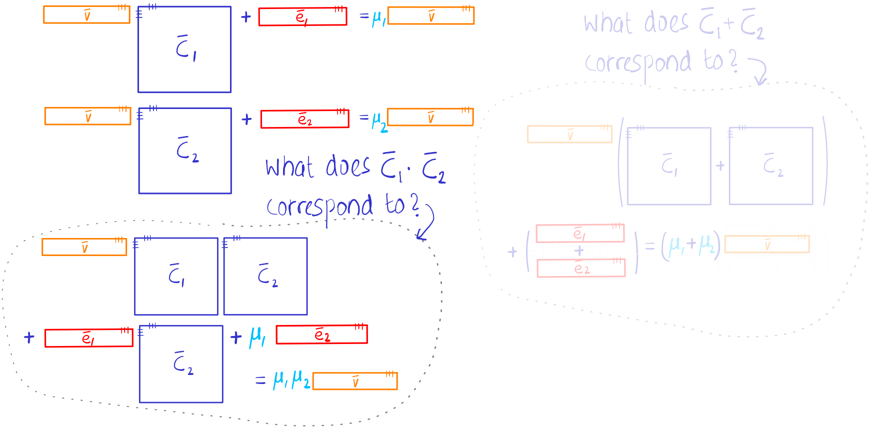
Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



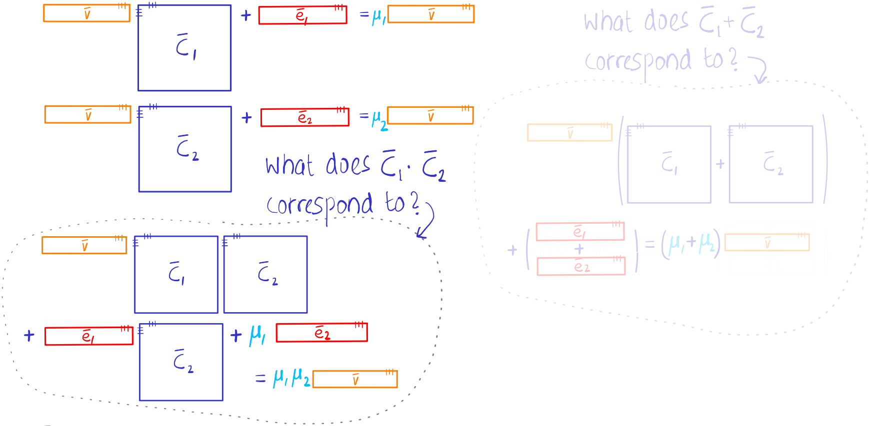
Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



Second Attempt: *Approximate* Eigenvector Encryption

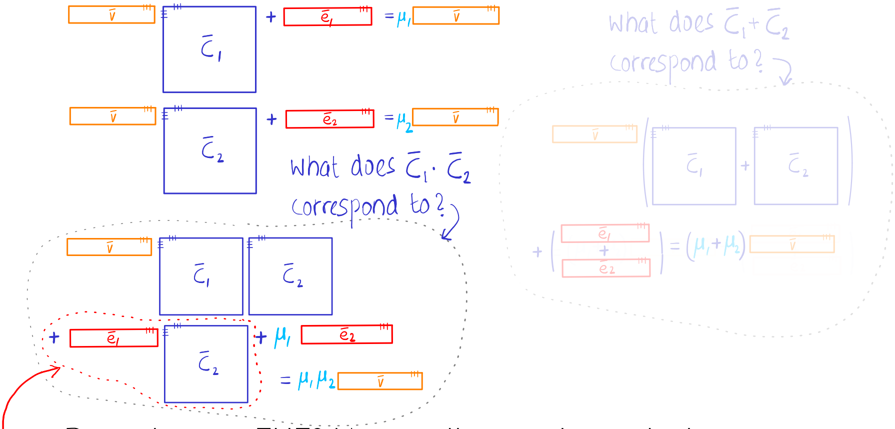
- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



- Do we have an FHE?

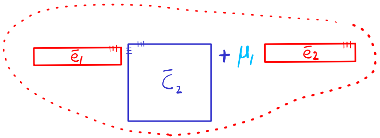
Second Attempt: *Approximate* Eigenvector Encryption

- New invariant: $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}$ for "short" $\bar{e}$ (as in LWE)



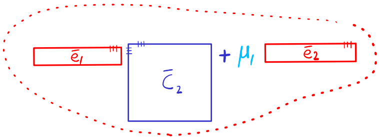
- Do we have an FHE? No, overall error when multiplying depends $\bar{C}_2$ which can be large (even if $\bar{e}_1$ is small)

Gentry-Sahai-Waters FHE...



Solution: use $\bar{C}_1 \cdot \mathrm{G}^{-1}(\bar{C}_2)$ for homomorphic multiplication

- $\mathrm{G}^{-1} : \mathbb{Z}_p^{n \times n} \to \mathbb{Z}_p^{n \times n\ell}$ is the bit-decomposition function
  ℓ: $\log(q)$
  1. $\mathrm{G}^{-1}(\bar{C}_2)$ has small entries (i.e., low infinity norm)
  2. Linear algebra we carried out before still holds



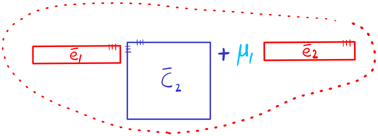
Solution: use $\bar{C}_1 \cdot \mathrm{G}^{-1}(\bar{C}_2)$ for homomorphic multiplication

- $\mathrm{G}^{-1} : \mathbb{Z}_p^{n \times n} \to \mathbb{Z}_p^{n \times n\ell}$ is the bit-decomposition function

  $\ell \hookrightarrow \log(q)$

  1. $\mathrm{G}^{-1}(\bar{C}_2)$ has small entries (i.e., low infinity norm)
  2. Linear algebra we carried out before still holds

$$\sum_{k \in [\ell]} a_{11k} 2^k = a_{11}$$



Solution: use $\bar{C}_1 \cdot \mathrm{G}^{-1}(\bar{C}_2)$ for homomorphic multiplication

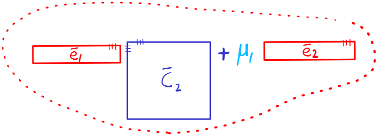
- $\mathrm{G}^{-1} : \mathbb{Z}_p^{n \times n} \to \mathbb{Z}_p^{n \times n\ell}$ is the bit-decomposition function

  ℓ ↳ $\log(q)$

  1. $\mathrm{G}^{-1}(\bar{C}_2)$ has small entries (i.e., low infinity norm)
  2. Linear algebra we carried out before still holds

$\sum_{k \in [\ell]} a_{11k} 2^k = a_{11}$ $\boxed{a_{111} \cdots a_{11\ell}}$

Gentry-Sahai-Waters FHE...

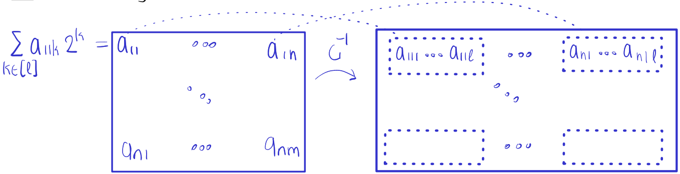


Solution: use $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ for homomorphic multiplication

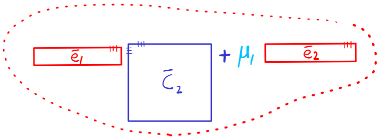
- $G^{-1} : \mathbb{Z}_p^{n \times n} \to \mathbb{Z}_p^{n \times n\ell}$ is the bit-decomposition function

  ℓ: log(q)

  1. $G^{-1}(\bar{C}_2)$ has small entries (i.e., low infinity norm)
  2. Linear algebra we carried out before still holds

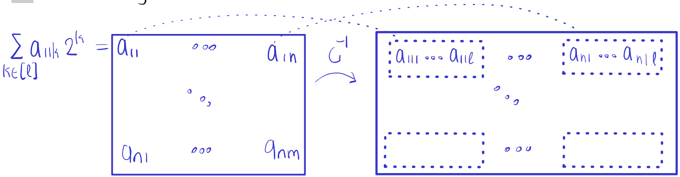


Gentry-Sahai-Waters FHE...



Solution: use $\bar{C}_1 \cdot \mathrm{G}^{-1}(\bar{C}_2)$ for homomorphic multiplication

- $\mathrm{G}^{-1} : \mathbb{Z}_p^{n \times n} \to \mathbb{Z}_p^{n \times n\ell}$ is the bit-decomposition function
  ℓ ↳ $\log(q)$
  1. $\mathrm{G}^{-1}(\bar{C}_2)$ has small entries (i.e., low infinity norm)
  2. Linear algebra we carried out before still holds



- G^{-1}'s inverse computed using gadget matrix $\bar{G} : \mathbb{Z}_p^{n \times n\ell} \to \mathbb{Z}_p^{n \times n}$
  - $\forall \bar{v} : \bar{G}\mathrm{G}^{-1}(\bar{v}) = \bar{v}$

Gentry-Sahai-Waters FHE...

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu \bar{v} \bar{G}$ for "short" $\bar{e}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v} \bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v} \bar{G}$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \stackrel{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

what does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1\bar{v}\bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2\bar{v}\bar{G}$

Gentry-Sahai-Waters FHE...

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}\bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}\bar{G}$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to?

Gentry-Sahai-Waters FHE...

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu \bar{v} \bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}\bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}\bar{G}$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$ $\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) \overset{*}{=} (\mu_1 \bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2)$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1\bar{v}\bar{G} \qquad \bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2\bar{v}\bar{G}$$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) \overset{*}{=} (\mu_1\bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2)$$
$$= \mu_1\bar{v}\bar{G} \cdot G^{-1}(\bar{C}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2)$$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

what does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1\bar{v}\bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2\bar{v}\bar{G}$

what does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\begin{aligned} \bar{v}\cdot\bar{C}_1\cdot G^{-1}(\bar{C}_2) &\overset{*}{=} (\mu_1\bar{v}\bar{G} - \bar{e}_1)\cdot G^{-1}(\bar{C}_2) \\ &= \mu_1\bar{v}\cancel{\bar{G}\cdot G^{-1}}(\bar{C}_2) - \bar{e}_1\cdot G^{-1}(\bar{C}_2) \\ &= \mu_1\bar{v}\bar{C}_2 - \bar{e}_1\cdot G^{-1}(\bar{C}_2) \end{aligned}$$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}\bar{G} \qquad \bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}\bar{G}$$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\begin{aligned}
\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) &\overset{*}{=} (\mu_1 \bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v} \cancel{\bar{G} \cdot G^{-1}}(\bar{C}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v} \bar{C}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&\overset{*}{=} \mu_1 (\mu_2 \bar{v}\bar{G} - \bar{e}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2)
\end{aligned}$$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1\bar{v}\bar{G}$ $\qquad$ $\bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2\bar{v}\bar{G}$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\begin{aligned}
\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) &\overset{*}{=} (\mu_1\bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1\bar{v}\cancel{\bar{G}G^{-1}}(\bar{C}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1\bar{v}\bar{C}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&\overset{*}{=} \mu_1(\mu_2\bar{v}\bar{G} - \bar{e}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1\mu_2\bar{v}\bar{G} - \mu_1\bar{e}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2)
\end{aligned}$$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}\bar{G} \qquad \bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}\bar{G}$$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\begin{aligned}
\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) &\overset{*}{=} (\mu_1 \bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v}\cancel{\bar{G} \cdot G^{-1}}(\bar{C}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v}\bar{C}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&\overset{*}{=} \mu_1(\mu_2 \bar{v}\bar{G} - \bar{e}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \mu_2 \bar{v}\bar{G} - \boxed{\mu_1 \bar{e}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2)} \leftarrow \text{short!}
\end{aligned}$$

- New invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} \overset{*}{=} \mu\bar{v}\bar{G}$ for "short" $\bar{e}$

What does $\bar{C}_1 + \bar{C}_2$ correspond to? $\rightsquigarrow$ $\bar{v}(\bar{C}_1 + \bar{C}_2) + (\bar{e}_1 + \bar{e}_2) = (\mu_1 + \mu_2)\bar{v}\bar{G}$

$$\bar{v}\bar{C}_1 + \bar{e}_1 = \mu_1 \bar{v}\bar{G} \qquad \bar{v}\bar{C}_2 + \bar{e}_2 = \mu_2 \bar{v}\bar{G}$$

What does $\bar{C}_1 \cdot G^{-1}(\bar{C}_2)$ correspond to? $\rightsquigarrow$

$$\begin{aligned}
\bar{v} \cdot \bar{C}_1 \cdot G^{-1}(\bar{C}_2) &\overset{*}{=} (\mu_1 \bar{v}\bar{G} - \bar{e}_1) \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v} \cancel{\bar{G} \cdot G^{-1}}(\bar{C}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \bar{v}\bar{C}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&\overset{*}{=} \mu_1(\mu_2 \bar{v}\bar{G} - \bar{e}_2) - \bar{e}_1 \cdot G^{-1}(\bar{C}_2) \\
&= \mu_1 \mu_2 \bar{v}\bar{G} - \boxed{\mu_1 \bar{e}_2 - \bar{e}_1 \cdot G^{-1}(\bar{C}_2)} \leftarrow \text{short!}
\end{aligned}$$

- How does the noise grow? Entries of μ_1, $\bar{e}_1$, $\bar{e}_2$ and $\bar{C}_2$ are at most $B \implies$ new noise at most $N \cdot B^2$

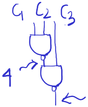
Putting it All Together

- Invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}\bar{G}$ for "short" $\bar{e}$
  - Secret key of the form $\bar{v} \in \mathbb{Z}_p^n$
  - Ciphertexts of the form $\bar{C} \in \mathbb{Z}_p^{n \times N}$

Putting it All Together

- Invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}\bar{G}$ for "short" $\bar{e}$
  - Secret key of the form $\bar{v} \in \mathbb{Z}_p^n$
  - Ciphertexts of the form $\bar{C} \in \mathbb{Z}_p^{n \times N}$
- To evaluate a NAND circuit $\mathrm{f} : \{0,1\}^\lambda \to \{0,1\}$ on ciphertexts $(\bar{C}_1, \ldots, \bar{C}_\lambda)$:
  1. Consider each gate G in f in topological order
  2. Let $\bar{C}_i$ and $\bar{C}_j$ denote ciphertexts corresponding to its inputs
  3. Output $\bar{C}_k := \bar{G} - \bar{C}_1 \mathrm{G}^{-1}(\bar{C}_2)$ as its output ciphertext

$\bar{G} - \bar{C}_1 \bar{G}^{-1}(C_2) = C_4$

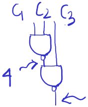


$C_5 = \bar{G} - \bar{C}_4 \bar{G}^{-1}(C_3)$

Putting it All Together

- Invariant: $n \times N$ matrix $\bar{C}$ encrypts a bit μ under secret $\bar{v}$ if $\bar{v}\bar{C} + \bar{e} = \mu\bar{v}\bar{G}$ for "short" $\bar{e}$
  - Secret key of the form $\bar{v} \in \mathbb{Z}_p^n$
  - Ciphertexts of the form $\bar{C} \in \mathbb{Z}_p^{n \times N}$
- To evaluate a NAND circuit $\mathrm{f} : \{0,1\}^\lambda \to \{0,1\}$ on ciphertexts $(\bar{C}_1, \ldots, \bar{C}_\lambda)$:
  1. Consider each gate G in f in topological order
  2. Let $\bar{C}_i$ and $\bar{C}_j$ denote ciphertexts corresponding to its inputs
  3. Output $\bar{C}_k := \bar{G} - \bar{C}_1 \mathrm{G}^{-1}(\bar{C}_2)$ as its output ciphertext

$\bar{G} - \bar{C}_1 \bar{G}^{-1}(C_2) = C_4$

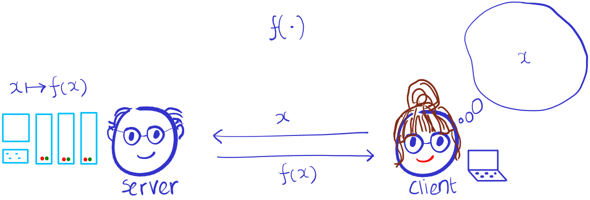


$C_5 = \bar{G} - \bar{C}_4 \bar{G}^{-1}(C_3)$

- If the depth is d then the noise in ciphertext of output wire is $B(N+1)^d$

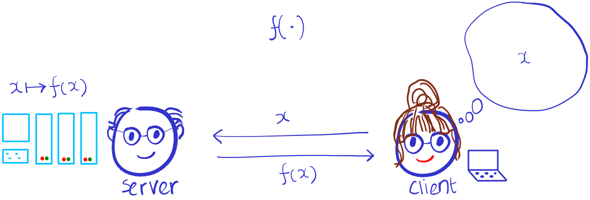
  $\Rightarrow$ modulus $q \gg B(N+1)^d$

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting



To Recap Today's Lecture

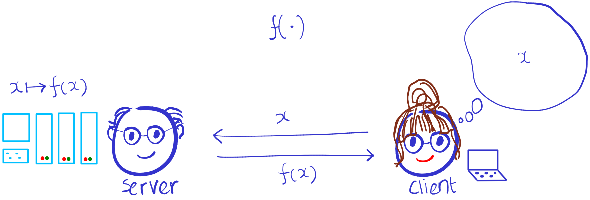
- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting



- Key tool: Fully homomorphic encryption (FHE)
  - FHE → private outsourcing of computation
  - Possible: FHE → 2PC of arbitrary functions!

To Recap Today's Lecture

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: *private* outsourcing in the client-server setting



- Key tool: Fully homomorphic encryption (FHE)
  - FHE → private outsourcing of computation
  - Possible: FHE → 2PC of arbitrary functions!
- GSW FHE from LWE assumption
  - Key idea: approximate eigenvectors
  - Similar idea used in approximate key exchange from LWE

Next Lecture

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: private outsourcing in the client-server setting
  - Task 7.b: *verifiable* outsourcing in the client-server setting

Next Lecture

- Task 7: secure outsourcing in the client-server setting
    - Task 7.a: private outsourcing in the client-server setting
    - Task 7.b: *verifiable* outsourcing in the client-server setting

- Key tool: succinct non-interactive argument (SNARG)

Next Lecture

- Task 7: secure outsourcing in the client-server setting
  - Task 7.a: private outsourcing in the client-server setting
  - Task 7.b: *verifiable* outsourcing in the client-server setting

- Key tool: succinct non-interactive argument (SNARG)
- SNARG for repeated squaring problem in RSA group
  - Pietrzak's interactive protocol
  - SNARG via Fiat-Shamir transform

References

1. Most of the lecture is based on Shai Halevi's survey [Hal17], which is a very nice resource on homomorphic encryption.
2. The partially homomorphic schemes we discussed are from [ElG84, GM82].
3. FHE was introduced in [RAD78], but the first candidate construction was given by Gentry only in [Gen09].
4. The GSW FHE was proposed in [GSW13]. The presentation here is taken from Halevi's survey [Hal17].

Taher ElGamal.
A public key cryptosystem and a signature scheme based on discrete logarithms.
In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

Craig Gentry.
Fully homomorphic encryption using ideal lattices.
In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

Shafi Goldwasser and Silvio Micali.
Probabilistic encryption and how to play mental poker keeping secret all partial information.
In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

Craig Gentry, Amit Sahai, and Brent Waters.
Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based.
In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

Shai Halevi.

Homomorphic encryption.
In *Tutorials on the Foundations of Cryptography*, pages 219–276. Springer International Publishing, 2017.

Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos.
On data banks and privacy homomorphisms.
In *Foundations of Secure Computation*, pages 165–179. 1978.