# On the Learning Parity with Noise Problem

**Luca Melis**

**Università degli Studi di Firenze**

**Århus Universitet**

**22 April 2013**

**Advisors:**
Prof. Alessandro Piva
Prof. Fabrizio Argenti

**Co-advisors:**
Dr. Claudio Orlandi
Prof. Ivan Damgård

# Scenario

## Cryptography schemes

- address the security of communication across an insecure medium
- are usually based only on complexity assumptions (standard model)

## Near Future:

- **Problem:** What if someone constructs large quantum computers?
- Cryptography world may fall apart:
  1. cryptographic assumptions broken by efficient quantum algorithms e.g. *factoring and discrete-logarithm broken by Shor's algorithm*
  2. proofs of security (or *reduction*) become unuseful

# Scenario

## Cryptography schemes

- address the security of communication across an insecure medium
- are usually based only on complexity assumptions (standard model)

## Near Future:

- Problem: What if someone constructs large quantum computers?
- Cryptography world may fall apart:
  1. cryptographic assumptions broken by efficient quantum algorithms e.g. *factoring and discrete-logarithm broken by Shor's algorithm*
  2. proofs of security (or *reduction*) become unuseful

# Scenario

## Cryptography schemes

- address the security of communication across an insecure medium
- are usually based only on complexity assumptions (standard model)

## Near Future:

- Problem: What if someone constructs large quantum computers?
- Cryptography world may fall apart:
  1. cryptographic assumptions broken by efficient quantum algorithms
     e.g. ***factoring*** *and* ***discrete-logarithm*** *broken by Shor's algorithm*
  2. proofs of security (or *reduction*) become unuseful

# Scenario

## Cryptography schemes

- address the security of communication across an insecure medium
- are usually based only on complexity assumptions (standard model)

## Near Future:

- Problem: What if someone constructs large quantum computers?
- Cryptography world may fall apart:
  1. cryptographic assumptions broken by efficient quantum algorithms e.g. ***factoring*** *and* ***discrete-logarithm*** *broken by Shor's algorithm*
  2. proofs of security (or *reduction*) become unuseful

# Scenario

## Cryptography schemes

- address the security of communication across an insecure medium
- are usually based only on complexity assumptions (standard model)

## Near Future:

- Problem: What if someone constructs large quantum computers?
- Cryptography world may fall apart:
  1. cryptographic assumptions broken by efficient quantum algorithms e.g. ***factoring*** *and* ***discrete-logarithm*** *broken by Shor's algorithm*
  2. proofs of security (or *reduction*) become unuseful

# Post-Quantum cryptography

Schemes that are believed to resist classical & quantum computers

- **Code-based cryptography**
- **Lattice-based cryptography**


**Our contribution**

- We investigate about the Learning Parity with Noise (LPN) problem
- We propose a Threshold Public-Key Encryption scheme based on LPN
- We propose a Commitment protocol based on LPN

# Post-Quantum cryptography

Schemes that are believed to resist classical & quantum computers

- **Code-based cryptography**
- **Lattice-based cryptography**


## Our contribution

- We investigate about the Learning Parity with Noise (LPN) problem
- We propose a Threshold Public-Key Encryption scheme based on LPN
- We propose a Commitment protocol based on LPN

# Post-Quantum cryptography

Schemes that are believed to resist classical & quantum computers

- **Code-based cryptography**
- **Lattice-based cryptography**


## Our contribution

- We investigate about the Learning Parity with Noise (LPN) problem
- We propose a Threshold Public-Key Encryption scheme based on LPN
- We propose a Commitment protocol based on LPN

# Post-Quantum cryptography

Schemes that are believed to resist classical & quantum computers

- **Code-based cryptography**
- **Lattice-based cryptography**


## Our contribution

- We investigate about the Learning Parity with Noise (LPN) problem
- We propose a Threshold Public-Key Encryption scheme based on LPN
- We propose a Commitment protocol based on LPN

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

$$\boldsymbol{a_1} \xleftarrow{R} \mathbb{Z}_2^\ell \quad , \quad b_1 = < \boldsymbol{a_1} , \boldsymbol{s} > \oplus e_1$$

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: <u>find</u> $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given “noisy random inner products”

$$\boldsymbol{a_1} \xleftarrow{R} \mathbb{Z}_2^\ell \quad , \quad b_1 =< \boldsymbol{a_1} \,,\, \boldsymbol{s} > \oplus e_1$$

$$\vdots$$

$$\boldsymbol{a_q} \xleftarrow{R} \mathbb{Z}_2^\ell \quad , \quad b_q =< \boldsymbol{a_q} \,,\, \boldsymbol{s} > \oplus e_q$$

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a_1} \\ \vdots \\ \boldsymbol{a_q} \end{pmatrix}, \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} \oplus \boldsymbol{e}$$

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

- **Decision**: distinguish $(\boldsymbol{A}, \boldsymbol{b})$ from uniform $(\boldsymbol{A}, \boldsymbol{b})$
- *decisional* and *search* LPN are *"polinomially equivalent"*

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a_1} \\ \vdots \\ \boldsymbol{a_q} \end{pmatrix}, \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} \oplus \boldsymbol{e}$$

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

- **Decision**: distinguish $(\boldsymbol{A}, \boldsymbol{b})$ from uniform $(\boldsymbol{A}, \boldsymbol{b})$
- *decisional* and *search* LPN are *"polinomially equivalent"*

# Learning Parity with Noise Problem LPN

- Dimension $\ell$ (security parameter), $q \gg \ell$, $\tau \in \left(0, \frac{1}{2}\right)$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a_1} \\ \vdots \\ \boldsymbol{a_q} \end{pmatrix}, \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} \oplus \boldsymbol{e}$$

Errors $e_i \leftarrow \mathrm{Ber}_\tau$, i.e. $\Pr(e_i = 1) = \tau$

- **Decision**: distinguish $(\boldsymbol{A}, \boldsymbol{b})$ from uniform $(\boldsymbol{A}, \boldsymbol{b})$
- *decisional* and *search* LPN are *"polinomially equivalent"*

# Learning Parity with Noise Problem LPN

## Hardness of LPN

The best known attacks against search LPN problem takes

- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples $q$
- $2^{\Theta(\ell/\log\log \ell)}$ having $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where $\ell$ is the security parameter

## Interesting features

- Efficiency $\rightarrow$ suitable for limited computing power devices(e.g. RFID)
- Quantum algorithms resistance

# Learning Parity with Noise Problem LPN

## Hardness of LPN

The best known attacks against search LPN problem takes

- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples $q$
- $2^{\Theta(\ell/\log\log \ell)}$ having $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where $\ell$ is the security parameter

## Interesting features

- **Efficiency** $\Rightarrow$ suitable for limited computing power devices (e.g. RFID).
- **Quantum algorithms resistance**

# Learning Parity with Noise Problem LPN

## Hardness of LPN

The best known attacks against search LPN problem takes

- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples $q$
- $2^{\Theta(\ell/\log\log \ell)}$ having $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where $\ell$ is the security parameter

## Interesting features

- Efficiency $\Rightarrow$ suitable for limited computing power devices (e.g. RFID)
- Quantum algorithms resistance

# Learning Parity with Noise Problem LPN

## Hardness of LPN

The best known attacks against search LPN problem takes

- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples $q$
- $2^{\Theta(\ell/\log\log \ell)}$ having $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where $\ell$ is the security parameter

## Interesting features

- **Efficiency** $\Rightarrow$ suitable for limited computing power devices (e.g. RFID).
- **Quantum algorithms resistance**

# Learning Parity with Noise Problem LPN

## Hardness of LPN

The best known attacks against search LPN problem takes

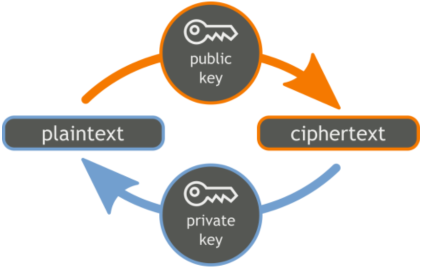
- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples $q$
- $2^{\Theta(\ell/\log\log \ell)}$ having $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where $\ell$ is the security parameter

## Interesting features

- **Efficiency** $\Rightarrow$ suitable for limited computing power devices (e.g. RFID).
- **Quantum algorithms resistance**

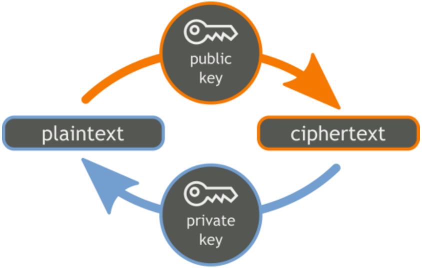
# Threshold Public-Key Encryption schemes




## Public-key cryptography

- The ability of decrypting or signing is restricted to the owner of the secret key.
- ⇒ only one person has all the power

# Threshold Public-Key Encryption schemes




## Public-key cryptography

- The ability of decrypting or signing is restricted to the owner of the secret key.
- $\Rightarrow$ only one person has all the power

# Threshold Public-Key Encryption schemes

## Solution: Threshold PKE

- The secret key is split into shares and each share is given to a group of users.
- Users can decrypt or sign only if enough, a *threshold*, cooperate

## Our contribution

A Threshold Public-Key Encryption scheme which is:

- based on LPN
- secure in the *Semi-honest* model

# Threshold Public-Key Encryption schemes

## Solution: Threshold PKE

- The secret key is split into shares and each share is given to a group of users.
- Users can decrypt or sign only if enough, a *threshold*, cooperate

## Our contribution

A Threshold Public-Key Encryption scheme which is:

- based on LPN
- secure in the *Semi-honest model*

# Threshold Public-Key Encryption schemes

## Solution: Threshold PKE

- The secret key is split into shares and each share is given to a group of users.
- Users can decrypt or sign only if enough, a *threshold*, cooperate

## **Our contribution**

A Threshold Public-Key Encryption scheme which is:

- based on LPN
- secure in the *Semi-honest* model

# Threshold Public-Key Encryption schemes

## Solution: Threshold PKE

- The secret key is split into shares and each share is given to a group of users.
- Users can decrypt or sign only if enough, a *threshold*, cooperate

## **Our contribution**

A Threshold Public-Key Encryption scheme which is:

- based on LPN
- secure in the *Semi-honest* model

# Threshold Public-Key Encryption schemes

## Solution: Threshold PKE

- The secret key is split into shares and each share is given to a group of users.
- Users can decrypt or sign only if enough, a *threshold*, cooperate

## **Our contribution**

A Threshold Public-Key Encryption scheme which is:

- based on LPN
- secure in the *Semi-honest* model

# Alekhnovich PKE scheme

## Key Generation

The receiver R chooses

- a secret key $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_2^{\ell}$
- $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q\times\ell}$ and the error $\boldsymbol{e} \leftarrow \mathrm{Ber}_\tau^q$, where $\tau \in \Theta(\frac{1}{\sqrt{\ell}})$ and computes the $pk$ as $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} \oplus \boldsymbol{e})$

**Encryption** of a message bit $m \in \mathbb{Z}_2$

| Sender S | | Receiver R |
|---|---|---|
| choose a vector $\boldsymbol{f} \leftarrow \mathrm{Ber}_\tau^q$ | | |
| compute $\boldsymbol{u} = \boldsymbol{f} \cdot \boldsymbol{A}$ | | |
| $c = \langle \boldsymbol{f}, \boldsymbol{b} \rangle \oplus m$ | $\xrightarrow{(\boldsymbol{u}, c)}$ | |

# Alekhnovich PKE scheme

## Key Generation

The receiver R chooses

- a secret key $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_2^{\ell}$
- $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q\times\ell}$ and the error $\boldsymbol{e} \leftarrow \mathrm{Ber}_\tau^q$, where $\tau \in \Theta(\frac{1}{\sqrt{\ell}})$ and computes the $pk$ as $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} \oplus \boldsymbol{e})$

**Encryption** of a message bit $m \in \mathbb{Z}_2$

| Sender S | | Receiver R |
|---|---|---|
| choose a vector $\boldsymbol{f} \leftarrow \mathrm{Ber}_\tau^q$ | | |
| compute $\boldsymbol{u} = \boldsymbol{f} \cdot \boldsymbol{A}$ | | |
| $c = \langle \boldsymbol{f}, \boldsymbol{b} \rangle \oplus m$ | $\xrightarrow{(\boldsymbol{u}, c)}$ | |

# Alekhnovich PKE scheme

## Key Generation

The receiver R chooses

- a secret key $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_2^\ell$
- $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q\times\ell}$ and the error $\boldsymbol{e} \leftarrow \mathrm{Ber}_\tau^q$, where $\tau \in \Theta(\frac{1}{\sqrt{\ell}})$ and computes the *pk* as $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} \oplus \boldsymbol{e})$

**Encryption** of a message bit $m \in \mathbb{Z}_2$

| Sender S | | Receiver R |
|---|---|---|
| choose a vector $\boldsymbol{f} \leftarrow \mathrm{Ber}_\tau^q$ | | |
| compute $\boldsymbol{u} = \boldsymbol{f} \cdot \boldsymbol{A}$ | | |
| $c = \langle \boldsymbol{f}, \boldsymbol{b}\rangle \oplus m$ | $\xrightarrow{(\boldsymbol{u}, c)}$ | |

## Decryption

The receiver R computes $d = c \oplus \langle \boldsymbol{s}, \boldsymbol{u}\rangle$

# Alekhnovich PKE scheme

## Key Generation

The receiver $\mathtt{R}$ chooses

- a secret key $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_2^{\ell}$
- $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q\times\ell}$ and the error $\boldsymbol{e} \leftarrow \mathrm{Ber}_{\tau}^{q}$, where $\tau \in \Theta(\frac{1}{\sqrt{\ell}})$ and computes the $pk$ as $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} \oplus \boldsymbol{e})$

**Encryption** of a message bit $m \in \mathbb{Z}_2$

| Sender $\underline{\mathtt{S}}$ | | Receiver $\underline{\mathtt{R}}$ |
|---|---|---|
| choose a vector $\boldsymbol{f} \leftarrow \mathrm{Ber}_{\tau}^{q}$ | | |
| compute $\boldsymbol{u} = \boldsymbol{f} \cdot \boldsymbol{A}$ | | |
| $c = \langle \boldsymbol{f}, \boldsymbol{b} \rangle \oplus m$ | $\xrightarrow{(\boldsymbol{u}, c)}$ | |

## Decryption

The receiver $\mathtt{R}$ computes $d = c \oplus \langle \boldsymbol{s}, \boldsymbol{u} \rangle = \cdots = \langle \boldsymbol{f}, \boldsymbol{e} \rangle \oplus m$

# ThPKE: Protocol phases

- **Key Generation**
- **Key Assembly**
- **Encryption**
- **Partial Decryption**
- **Finish Decryption**

# ThPKE: Protocol phases

- **Key Generation**
- Key Assembly
- Encryption
- Partial Decryption
- Finish Decryption

# ThPKE: Protocol phases

- **Key Generation**
- **Key Assembly**
- **Encryption**
- **Partial Decryption**
- **Finish Decryption**


## Key Generation

- All the receivers share a matrix $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q \times \ell}$
- Each receiver $\mathtt{R_i}$ indipendently choose a secret key $\boldsymbol{s}_i \xleftarrow{R} \mathbb{Z}_2^{\ell}$ and an error $\boldsymbol{e}_i \leftarrow \mathrm{Ber}_{\tau}^{q}$
- the public key for $\mathtt{R_i}$ is the pair $(\boldsymbol{A}, \boldsymbol{b}_i = \boldsymbol{A}\boldsymbol{s}_i \oplus \boldsymbol{e}_i)$

# ThPKE: Protocol phases

- **Key Generation**
- Key Assembly
- Encryption
- Partial Decryption
- Finish Decryption


## Key Generation

- All the receivers share a matrix $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q \times \ell}$
- Each receiver $\mathtt{R_i}$ indipendently choose a secret key $\boldsymbol{s_i} \xleftarrow{R} \mathbb{Z}_2^{\ell}$ and an error $\boldsymbol{e_i} \leftarrow \mathrm{Ber}_\tau^q$
- the public key for $\mathtt{R_i}$ is the pair $(\boldsymbol{A}, \boldsymbol{b}_i = \boldsymbol{A}\boldsymbol{s}_i \oplus \boldsymbol{e}_i)$

# ThPKE: Protocol phases

- **Key Generation**
- Key Assembly
- Encryption
- Partial Decryption
- Finish Decryption


## Key Generation

- All the receivers share a matrix $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q \times \ell}$
- Each receiver $\mathtt{R_i}$ indipendently choose a secret key $\boldsymbol{s_i} \xleftarrow{R} \mathbb{Z}_2^{\ell}$ and an error $\boldsymbol{e_i} \leftarrow \mathrm{Ber}_\tau^q$
- the public key for $\mathtt{R_i}$ is the pair $(\boldsymbol{A}, \boldsymbol{b_i} = \boldsymbol{A}\boldsymbol{s_i} \oplus \boldsymbol{e_i})$

# ThPKE: Protocol phases

- Key Generation
- **Key Assembly**
- Encryption
- Partial Decryption
- Finish Decryption

# ThPKE: Protocol phases

- Key Generation
- **Key Assembly**
- Encryption
- Partial Decryption
- Finish Decryption


## Key Assembly

The combined public key is the pair $(\boldsymbol{A}, \boldsymbol{b})$, where

$$\boldsymbol{b} = \bigoplus_{i \in I} \boldsymbol{b}_i$$

and $I$ is the users subset

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- **Encryption**
- Partial Decryption
- Finish Decryption

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- **Encryption**
- Partial Decryption
- Finish Decryption


Sender $\underline{\mathtt{S}}$ Receivers $\underline{\mathtt{R_i}, \mathtt{R_j}}$

$$(\boldsymbol{C_1}, \boldsymbol{c_2}) \leftarrow \mathtt{ThLPN.Enc}(m, \boldsymbol{b})$$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- **Encryption**
- Partial Decryption
- Finish Decryption


Sender $\underline{\mathtt{S}}$

Receivers $\underline{\mathtt{R_i}, \mathtt{R_j}}$

$$(\boldsymbol{C_1}, \boldsymbol{c_2}) \leftarrow \mathtt{ThLPN.Enc}(m, \boldsymbol{b})$$

## Encryption function (Alekhnovich scheme)

$$\boldsymbol{C_1} = \boldsymbol{F} \cdot \boldsymbol{A},\ \boldsymbol{c_2} = \boldsymbol{F} \cdot \boldsymbol{b} \oplus \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{q} \cdot m \qquad \text{where } \boldsymbol{F} := \begin{bmatrix} \boldsymbol{f_1} \\ \vdots \\ \boldsymbol{f_q} \end{bmatrix}, \boldsymbol{f_i} \leftarrow \mathrm{Ber}_\tau^q$$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- **Encryption**
- Partial Decryption
- Finish Decryption


Sender S | Receivers $\mathtt{R_i}, \mathtt{R_j}$

$$(\boldsymbol{C_1}, \boldsymbol{c_2}) \leftarrow \texttt{ThLPN.Enc}(m, \boldsymbol{b}) \xrightarrow{(\boldsymbol{C_1}, \boldsymbol{c_2})}$$

## Encryption function (Alekhnovich scheme)

$$\boldsymbol{C_1} = \boldsymbol{F} \cdot \boldsymbol{A}, \ \boldsymbol{c_2} = \boldsymbol{F} \cdot \boldsymbol{b} \oplus \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{q} \cdot m \qquad \text{where } \boldsymbol{F} := \begin{bmatrix} \boldsymbol{f_1} \\ \vdots \\ \boldsymbol{f_q} \end{bmatrix}, \ \boldsymbol{f_i} \leftarrow \mathrm{Ber}_\tau^q$$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption


Receiver $\underline{\mathtt{R_i}}$ Receiver $\underline{\mathtt{R_j}}$

$$\boldsymbol{d_i} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i})$$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption


Receiver $\mathtt{R_i}$ Receiver $\mathtt{R_j}$

$$\boldsymbol{d_i} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i})$$

**Partial decryption function (Alekhnovich scheme)**

$$\boldsymbol{d_i} = \boldsymbol{C_1} \cdot \boldsymbol{s_i} \oplus \boldsymbol{\nu_i}$$

where $\boldsymbol{\nu_i} \leftarrow \mathrm{Ber}_\sigma^q$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption


Receiver $\mathtt{R_i}$ | Receiver $\mathtt{R_j}$

$\boldsymbol{d_i} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i}) \xrightarrow{\boldsymbol{d_i}}$

**Partial decryption function (Alekhnovich scheme)**

$$\boldsymbol{d_i} = \boldsymbol{C_1} \cdot \boldsymbol{s_i} \oplus \boldsymbol{\nu_i}$$

where $\boldsymbol{\nu_i} \leftarrow \mathrm{Ber}_\sigma^q$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption


Receiver $\mathtt{R_i}$ | Receiver $\mathtt{R_j}$

$\boldsymbol{d_i} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i}) \xrightarrow{\boldsymbol{d_i}}$

$\boldsymbol{d_j} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_j})$

**Partial decryption function (Alekhnovich scheme)**

$$\boldsymbol{d_i} = \boldsymbol{C_1} \cdot \boldsymbol{s_i} \oplus \boldsymbol{\nu_i}$$

where $\boldsymbol{\nu_i} \leftarrow \mathrm{Ber}_{\sigma}^{q}$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption


Receiver $\mathtt{R_i}$ | Receiver $\mathtt{R_j}$

$\boldsymbol{d_i} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i})$ $\xrightarrow{\boldsymbol{d_i}}$

$\xleftarrow{\boldsymbol{d_j}}$ $\boldsymbol{d_j} \leftarrow \mathtt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_j})$

**Partial decryption function (Alekhnovich scheme)**

$$\boldsymbol{d_i} = \boldsymbol{C_1} \cdot \boldsymbol{s_i} \oplus \boldsymbol{\nu_i}$$

where $\boldsymbol{\nu_i} \leftarrow \mathrm{Ber}_\sigma^q$

# ThPKE: Protocol phases

- **Key Generation**
- **Key Assembly**
- **Encryption**
- **Partial Decryption**
- **Finish Decryption**

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- Partial Decryption
- **Finish Decryption**


## Finish decryption

- Each receiver indipendently computes the vector

$$\boldsymbol{d} = \boldsymbol{c_2} \bigoplus_{i \in I} (\boldsymbol{d_i}) = \cdots = \boldsymbol{F} \cdot \boldsymbol{e} \oplus \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{q} \cdot m \bigoplus_{i \in I} (\boldsymbol{\nu_i}) \, .$$

- the bit in $\boldsymbol{d}$ that is in majority is separately chosen by each receiver as $m$

# ThPKE: Protocol phases

- Key Generation
- Key Assembly
- Encryption
- Partial Decryption
- **Finish Decryption**


## Finish decryption

- Each receiver indipendently computes the vector

$$\boldsymbol{d} = \boldsymbol{c_2} \bigoplus_{i \in I} (\boldsymbol{d_i}) = \cdots = \boldsymbol{F} \cdot \boldsymbol{e} \oplus \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{q} \cdot m \bigoplus_{i \in I} (\boldsymbol{\nu_i}) \, .$$

- the bit in $\boldsymbol{d}$ that is in majority is separately chosen by each receiver as $m$

# Protocol Security Analysis

## Semi-honest model

A semi-honest party:

1. Follows the protocol properly
2. Keeps a record of all its intermediate computations

## Security

- **Encryption:** from the Alekhnovich's scheme security
- **Decryption:** from the LPN hardness assumption, as each $\mathrm{P}_i$ is generating LPN samples

$$d_i = C_1 \cdot s_i \oplus \nu_i$$

# Protocol Security Analysis

## Semi-honest model

A semi-honest party:

1. Follows the protocol properly
2. Keeps a record of all its intermediate computations

## Security

- **Encryption:** from the Alekhnovich's scheme security
- **Decryption:** from the LPN hardness assumption, as each $\mathtt{R}_i$ is generating LPN samples

$$\boldsymbol{d_i = C_1 \cdot s_i \oplus \nu_i}$$

# Protocol Security Analysis

## Semi-honest model

A semi-honest party:

1. Follows the protocol properly
2. Keeps a record of all its intermediate computations

## Security

- **Encryption:** from the Alekhnovich's scheme security
- **Decryption:** from the LPN hardness assumption, as each $\mathtt{R}_i$ is generating LPN samples

$$\boldsymbol{d_i} = \boldsymbol{C_1} \cdot \boldsymbol{s_i} \oplus \boldsymbol{\nu_i}$$

# Protocol Security Analysis

## Relaxed Semi-honest model

- Semi-honest model not so realistic (*replay attacks* may occur)
- Problem: if the same message is encrypted multiple times then it is possible to recover information about the secret key from the ciphertexts

## Proposed solutions

1. implement the receivers as stateful machines (not good in resource-constrained devices)
2. make use of (i.e. deterministic algorithms that simulate truly random functions, given a "seed")

# Protocol Security Analysis

## Relaxed Semi-honest model

- Semi-honest model not so realistic (*replay attacks* may occur)
- Problem: if the same message is encrypted multiple times then it is possible to recover information about the secret key from the ciphertexts

## Proposed solutions

1. implement the receivers as stateful machines (not good in resource-constrained devices)
2. make use of pseudo-random functions (i.e. deterministic algorithms that simulate truly random functions, given a "seed")

# Protocol Security Analysis

## Relaxed Semi-honest model

- Semi-honest model not so realistic (*replay attacks* may occur)
- Problem: if the same message is encrypted multiple times then it is possible to recover information about the secret key from the ciphertexts

## Proposed solutions

1. implement the receivers as stateful machines (not good in resource-constrained devices)
2. make use of pseudorandom functions (i.e. deterministic algorithms that simulate truly random functions, given a "seed")

# Protocol Security Analysis

## Relaxed Semi-honest model

- Semi-honest model not so realistic (*replay attacks* may occur)
- Problem: if the same message is encrypted multiple times then it is possible to recover information about the secret key from the ciphertexts

## Proposed solutions

1. implement the receivers as stateful machines (not good in resource-constrained devices)
2. make use of pseudorandom functions (i.e. deterministic algorithms that simulate truly random functions, given a "seed")

# Conclusions and Open Problems

## Our contribution

- study the security of our Threshold Public-Key Encryption scheme in the *malicious model*

## LPN open problems

- relation between standard LPN and some variants
- Does LPN with noise rate $\tau$ imply anything about LPN with $\tau' < \tau$? Is there a threshold?
- how to get some basic primitives from standard LPN?

# Conclusions and Open Problems

## Our contribution

- study the security of our Threshold Public-Key Encryption scheme in the *malicious model*

## LPN open problems

- relation between standard LPN and some variants
- Does LPN with noise rate $\tau$ imply anything about LPN with $\tau' < \tau$? Is there a threshold?
- how to get some basic primitives from standard LPN?

# Conclusions and Open Problems

## Our contribution

- study the security of our Threshold Public-Key Encryption scheme in the *malicious model*

## LPN open problems

- relation between standard LPN and some variants
- Does LPN with noise rate $\tau$ imply anything about LPN with $\tau' < \tau$? Is there a threshold?
- how to get some basic primitives from standard LPN?

# Conclusions and Open Problems

## Our contribution

- study the security of our Threshold Public-Key Encryption scheme in the *malicious model*

## LPN open problems

- relation between standard LPN and some variants
- Does LPN with noise rate $\tau$ imply anything about LPN with $\tau' < \tau$? Is there a threshold?
- how to get some basic primitives from standard LPN?

# Commitment Protocols

**Scenario:**

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

**Commitment protocol**

- Alice commits the message and Bob does not learn any information about it (hiding property)
- Alice chooses to open the commitment and reveal the message, but she cannot change the value committed ( property)

**Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al.*
- based on Exact-LPN problem (where [illegible])
- does not need a trusted third party

# Commitment Protocols

## Scenario:

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

## Commitment protocol

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

## Our contribution

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al.*
- based on Exact-LPN problem (where $wt(e) = \lceil r \cdot \ell \rceil$)
- does not need a trusted third party

# Commitment Protocols

**Scenario:**

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

**Commitment protocol**

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

**Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathrm{wt}(e) = \lceil \tau \cdot \ell \rceil$)
- does not need a trusted third party

# Commitment Protocols

**Scenario:**

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

**Commitment protocol**

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

**Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathrm{wt}(e) = \lfloor \tau \cdot \ell \rfloor$)
- does not need a trusted third party

# Commitment Protocols

## Scenario:

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

## Commitment protocol

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

## Our contribution

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathbf{wt}(\boldsymbol{e}) = \lfloor \tau \cdot \ell \rceil$)
- does not need a trusted third party

# Commitment Protocols

## Scenario:

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

## Commitment protocol

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

## **Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathbf{wt}(\boldsymbol{e}) = \lfloor \tau \cdot \ell \rceil$)
- does not need a trusted third party

# Commitment Protocols

## Scenario:

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

## Commitment protocol

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

## **Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathbf{wt}(\boldsymbol{e}) = \lfloor \tau \cdot \ell \rceil$)
- does not need a trusted third party

# Commitment Protocols

## Scenario:

Alice wants to keep a message secret from Bob for now but she intends to reveal it to Bob at some time in the future

## Commitment protocol

- Alice **commits** the message and Bob does not learn any information about it (hiding property)
- Alice chooses to **open** the commitment and reveal the message, but she cannot change the value committed (binding property)

## **Our contribution**

We presented a Commitment protocol

- based on the commitment protocol by *Jain et al*
- based on Exact-LPN problem (where $\mathbf{wt}(\boldsymbol{e}) = \lfloor \tau \cdot \ell \rceil$)
- does not need a trusted third party

# The commitment protocol by *Jain et al*

## Setup Phase

In order to commit a message $\boldsymbol{m} \in \mathbb{Z}_2^k$ where $k \in \Theta(\ell + v)$
We state $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}] \in \mathbb{Z}_2^{k \times (\ell+v)}$ as the common reference string (CRS).
Finally, we set $w = \lfloor \tau k \rceil$.

# The commitment protocol by *Jain et al*

## Setup Phase

In order to commit a message $\boldsymbol{m} \in \mathbb{Z}_2^k$ where $k \in \Theta(\ell + v)$
We state $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}] \in \mathbb{Z}_2^{k \times (\ell+v)}$ as the common reference string (CRS).
Finally, we set $w = \lfloor \tau k \rfloor$.

**Commitment phase**

Sender $\underline{\mathtt{S}}$ | Receiver $\underline{\mathtt{R}}$

chooses $\boldsymbol{r} \xleftarrow{R} \mathbb{Z}_2^\ell$, $\boldsymbol{e} \in \mathbb{Z}_2^k$ s.t. $\boldsymbol{wt}(\boldsymbol{e}) = w$
computes $\boldsymbol{c} = \boldsymbol{A}(\boldsymbol{r} \| \boldsymbol{m}) \oplus \boldsymbol{e}$

# The commitment protocol by *Jain et al*

## Setup Phase

In order to commit a message $\boldsymbol{m} \in \mathbb{Z}_2^k$ where $k \in \Theta(\ell + v)$
We state $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}] \in \mathbb{Z}_2^{k \times (\ell+v)}$ as the common reference string (CRS).
Finally, we set $w = \lfloor \tau k \rfloor$.

**Commitment phase**

Sender $\underline{\mathtt{S}}$ — Receiver $\underline{\mathtt{R}}$

chooses $\boldsymbol{r} \xleftarrow{R} \mathbb{Z}_2^\ell$, $\boldsymbol{e} \in \mathbb{Z}_2^k$ s.t. $\boldsymbol{wt}(\boldsymbol{e}) = w$
computes $\boldsymbol{c} = \boldsymbol{A}(\boldsymbol{r} \| \boldsymbol{m}) \oplus \boldsymbol{e}$ $\xrightarrow{\boldsymbol{c}}$

# The commitment protocol by *Jain et al*

## Setup Phase

In order to commit a message $\boldsymbol{m} \in \mathbb{Z}_2^k$ where $k \in \Theta(\ell + v)$
We state $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}] \in \mathbb{Z}_2^{k \times (\ell+v)}$ as the common reference string (CRS).
Finally, we set $w = \lfloor \tau k \rfloor$.

**Commitment phase**

Sender $\underline{\mathtt{S}}$ Receiver $\underline{\mathtt{R}}$

chooses $\boldsymbol{r} \xleftarrow{R} \mathbb{Z}_2^{\ell}$, $\boldsymbol{e} \in \mathbb{Z}_2^k$ s.t. $\boldsymbol{wt}(\boldsymbol{e}) = w$
computes $\boldsymbol{c} = \boldsymbol{A}(\boldsymbol{r} \| \boldsymbol{m}) \oplus \boldsymbol{e}$ $\xrightarrow{\quad \boldsymbol{c} \quad}$

**Opening phase**

define $\boldsymbol{d} = (\boldsymbol{m'}, \boldsymbol{r'})$ $\xrightarrow{\quad \boldsymbol{d} \quad}$

# The commitment protocol by *Jain et al*

## Setup Phase

In order to commit a message $\boldsymbol{m} \in \mathbb{Z}_2^k$ where $k \in \Theta(\ell + v)$
We state $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}] \in \mathbb{Z}_2^{k \times (\ell+v)}$ as the common reference string (CRS).
Finally, we set $w = \lfloor \tau k \rfloor$.

**Commitment phase**

Sender $\underline{\mathtt{S}}$ | Receiver $\underline{\mathtt{R}}$

chooses $\boldsymbol{r} \xleftarrow{R} \mathbb{Z}_2^\ell$, $\boldsymbol{e} \in \mathbb{Z}_2^k$ s.t. $\boldsymbol{wt}(\boldsymbol{e}) = w$
computes $\boldsymbol{c} = \boldsymbol{A}(\boldsymbol{r} \| \boldsymbol{m}) \oplus \boldsymbol{e}$ $\xrightarrow{\boldsymbol{c}}$

**Opening phase**

define $\boldsymbol{d} = (\boldsymbol{m'}, \boldsymbol{r'})$ $\xrightarrow{\boldsymbol{d}}$
computes $\boldsymbol{e'} = \boldsymbol{c} \oplus \boldsymbol{A}(\boldsymbol{r'} \| \boldsymbol{m'})$
$\xleftarrow{Yes, No}$ accepts iff $\boldsymbol{wt}(\boldsymbol{e'}) = w$

# Proposed commitment protocol

**Problem**

We need a trusted third party for the common matrix $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}]$

**Solution:**

The Commitment and Opening phases are the same as in the original scheme

# Proposed commitment protocol

**Problem**

We need a trusted third party for the common matrix $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}]$

**Solution:**

**Setup phase**

| Sender $\underline{\mathtt{S}}$ | | Receiver $\underline{\mathtt{R}}$ |
|---|---|---|
| chooses $\boldsymbol{A'} \xleftarrow{R} \mathbb{Z}_2^{k \times \ell}$ | $\xrightarrow{\boldsymbol{A'}}$ | |

The Commitment and Opening phases are the same as in the original scheme

# Proposed commitment protocol

**Problem**

We need a trusted third party for the common matrix $\boldsymbol{A} = [\boldsymbol{A}' \| \boldsymbol{A}'']$

**Solution:**

**Setup phase**

| Sender $\underline{\mathtt{S}}$ | | Receiver $\underline{\mathtt{R}}$ |
|---|---|---|
| chooses $\boldsymbol{A}' \xleftarrow{R} \mathbb{Z}_2^{k \times \ell}$ | $\xrightarrow{\boldsymbol{A}'}$ | |
| | $\xleftarrow{\boldsymbol{A}''}$ | chooses $\boldsymbol{A}'' \xleftarrow{R} \mathbb{Z}_2^{k \times v}$ |

The Commitment and Opening phases are the same as in the original scheme

# Proposed commitment protocol

**Problem**

We need a trusted third party for the common matrix $\boldsymbol{A} = [\boldsymbol{A'} \| \boldsymbol{A''}]$

**Solution:**

**Setup phase**

| Sender $\underline{\mathtt{S}}$ | | Receiver $\underline{\mathtt{R}}$ |
|---|---|---|
| chooses $\boldsymbol{A'} \xleftarrow{R} \mathbb{Z}_2^{k\times\ell}$ | $\xrightarrow{\quad \boldsymbol{A'} \quad}$ | |
| | $\xleftarrow{\quad \boldsymbol{A''} \quad}$ | chooses $\boldsymbol{A''} \xleftarrow{R} \mathbb{Z}_2^{k\times v}$ |

The Commitment and Opening phases are the same as in the original scheme

# The commitment protocol: a LPN-based variant

## LPN variant

- security directly based on the standard LPN problem
- **Commit phase:** we set $w' = 2 \cdot \lfloor \tau k \rceil$ and we choose $\boldsymbol{e}$ such that $\boldsymbol{wt}(\boldsymbol{e}) \leq w'$

## Choice of parameters

According to

**Levieil, Éric and Fouque, Pierre-Alain**
An Improved LPN Algorithm
*Springer Berlin Heidelberg*, 2006

we choose $\ell = 768$ and noise rate $\tau = \frac{1}{8} \Rightarrow 2^{90}$ bytes of memory to solve LPN

## Theorem

*Our commitment scheme is statistically binding and computationally hiding*

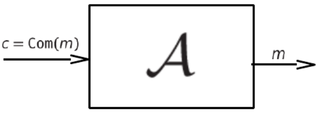
# Proof

## Statistically binding

even if `S` is computationally unbounded she cannot cheat with probability greater than $2^{-k}$

## Computationally hiding

- proof for reduction (single bit message)
- we assume that $\mathcal{A}$ is able to break the commitment scheme




- Let $\mathcal{B}$ an oracle

images/oracle

where $\boldsymbol{b} = \begin{cases} \text{random} & w.p.\ 1/2 \\ \boldsymbol{A's} \oplus \boldsymbol{e} & w.p.\ 1/2 \end{cases}$

# Proof

images/proof$_0$

# Proof

images/proof$_1$

# Proof

images/proof$_2$

# Proof

images/proof$_3$

# Proof

images/proof$_4$

**case 1)**

$\boldsymbol{b}$ is random $\Rightarrow \boldsymbol{c} = \boldsymbol{b} \oplus \boldsymbol{A''}m$ is a onetime-pad encryption

$\Rightarrow \mathcal{A}$ guesses w.p. $\frac{1}{2}$

# Proof


**case 2)**

$\boldsymbol{b}$ is a Exact-LPN sample $\Rightarrow$ $\boldsymbol{c}$ is a well formed commitment
$\Rightarrow \mathcal{A}$ guesses w.p. 1 (by hypothesis)

# Proof

images/proof$_3$

## case 1) and 2)

Let E = the reduction breaks the Exact-LPN problem,

# Proof

images/proof$_3$

## case 1) and 2)

Let E = the reduction breaks the Exact-LPN problem,

$$\Pr(E) = \Pr\left(E \middle|\ \boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) \cdot \Pr\left(\boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) + \Pr\left(E \middle|\ \boldsymbol{b} \text{ is random}\right) \cdot \Pr\left(\boldsymbol{b} \text{ is random}\right)$$

# Proof

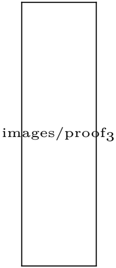
images/proof$_3$

## case 1) and 2)

Let E = the reduction breaks the Exact-LPN problem,

$$\begin{aligned}
\Pr(E) &= \Pr\left(E \middle|\, \boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) \cdot \Pr\left(\boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) + \Pr\left(E \middle|\, \boldsymbol{b} \text{ is random}\right) \cdot \Pr\left(\boldsymbol{b} \text{ is random}\right) \\
&= 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}
\end{aligned}$$

# Proof




## case 1) and 2)

Let E = the reduction breaks the Exact-LPN problem,

$$\begin{aligned}\Pr(E) &= \Pr\left(E \middle| \, \boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) \cdot \Pr\left(\boldsymbol{b} = \boldsymbol{A'}\boldsymbol{s} \oplus \boldsymbol{e}\right) + \Pr\left(E \middle| \, \boldsymbol{b} \text{ is random}\right) \cdot \Pr\left(\boldsymbol{b} \text{ is random}\right) \\ &= 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}\end{aligned}$$

Exact-LPN hardness $\Rightarrow$ Hiding commitment