

# On the Learning Parity with Noise Problem

Luca Melis

Università degli Studi di Firenze

Århus Universitet

19 Aprile 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
  - 2 proof of security becomes invalid

# Post-Quantum cryptography

Schemes that are believed to resist classical computers and quantum computers

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

Some Issues:

- **Efficiency** time and space
- **Confidence** cryptanalysts experience
- **Usability** infrastructure

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),  $q \gg \ell$ ,  $\tau \in (0, \frac{1}{2}]$
- **Search:** find  $s \in \mathbb{Z}_2^\ell$  given “noisy random inner products”

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

- **Decision:** distinguish ( $\mathbf{a}_i, \mathbf{b}_i$ ) from uniform ( $\mathbf{a}_i, \mathbf{b}_i$ )
- LPN becomes trivial with no error  $\tau = 0$  (Gaussian elimination)
- *decisional* and *search* LPN are “*polynomially equivalent*”

# Learning Parity with Noise Problem LPN

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

$$\mathbf{a}_1 \xleftarrow{R} \mathbb{Z}_2^\ell, \quad \mathbf{b}_1 = \langle \mathbf{a}_1, \mathbf{s} \rangle \oplus e_1$$

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

- **Decision:** distinguish  $(\mathbf{a}_i, \mathbf{b}_i)$  from uniform  $(\mathbf{a}_i, \mathbf{b}_i)$
- LPN becomes trivial with no error  $\tau = 0$  (Gaussian elimination)
- *decisional* and *search* LPN are “*polynomially equivalent*”

# Learning Parity with Noise Problem LPN

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

$$\mathbf{a}_1 \xleftarrow{R} \mathbb{Z}_2^\ell, \quad \mathbf{b}_1 = \langle \mathbf{a}_1, \mathbf{s} \rangle \oplus e_1$$

$$\mathbf{a}_2 \xleftarrow{R} \mathbb{Z}_2^\ell, \quad \mathbf{b}_2 = \langle \mathbf{a}_2, \mathbf{s} \rangle \oplus e_2$$

$$\vdots$$

$$\mathbf{a}_q \xleftarrow{R} \mathbb{Z}_2^\ell, \quad \mathbf{b}_q = \langle \mathbf{a}_q, \mathbf{s} \rangle \oplus e_q$$

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

- **Decision:** distinguish  $(\mathbf{a}_i, \mathbf{b}_i)$  from uniform  $(\mathbf{a}_i, \mathbf{b}_i)$
- LPN becomes trivial with no error  $\tau = 0$  (Gaussian elimination)
- *decisional* and *search* LPN are “*polynomially equivalent*”

# Learning Parity with Noise Problem LPN

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

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_q \end{pmatrix}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$$

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

- **Decision:** distinguish  $(\mathbf{a}_i, \mathbf{b}_i)$  from uniform  $(\mathbf{a}_i, \mathbf{b}_i)$
- LPN becomes trivial with no error  $\tau = 0$  (Gaussian elimination)
- *decisional and search LPN are “polynomially equivalent”*

# Learning Parity with Noise Problem LPN

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

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_q \end{pmatrix}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$$

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

- **Decision:** distinguish  $(\mathbf{a}_i, \mathbf{b}_i)$  from uniform  $(\mathbf{a}_i, \mathbf{b}_i)$
- LPN becomes trivial with no error  $\tau = 0$  (Gaussian elimination)
- *decisional* and *search* LPN are “*polynomially equivalent*”



# Learning Parity with Noise Problem LPN

## LPN variants

- Ring LPN
- Subspace LPN
- Exact LPN

## Hardness of LPN

Breaking the search LPN problem takes time

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

## Interesting features

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

# Threshold Public-Key Encryption schemes

## Scenario

- In public-key cryptography in general, the ability of decrypting or signing is restricted to the owner of the secret key.
- $\Rightarrow$  **only one person has all the power**

## Solution

- Threshold PKE shares trust among a group of users, such that *enough* of them, the *threshold*, is needed to sign or decrypt
- The secret key is split into shares and each share is given to a group of users.

## Our contribution

A Threshold Public-Key Encryption scheme which is:

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

# Alekhnovich Public Key Encryption scheme

## Key Generation

The sender **S** chooses

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

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

Sender S

Receiver R

choose a vector  $\mathbf{f} \leftarrow \text{Ber}_\tau^q$

compute  $\mathbf{u} = \mathbf{f} \cdot \mathbf{A}$

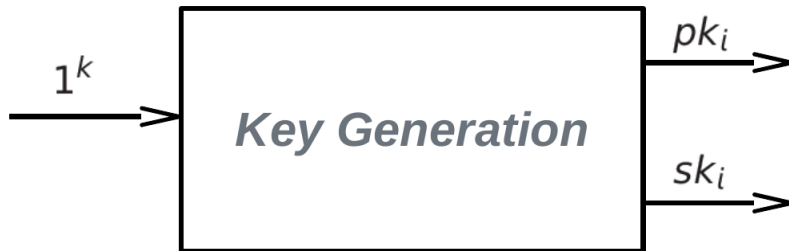
$$c = \langle \mathbf{f}, \mathbf{b} \rangle \oplus m \xrightarrow{(\mathbf{u}, c)}$$

## Decryption

The receiver **R** computes  $d = c \oplus \langle \mathbf{s}, \mathbf{u} \rangle = \dots = \langle \mathbf{f}, \mathbf{e} \rangle \oplus m$

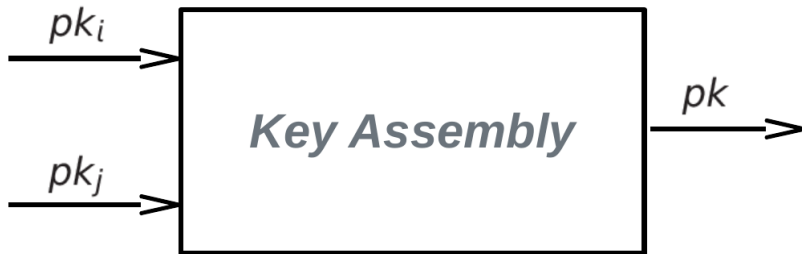
## Protocol phases:

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



## Protocol phases:

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



## Protocol phases:

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



## Protocol phases:

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



## Protocol phases:

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





## Key Generation

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

## Key Assembly

The combined public key is the pair  $(\mathbf{A}, \mathbf{b})$ , where  $\mathbf{b} = \bigoplus_{i \in I} \mathbf{b}_i$  ( $I$  is the users subset)

## Encryption Phase

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

$$(\mathbf{C}_1, \mathbf{c}_2) \leftarrow \text{ThLPN.Enc}(m, \mathbf{b})$$

Encryption function (Alekhnovich scheme)

$$\mathbf{C}_1 = \mathbf{F} \cdot \mathbf{A},$$

$$\mathbf{c}_2 = \mathbf{F} \cdot \mathbf{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

where  $\mathbf{F} := \begin{bmatrix} f_1 \\ \dots \\ f_q \end{bmatrix}$ ,  $f_i \leftarrow \text{Ber}_\tau^q$

## Encryption Phase

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

$$(\mathbf{C}_1, \mathbf{c}_2) \leftarrow \text{ThLPN.Enc}(m, \mathbf{b})$$

Encryption function (Alekhnovich scheme)

$$\mathbf{C}_1 = \mathbf{F} \cdot \mathbf{A},$$

$$\mathbf{c}_2 = \mathbf{F} \cdot \mathbf{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

where  $\mathbf{F} := \begin{bmatrix} \mathbf{f}_1 \\ \dots \\ \mathbf{f}_q \end{bmatrix}$ ,  $\mathbf{f}_i \leftarrow \text{Ber}_\tau^q$

## Encryption Phase

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

$$(\mathbf{C}_1, \mathbf{c}_2) \leftarrow \text{ThLPN.Enc}(m, \mathbf{b}) \xrightarrow{(\mathbf{C}_1, \mathbf{c}_2)}$$

Encryption function (Alekhnovich scheme)

$$\mathbf{C}_1 = \mathbf{F} \cdot \mathbf{A},$$

$$\mathbf{c}_2 = \mathbf{F} \cdot \mathbf{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

where  $\mathbf{F} := \begin{bmatrix} \mathbf{f}_1 \\ \dots \\ \mathbf{f}_q \end{bmatrix}$ ,  $\mathbf{f}_i \leftarrow \text{Ber}_\tau^q$

## Partial Decryption Phase

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

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

## Partial Decryption Phase

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

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

Partial decryption function (Alekhnovich scheme)

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

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

## Partial Decryption Phase

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

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

Partial decryption function (Alekhnovich scheme)

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

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

## Partial Decryption Phase

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

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

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

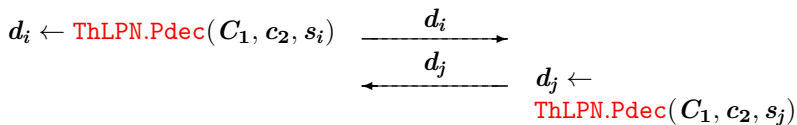
Partial decryption function (Alekhnovich scheme)

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

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



## Partial Decryption Phase

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

Partial decryption function (Alekhnovich scheme)

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

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

## Partial Decryption Phase

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

$$\begin{array}{ccc}
 d_i \leftarrow \text{ThLPN.Pdec}(C_1, c_2, s_i) & \xrightarrow{d_i} & \\
 & \xleftarrow{d_j} & d_j \leftarrow \text{ThLPN.Pdec}(C_1, c_2, s_j)
 \end{array}$$

## Finish decryption

- Each receiver independently computes the vector

$$d = c_2 \bigoplus_{i \in I} (d_i) = F \cdot e \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m \bigoplus_{i \in I} (\nu_i).$$

- the bit in the vector  $d$  that is in majority is separately chosen by each receiver as the plaintext  $m$

# Protocol Security Analysis

## Semi-honest model

We make the following two assumptions:

- 1 The semi-honest party will indeed toss a fair coin
- 2 The semi-honest party will send all messages as instructed by the protocol

## Security

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

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

## 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

## Possible 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”)

# Commitment Protocols

## Commitment protocol

- can be thought as the digital analogue of a sealed envelope
- **Commit:** the sender **S** commit to a message  $m$  and the receiver **R** does not learn any information about  $m$  (**hiding** property)
- **Open:** **S** can choose to open the commitment and reveal the content  $m$ , but no other value (**binding** property)

## Our contribution

We presented a commitment protocol

- based on the commitment protocol by Jain et al
- based on Exact-LPN problem (where  $\mathbf{wt}(e) = w$ )
- not in a *common reference string (CRS)* model

# The commitment protocol

## Setup Phase

In order to commit a message  $\mathbf{m} \in \mathbb{Z}_2^k$  where  $k \in \Theta(\ell + v)$

We let  $\mathbf{A}' \xleftarrow{R} \mathbb{Z}_2^{k \times \ell}$  and  $\mathbf{A}'' \xleftarrow{R} \mathbb{Z}_2^{k \times v}$ . We state  $\mathbf{A} = [\mathbf{A}' \parallel \mathbf{A}''] \in \mathbb{Z}_2^{k \times (\ell + v)}$  as **the common reference string (CRS)**. Finally, we set  $w = \lfloor \tau k \rfloor$ .

## Commitment phase

Sender  $\underline{\mathbf{S}}$

Receiver  $\underline{\mathbf{R}}$

chooses  $\mathbf{r} \xleftarrow{R} \mathbb{Z}_2^\ell$ ,  $\mathbf{e} \in \mathbb{Z}_2^k$  s.t.  $\text{wt}(\mathbf{e}) = w$

computes  $\mathbf{c} = \mathbf{A}(\mathbf{r} \parallel \mathbf{m}) \oplus \mathbf{e} \xrightarrow{\mathbf{c}}$

## Opening phase

computes  $\mathbf{d} = (\mathbf{m}', \mathbf{r}') \xrightarrow{\mathbf{d}}$

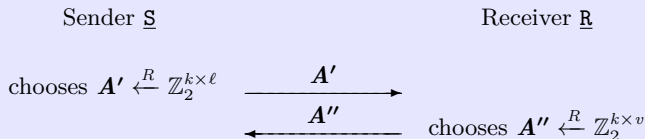
computes  $\mathbf{e}' = \mathbf{c} \oplus \mathbf{A}(\mathbf{r}' \parallel \mathbf{m}')$

$\xleftarrow{\text{Yes, No}}$

accepts iff  $\text{wt}(\mathbf{e}') = w$

**Problem**

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

**Solution:****Setup phase**

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

**Theorem**

*Our commitment scheme is statistically binding and computationally hiding*

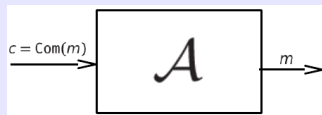
# Proof

## Statistically binding

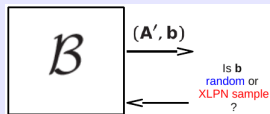
even if  $\mathcal{S}$  is computationally unbounded she cannot cheat with probability greater than  $2^{-k}$

## Computationally hiding

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

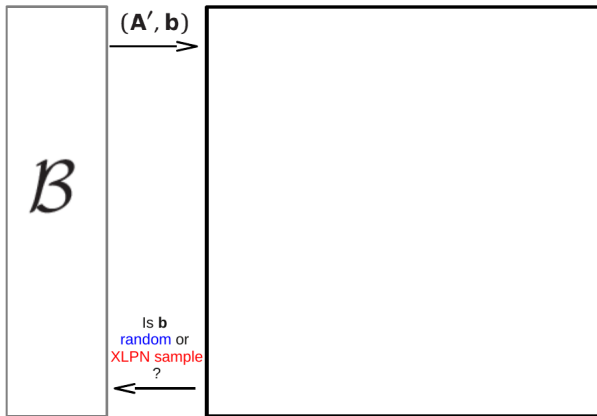


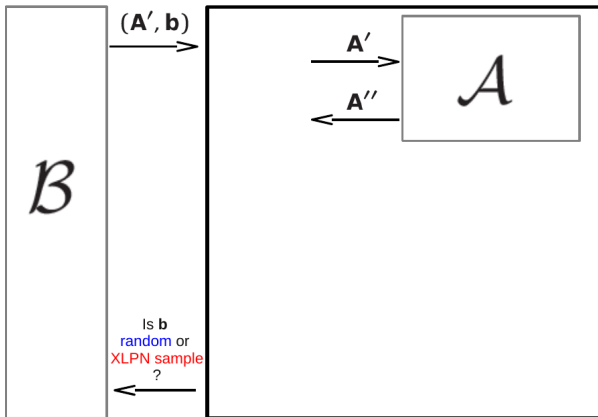
- Let  $\mathcal{B}$  an oracle

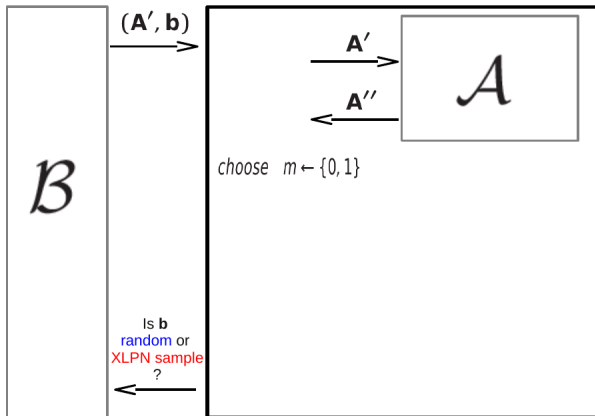


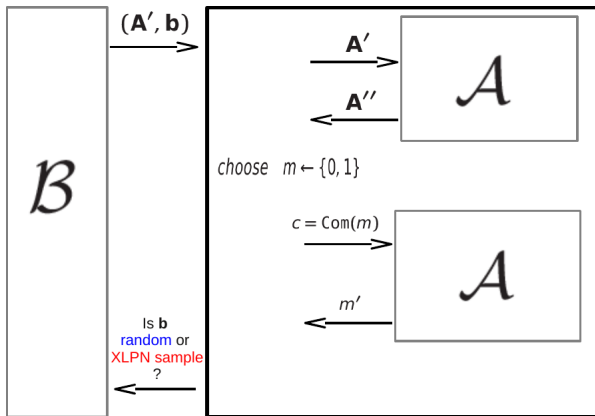
$$\text{where } b = \begin{cases} \text{random} & w.p. 1/2 \\ \mathcal{A}'s \oplus e & w.p. 1/2 \end{cases}$$

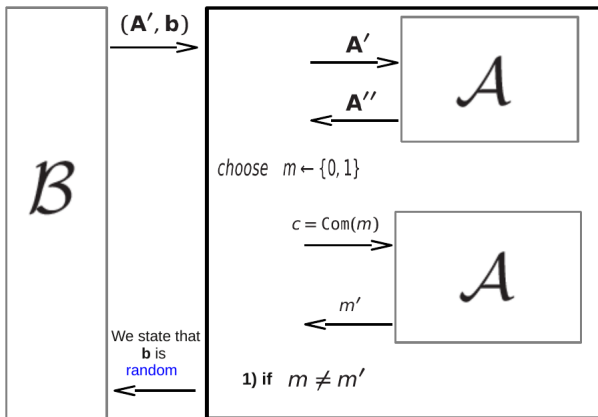










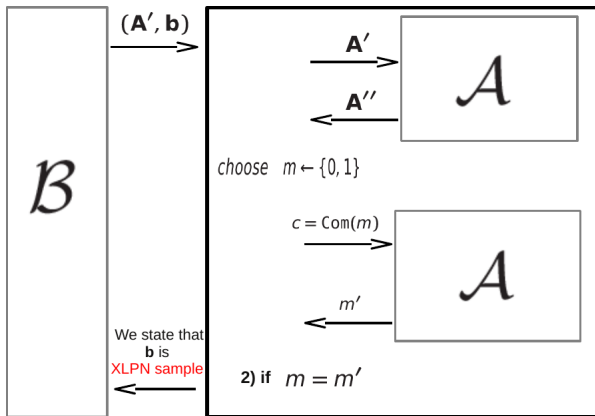


case 1)

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

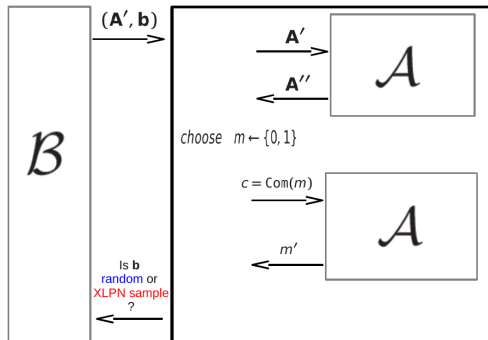
$\Rightarrow \mathcal{A}$  guesses w.p.  $\frac{1}{2}$

Exact-LPN hardness  $\Rightarrow$  Hiding commitment



case 2)

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



case 1) and 2)

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

$$\begin{aligned} \Pr(E) &= \Pr(E \mid \mathbf{b} = \mathbf{A}'\mathbf{s} \oplus \mathbf{e}) \cdot \Pr(\mathbf{b} = \mathbf{A}'\mathbf{s} \oplus \mathbf{e}) + \Pr(E \mid \mathbf{b} \text{ is random}) \cdot \Pr(\mathbf{b} \text{ is random}) \\ &= 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4} \gg 2^{-k} \end{aligned}$$

Exact-LPN hardness  $\Rightarrow$  Commitment protocol computational hiding

# Conclusions and Open Problems

## LPN open problems

- relation between standard LPN and some variants
- 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?

## Our contribution

- study the security of our Threshold Public-Key Encryption scheme in the *malicious model*
- find statistically hiding commitments
- find efficient statistically binding commitments