

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:
 - cryptographic assumptions broken by efficient quantum algorithms
e.g. factoring and discrete-logarithm broken by Shor's algorithm
 - proofs of security (or reductions) become unusable

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 reductions) become unusable

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 *reductions*) 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 *reductions*) 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 *reductions*) become useless

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 Threshold Public Key Encryption scheme 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 ℓ (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. $\Pr(e_i = 1) = \tau$

Decisional distinguish (A, τ) from uniform $(A, 0)$

decisional and search LPN are “polynomially equivalent”

Learning Parity with Noise Problem LPN

- Dimension ℓ (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”

$$\begin{array}{ccc} \mathbf{a}_1 & \xleftarrow{R} \mathbb{Z}_2^\ell & , \quad \mathbf{b}_1 \\ & \vdots & \\ \mathbf{a}_q & \xleftarrow{R} \mathbb{Z}_2^\ell & , \quad \mathbf{b}_q \end{array}$$

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

Goal: find \mathbf{s} such that $\langle \mathbf{a}_i, \mathbf{s} \rangle = \mathbf{b}_i$ for all i
 (if $\tau = 0$, this is the standard LPN problem)
 (if $\tau > 0$, this is the Learning Parity with Noise problem)

Learning Parity with Noise Problem LPN

- Dimension ℓ (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”

$$\begin{aligned}
 \mathbf{a}_1 &\stackrel{R}{\leftarrow} \mathbb{Z}_2^\ell, & \mathbf{b}_1 &= \langle \mathbf{a}_1, \mathbf{s} \rangle \oplus e_1 \\
 & & \vdots & \\
 \mathbf{a}_q &\stackrel{R}{\leftarrow} \mathbb{Z}_2^\ell, & \mathbf{b}_q &= \langle \mathbf{a}_q, \mathbf{s} \rangle \oplus e_q
 \end{aligned}$$

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

Decisional: distinguish (\mathbf{A}, \mathbf{b}) from uniform (\mathbf{A}, \mathbf{b})

• **Decisional and search LPN are computationally equivalent**

Learning Parity with Noise Problem LPN

- Dimension ℓ (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. $\Pr(e_i = 1) = \tau$

Decisional: distinguish (\mathbf{A}, \mathbf{b}) from uniform (\mathbf{A}, \mathbf{b})

decisional and search LPN are “polynomially equivalent”

Learning Parity with Noise Problem LPN

- Dimension ℓ (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. $\Pr(e_i = 1) = \tau$

- **Decisional:** distinguish (\mathbf{A}, \mathbf{b}) from **uniform** (\mathbf{A}, \mathbf{b})
- *decisional and search LPN are “polynomially equivalent”*

Learning Parity with Noise Problem LPN

- Dimension ℓ (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. $\Pr(e_i = 1) = \tau$

- **Decisional:** distinguish (\mathbf{A}, \mathbf{b}) from uniform (\mathbf{A}, \mathbf{b})
- *decisional* and *search* LPN are “*polynomially 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 $q = \text{poly}(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where ℓ is the security parameter

Interesting features

- LPN is a natural model for learning noisy devices (e.g. RFID)
- LPN is a natural model for learning noisy channels

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 $q = \text{poly}(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where ℓ is the security parameter

Interesting features

Efficiency \Rightarrow suitable for limited computing power devices (e.g. RFID).

Can be used as a pseudo-random generator.

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 $q = \text{poly}(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where ℓ 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 $q = \text{poly}(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where ℓ 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

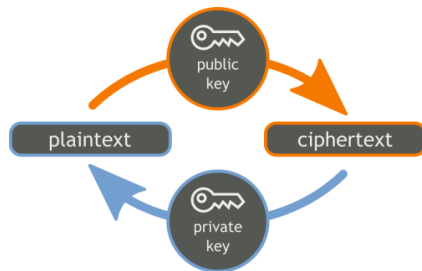
- $2^{\Theta(\ell/\log \ell)}$ having the same number of samples q
- $2^{\Theta(\ell/\log \log \ell)}$ having $q = \text{poly}(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

where ℓ is the security parameter

Interesting features

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

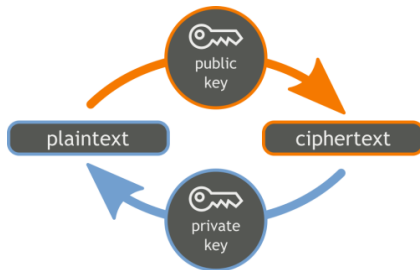
Public-Key Encryption schemes



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

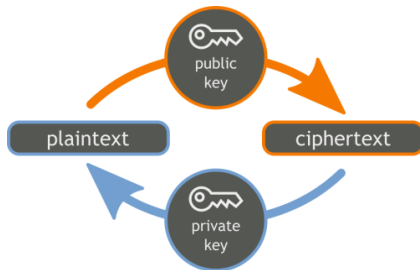
Public-Key Encryption schemes



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

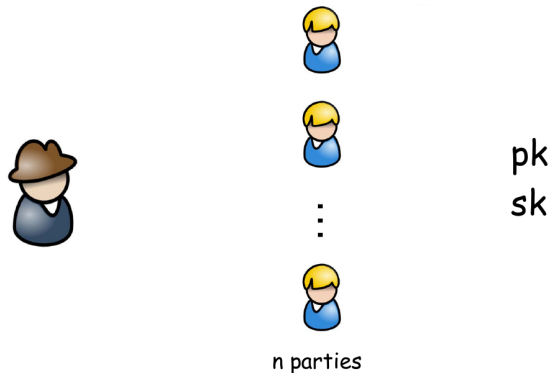
Public-Key Encryption schemes



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**

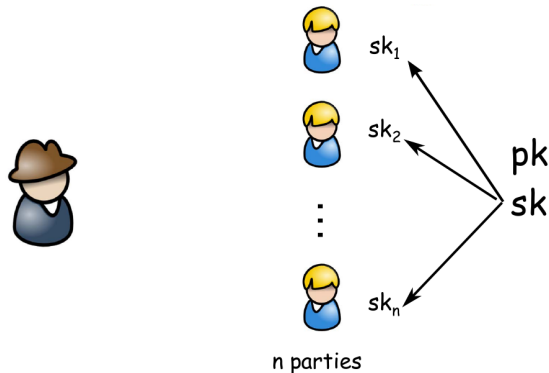
Threshold Public-Key Encryption schemes



Solution: Threshold PKE

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

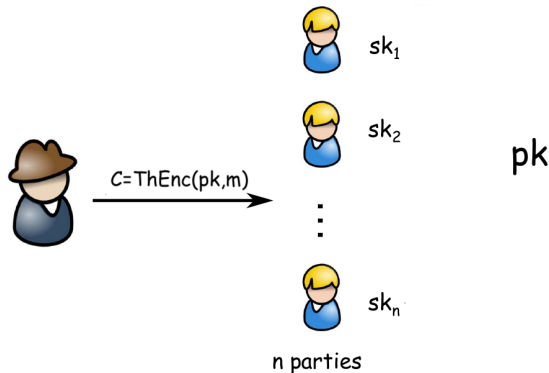
Threshold Public-Key Encryption schemes



Solution: Threshold PKE

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

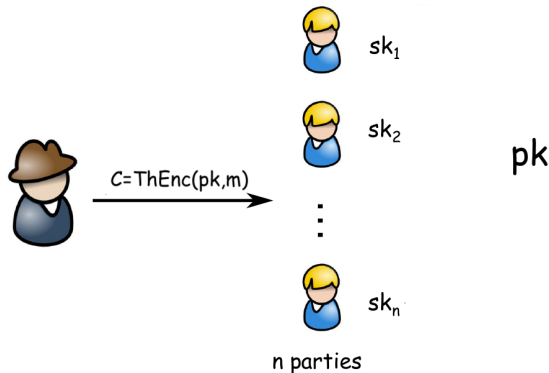
Threshold Public-Key Encryption schemes



Solution: Threshold PKE

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

Threshold Public-Key Encryption schemes

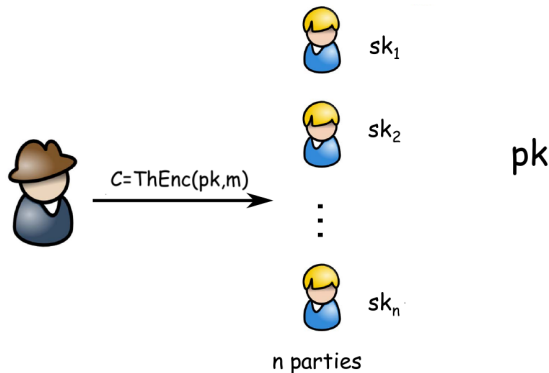


Our contribution

A **Threshold Public-Key Encryption** scheme which is:

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

Threshold Public-Key Encryption schemes

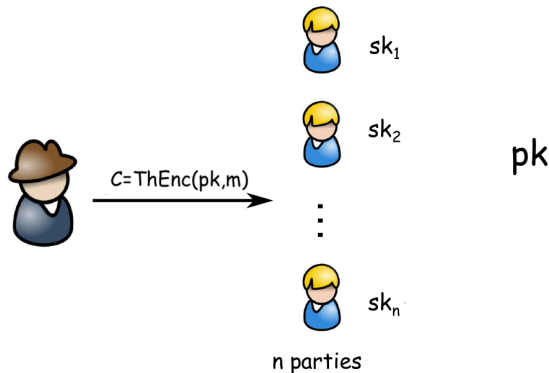


Our contribution

A **Threshold Public-Key Encryption** scheme which is:

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

Threshold Public-Key Encryption schemes



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 \mathbf{R} 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 $\underline{\mathbf{S}}$

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

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

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

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

Alekhnovich PKE scheme

Key Generation

The receiver \mathbf{R} 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 $\underline{\mathbf{S}}$

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

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

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

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

Alekhnovich PKE scheme

Key Generation

The receiver \mathbf{R} 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 $\underline{\mathbf{S}}$

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

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 \mathbf{R} computes $d = c \oplus \langle \mathbf{s}, \mathbf{u} \rangle$

Alekhnovich PKE scheme

Key Generation

The receiver **R** 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 $\mathbf{u} = \mathbf{f} \cdot \mathbf{A}$

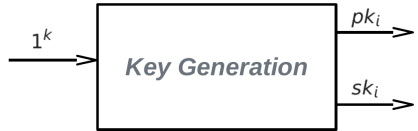
$$c = \langle \mathbf{f}, \mathbf{b} \rangle \oplus m \xrightarrow{(\mathbf{u}, c)}$$

Decryption

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

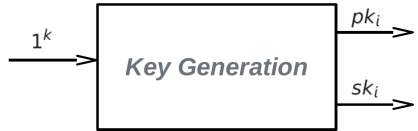
ThPKE: Protocol phases

- **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

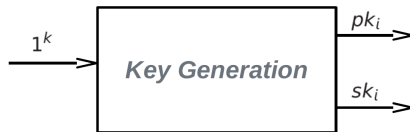


Key Generation

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

ThPKE: Protocol phases

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

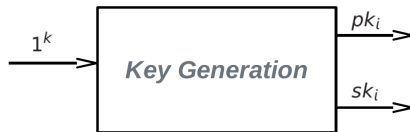


Key Generation

- 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 $(\mathbf{A}, \mathbf{b}_i = \mathbf{A}\mathbf{s}_i \oplus \mathbf{e}_i)$

ThPKE: Protocol phases

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

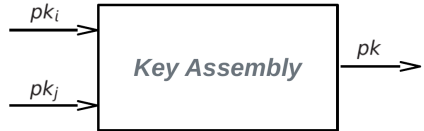


Key Generation

- 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 $(\mathbf{A}, \mathbf{b}_i = \mathbf{A}\mathbf{s}_i \oplus \mathbf{e}_i)$

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 (A, b) , where

$$b = \bigoplus_{i \in I} b_i$$

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 S



Receivers R_i, R_j

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

ThPKE: Protocol phases

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

Sender S



Receivers R_i, R_j

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

Encryption function (Alekhnovich scheme)

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

ThPKE: Protocol phases

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

Sender S



Receivers R_i, R_j

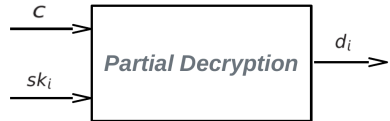
$$(C_1, c_2) \leftarrow \text{ThLPN.Enc}(m, b) \xrightarrow{(C_1, c_2)}$$

Encryption function (Alekhnovich scheme)

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

ThPKE: Protocol phases

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

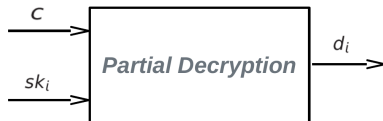


ThPKE: Protocol phases

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

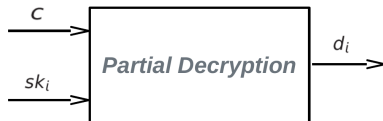
Receiver $\underline{R_i}$

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

Receiver $\underline{R_j}$

ThPKE: Protocol phases

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

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

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

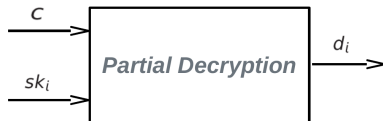
Partial decryption function (Alekhnovich scheme)

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

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

ThPKE: Protocol phases

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

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

ThPKE: Protocol phases

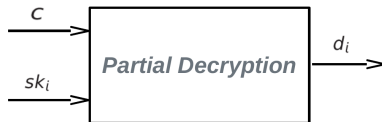
- Key Generation
- Key Assembly
- Encryption
- **Partial Decryption**
- Finish Decryption

Receiver $\underline{R_i}$

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

Receiver $\underline{R_j}$

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



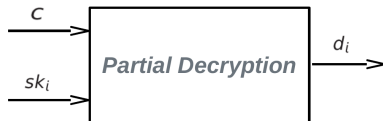
Partial decryption function (Alekhnovich scheme)

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

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

ThPKE: Protocol phases

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

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

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

Partial decryption function (Alekhnovich scheme)

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

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

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 **independently** computes the vector

$$d = c_2 \bigoplus_{i \in I} (d_i)$$

- the bit in 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 **independently** computes the vector

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

- the bit in 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 **independently** computes the vector

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

- the bit in \mathbf{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 R_i is generating LPN samples

$$d_i = C_i \cdot x_i \oplus v_i$$

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 R_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 \mathbf{R}_i is generating LPN samples

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

Protocol Security Analysis

Relaxed Semi-honest model

- Semi-honest model is not realistic
 - *replay attacks*: the same message is fraudolently encrypted multiple times
 - **Problem**: it is possible to recover information about the secret key from the ciphertexts

Proposed solutions

1. implement the receiver as a physical machine (not good in reverse engineered devices)
2. make use of *secret sharing* schemes (i.e. deterministic algorithms that split a secret into multiple shares)

Protocol Security Analysis

Relaxed Semi-honest model

- Semi-honest model is not realistic
- *replay attacks*: the same message is fraudolently encrypted multiple times
- **Problem**: it is possible to recover information about the secret key from the ciphertexts

Proposed solutions

1. Implement the decryption as a distributed computation (not good in practice, see next lecture)
2. Make use of homomorphic encryption (i.e. determine in advance the number of encryptions that can be applied to the message)

Protocol Security Analysis

Relaxed Semi-honest model

- Semi-honest model is not realistic
- *replay attacks*: the same message is fraudolently encrypted multiple times
- **Problem**: 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 a secure channel (the sender can't do anything if the channel is not secure)

Protocol Security Analysis

Relaxed Semi-honest model

- Semi-honest model is not realistic
- *replay attacks*: the same message is fraudolently encrypted multiple times
- **Problem**: 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 is not realistic
- *replay attacks*: the same message is fraudolently encrypted multiple times
- **Problem**: 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 is not realistic
- *replay attacks*: the same message is fraudolently encrypted multiple times
- **Problem**: 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 τ imply anything about LPN with $\tau' < \tau$?

is there a threshold?

how to get some back guarantees from 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 τ imply anything about LPN with $\tau' < \tau$?
Is there a threshold?
- how to get some basic primitives from 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 τ imply anything about LPN with $\tau' < \tau$?
Is there a threshold?
- how to get some basic primitives from 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 τ imply anything about LPN with $\tau' < \tau$?
Is there a threshold?
- how to get some basic primitives from 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 decommits to reveal the commitment, and reveal the message. Bob also learns the value committed (debinding property)

Our contribution

We presented a **Commitment protocol**

• based on the commitment protocol by Jakobsson

• based on Reed-Solomon polynomials (where $m(x) = [x^d + 1]$)

• the decommitment is done by the 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 Jakobsson

• based on ElGamal PKE problem (where $\mathcal{G} = \mathbb{Z}_p^*$)

• we proved the **hiding** property

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 decommit

based on the commitment protocol by decommit

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*

<https://arxiv.org/abs/1406.2656> [14]

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 $\text{wt}(e) = \lfloor \tau \cdot \ell \rfloor$)
- 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 $\text{wt}(e) = \lfloor \tau \cdot \ell \rfloor$)
- 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 $\mathbf{wt}(e) = \lfloor \tau \cdot \ell \rfloor$)
- 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 $\mathbf{wt}(e) = \lfloor \tau \cdot \ell \rfloor$)
- does not need a trusted third party

The commitment protocol by *Jain et al*

Setup Phase

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

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 $w = \lfloor \tau k \rfloor$.

The commitment protocol by *Jain et al*

Setup Phase

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

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. $wt(\mathbf{e}) = w$

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

The commitment protocol by *Jain et al*

Setup Phase

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

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. $wt(\mathbf{e}) = w$

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

The commitment protocol by *Jain et al*

Setup Phase

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

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. $wt(\mathbf{e}) = w$

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

Opening phase

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

The commitment protocol by *Jain et al*

Setup Phase

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

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

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

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

$\xleftarrow{\text{Yes, No}}$ accepts iff $\text{wt}(\mathbf{e}') = w$

Proposed commitment protocol

Problem

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

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 $\mathbf{A} = [\mathbf{A}' \parallel \mathbf{A}'']$

Solution:

Setup phase

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

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

chooses $\mathbf{A}' \xleftarrow{R} \mathbb{Z}_2^{k \times \ell} \xrightarrow{\mathbf{A}'}$

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

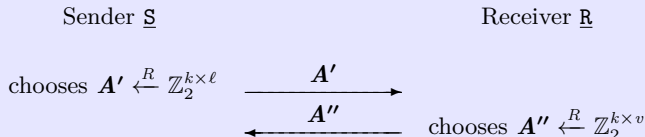
Proposed commitment protocol

Problem

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

Solution:

Setup phase



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

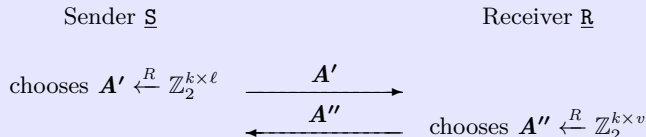
Proposed commitment protocol

Problem

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

Solution:

Setup phase



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 $w' = 2 \cdot \lfloor \tau k \rfloor$ and we choose e such that $wt(e) \leq w'$

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

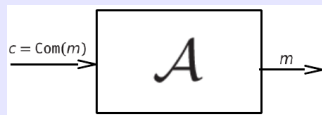
Proof

Statistically binding

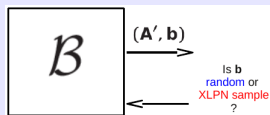
even if \mathcal{S} is computationally unbounded she cannot cheat with probability greater than 2^{-k}

Computationally hiding

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

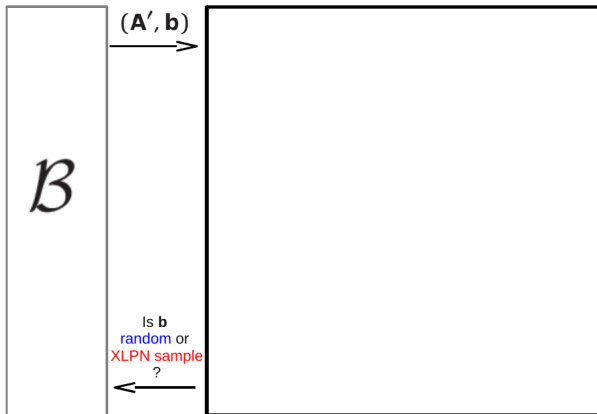


- Let \mathcal{B} an oracle

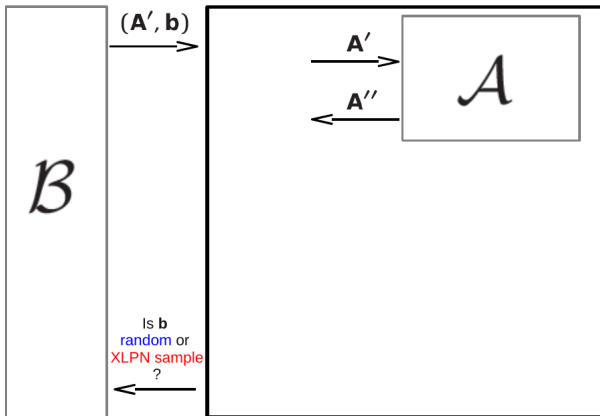


$$\text{where } b = \begin{cases} \text{random} & w.p. 1/2 \\ \mathcal{A}'s \oplus e & w.p. 1/2 \end{cases}$$

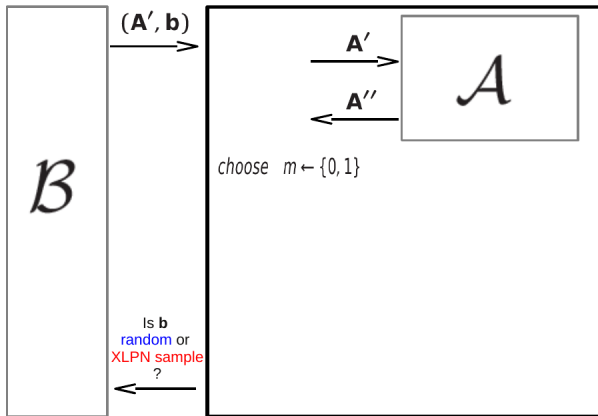
Proof



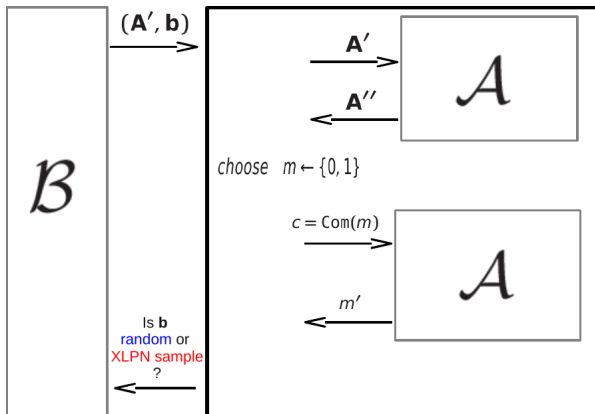
Proof



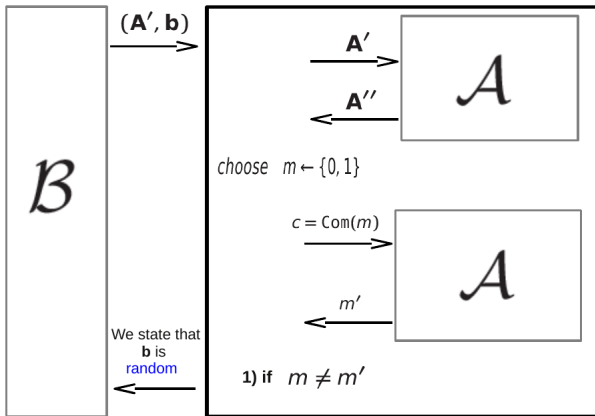
Proof



Proof



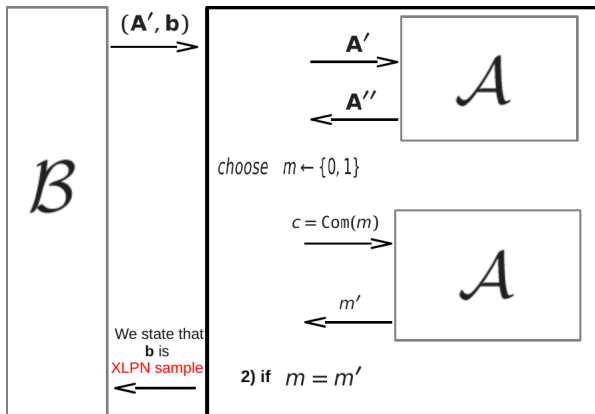
Proof



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. $\frac{1}{2}$

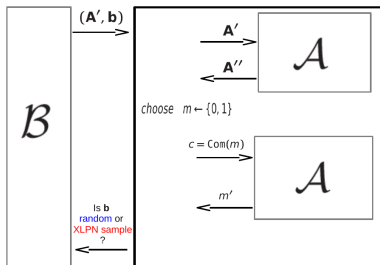
Proof



case 2)

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

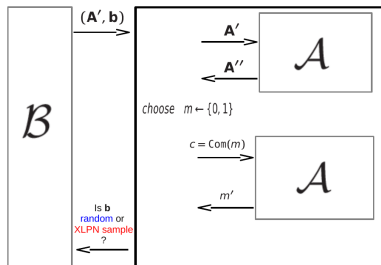
Proof



case 1) and 2)

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

Proof

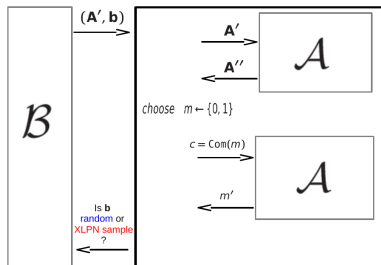


case 1) and 2)

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

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

Proof

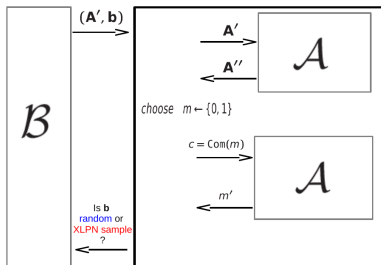


case 1) and 2)

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

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

Proof



case 1) and 2)

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

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

Exact-LPN hardness \Rightarrow Hiding commitment