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

- Il commercio elettronico non è ancora percepito come sicuro
- Risulta difficile proteggere il diritto d'autore
- tecnologie disponibili: protocolli Buyer-Seller

Learning Parity with Noise Problem LPN

- Dimension ℓ (security parameter), $q \gg \ell$
- **Search**: find $\boldsymbol{s} \in \mathbb{Z}_2^\ell$ given "noisy random inner products"

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

$$\boldsymbol{a_2} \xleftarrow{R} \mathbb{Z}_2^\ell \quad , \quad b_2 =< \boldsymbol{a_2} \, , \, \boldsymbol{s} > \oplus e_2$$

$$\vdots$$

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

Errors $e_i \leftarrow \chi =$ Bernoulli over $\mathbb{Z}_2$, param $\tau \in \left(0, \frac{1}{2}\right]$

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

Learning Parity with Noise Problem LPN

- Dimension ℓ (security parameter), $q \gg \ell$
- **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 $e_i \leftarrow \chi =$ Bernoulli over $\mathbb{Z}_2$, param $\tau \in \left(0, \frac{1}{2}\right]$

- **Decision**: distinguish $(\boldsymbol{a_i}, \boldsymbol{b_i})$ from uniform $(\boldsymbol{a_i}, b_i)$
- decisional and search LPN are *"polinomially 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 $q = poly(\ell)$ samples
- $2^{\Theta(\ell)}$ having $q = \Theta(\ell)$ samples

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 $\mathtt{S}$ 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 $\boldsymbol{e} \leftarrow \mathrm{Ber}_{\tau}^{q}$ 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 $d = c \oplus \langle \boldsymbol{s}, \boldsymbol{u} \rangle = \cdots = \langle \boldsymbol{f}, \boldsymbol{e} \rangle \oplus m$

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 $\boldsymbol{A} \xleftarrow{R} \mathbb{Z}_2^{q \times \ell}$
- 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 $(\boldsymbol{A}, \boldsymbol{b_i} = \boldsymbol{A}\boldsymbol{s_i} \oplus \boldsymbol{e_i})$

Key Assembly

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

Encryption Phase

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)

$$\boldsymbol{C_1} = \boldsymbol{F} \cdot \boldsymbol{A},$$

$$\boldsymbol{c_2} = \boldsymbol{F} \cdot \boldsymbol{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

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

Encryption Phase

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)

$$\boldsymbol{C_1} = \boldsymbol{F} \cdot \boldsymbol{A},$$

$$\boldsymbol{c_2} = \boldsymbol{F} \cdot \boldsymbol{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

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

Encryption Phase

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

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

Encryption function (Alekhnovich scheme)

$$\boldsymbol{C_1} = \boldsymbol{F} \cdot \boldsymbol{A},$$

$$\boldsymbol{c_2} = \boldsymbol{F} \cdot \boldsymbol{b} \oplus \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix} \cdot m$$

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

Partial Decryption Phase

Receiver $\underline{\mathtt{R_i}}$	Receiver $\underline{\mathtt{R_j}}$
$\boldsymbol{d_i} \leftarrow \texttt{ThLPN.Pdec}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i})$	

Partial Decryption Phase

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

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

Partial decryption function (Alekhnovich scheme)

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

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

Partial Decryption Phase

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

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

Partial decryption function (Alekhnovich scheme)

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

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

Partial Decryption Phase

Receiver $\underline{\mathtt{R_i}}$		Receiver $\underline{\mathtt{R_j}}$
$\boldsymbol{d_i} \leftarrow \textcolor{red}{\mathtt{ThLPN.Pdec}}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_i})$	$\xrightarrow{\boldsymbol{d_i}}$	
		$\boldsymbol{d_j} \leftarrow \textcolor{red}{\mathtt{ThLPN.Pdec}}(\boldsymbol{C_1}, \boldsymbol{c_2}, \boldsymbol{s_j})$

Partial decryption function (Alekhnovich scheme)

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

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

Partial Decryption Phase

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

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

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

Partial decryption function (Alekhnovich scheme)

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

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

Partial Decryption Phase

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

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

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

Finish decryption

- Each receiver indipendently computes the vector

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

- the bit in the vector $\boldsymbol{d}$ 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:** it follows directly from the Alekhnovich's scheme security
- **Decryption:** it follows directly from the LPN hardness assumption, as each $\mathtt{R}_i$ is generating LPN samples

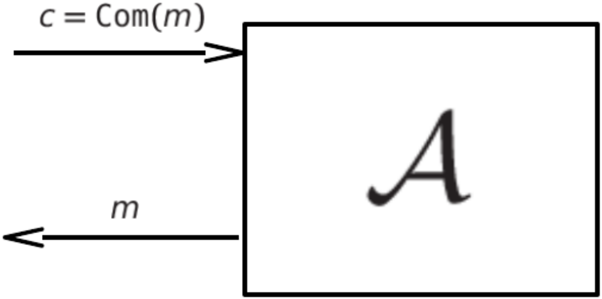
Relaxed Semi-honest model

- 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


$c = \mathrm{Com}(m)$
$\mathcal{A}$
m