LECTURE 5
MCELIECE AND ALEKHNOVICH ENCRYPTION' SCHEMES

Summer School: *Introduction to Quantum-Safe Cryptography*

Thomas Debris-Alazard

July 05, 2024

Inria, École Polytechnique

THE OBJECTIVE OF THE DAY

Code-Based Encryption Schemes

- McEliece ('78)
- Alekhnovich ('03)

$\longrightarrow$ Focus on the proof that their security *reduces* to the hardness of decoding random codes

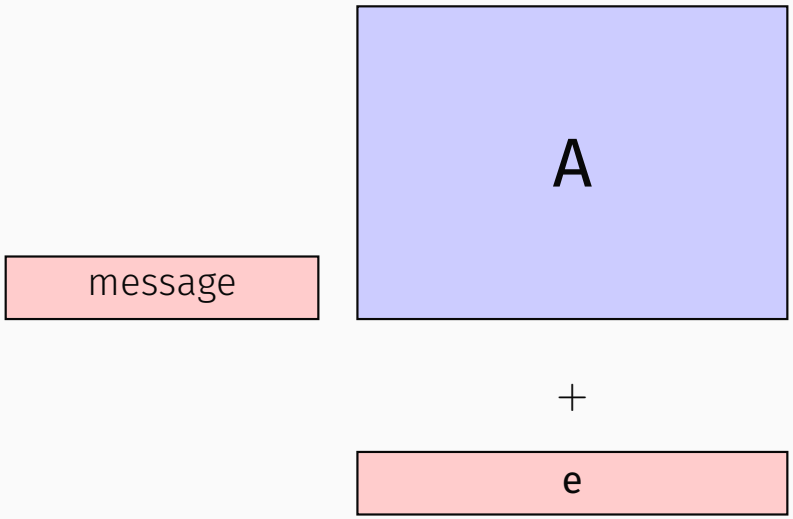
COURSE OUTLINE

- McEliece's Public-Key Encryption Scheme
- How to Rely the Encryption Security on only DP: Alekhnovich's Approach
- Search-to-Decision Reduction
- Modern Instantiations of Alekhnovich's Approach

McEliece's encryption

McEliece (1978):

$\mathbf{A} \leftarrow \text{Trapdoor}()$: public-key



- With the trapdoor: easy to recover message if $\mathbf{e}$ "short" (with few 1, a lot of 0),
- Without: hard

BE CAREFUL

What is a trapdoor in McEliece's approach?

Key Generation:

- $(\mathbf{G}_{pk}, t, T) \leftarrow$ Trappdoor() where $\mathbf{G}_{pk}$ represents a code such that,

$$(\mathbf{m}\mathbf{G}_{pk} + \mathbf{e}, T) \xrightarrow{\text{easy}} \mathbf{m} \quad \left(\text{if } |\mathbf{e}| \leq t\right)$$

- Secret Key: T
- Public Key: $\left(\mathbf{G}_{pk}, t\right)$

Encryption of m with Public-Key:

Pick random $\mathbf{e} \in \{\mathbf{z} \ : \ |\mathbf{z}| = t\}$ and output as ciphertext

$$\mathbf{m}\mathbf{G}_{pk} + \mathbf{e}$$

Decryption of $\mathbf{m}\mathbf{G}_{pk} + \mathbf{e}$:

Use T to compute

$$(\mathbf{m}\mathbf{G}_{pk} + \mathbf{e}, T) \longrightarrow \mathbf{m}$$

Berlekamp-Welsh Algorithm:

Codes that we know how to decode: $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

- Public Key: a representation of $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix} \begin{pmatrix} z_1 & & & 0 \\ & z_2 & & \\ & & \ddots & \\ 0 & & & z_n \end{pmatrix}$$

- Secret Key:

What is the secret key? Can we give the above matrix as a public key?

Berlekamp-Welsh Algorithm:

Codes that we know how to decode: $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

- Public Key: a representation of $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

$$\mathbf{G}_{\mathrm{pk}} = \mathbf{S} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix} \begin{pmatrix} z_1 & & & 0 \\ & z_2 & & \\ & & \ddots & \\ 0 & & & z_n \end{pmatrix}$$ where $\mathbf{S}$ non-singular picks random basis

- Secret Key: $T = (\mathbf{x}, \mathbf{z})$

Is the security of this scheme well-identified? Does it reduce to well identified problems?

Berlekamp-Welsh Algorithm:

Codes that we know how to decode: $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

- Public Key: a representation of $\mathrm{GRS}_k(\mathbf{x}, \mathbf{z})$

$$\mathbf{G}_{\mathrm{pk}} = \mathbf{S} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^k & x_2^k & \cdots & x_n^k \end{pmatrix} \begin{pmatrix} z_1 & & & 0 \\ & z_2 & & \\ & & \ddots & \\ 0 & & & z_n \end{pmatrix}$$ where $\mathbf{S}$ non-singular picks random basis

- Secret Key: $T = (\mathbf{x}, \mathbf{z})$

Is the security of this scheme well-identified? Does it reduce to well identified problems?

Yes!

McEliece's Encryption:

pk: $\mathbf{G}_{pk}$ representation of a code *and* sk: a trapdoor T

The security of McEliece relies on 2 **assumptions:**

1. The hardness of DP,
2. We can't distinguish $\mathbf{G}_{pk}$ and $\mathbf{G}_u$ $\left(\text{uniform}\right)$.

But how to prove this statement?

Computational Distance:

Given two random variables X_0, X_1 and T,

$$\Delta_C(X_0, X_1)(T) \stackrel{\text{def}}{=} \max_{\mathcal{A}:\ |\mathcal{A}| \leq T} \left\{ \mathbb{P}_{X_0}\left(\mathcal{A}(x_0) = 0\right) - \mathbb{P}_{X_1}\left(\mathcal{A}(x_1) = 1\right) \right\}$$

where $|\mathcal{A}|$ denotes the running-time of the algorithm $\mathcal{A}$.

Security Reduction of McEliece's Encryption:

Given G_{pk} be distributed as a public-key in McEliece's encryption and G_u being uniform with the same size,

$$\max_{|\mathcal{A}| \leq T} \left\{ \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{m}\mathsf{G}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right) \right\} \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right)(T) + \underbrace{\max_{|\mathcal{A}| \leq T} \left\{ \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{m}\mathsf{G}_u + \mathsf{e}\right) = \mathsf{e}\right) \right\}}_{\text{solving decoding problem}}$$

G_{pk} is computationally indistinguishable from G_u, is equivalent to

$\Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right)(T)$ is small

Proof.

Let $\mathcal{A}$ be an algorithm breaking McEliece.

1. We Claim:

$$\mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right) \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right) + \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

Let us design $\mathcal{B}$ an algorithm to distinguish G_{pk} and G_{pk} with the help of $\mathcal{A}$

- Given $\mathsf{G}_?$

(*i*) We pick a uniform m and e with Hamming weight t

(*ii*) We feed $(\mathsf{G}_?, \mathsf{mG}_? + \mathsf{e})$ to $\mathcal{A}$

(*iii*) If the answer is m, we decide that G_{pk} were given, otherwise G_u

We have,

$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_{\mathsf{pk}}\right) = \text{“pk”}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right)$$
$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk”}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

Proof.

Let $\mathcal{A}$ be an algorithm breaking McEliece.

1. We Claim:

$$\mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right) \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right) + \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

Let us design $\mathcal{B}$ an algorithm to distinguish G_{pk} and G_{pk} with the help of $\mathcal{A}$

- Given $\mathsf{G}_?$

(*i*) We pick a uniform m and e with Hamming weight t

(*ii*) We feed $(\mathsf{G}_?, \mathsf{mG}_? + \mathsf{e})$ to $\mathcal{A}$

(*iii*) If the answer is m, we decide that G_{pk} were given, otherwise G_u

We have,

$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_{\mathsf{pk}}\right) = \text{“pk''}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right)$$
$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk''}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

But at the same time,

$$\left|\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_{\mathsf{pk}}\right) = \text{“pk''}\right) - \mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk''}\right)\right| \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right)(T)$$

Proof.

Let $\mathcal{A}$ be an algorithm breaking McEliece.

1. We Claim:

$$\mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right) \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right) + \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

Let us design $\mathcal{B}$ an algorithm to distinguish G_{pk} and G_{pk} with the help of $\mathcal{A}$

- Given $\mathsf{G}_?$

(i) We pick a uniform m and e with Hamming weight t

(ii) We feed $(\mathsf{G}_?, \mathsf{mG}_? + \mathsf{e})$ to $\mathcal{A}$

(iii) If the answer is m, we decide that G_{pk} were given, otherwise G_u

We have,

$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_{\mathsf{pk}}\right) = \text{“pk''}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{mG}_{\mathsf{pk}} + \mathsf{e}\right) = \mathsf{m}\right)$$
$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk''}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right)$$

But at the same time,

$$\left|\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_{\mathsf{pk}}\right) = \text{“pk''}\right) - \mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk''}\right)\right| \leq \Delta_C\left(\mathsf{G}_{\mathsf{pk}}, \mathsf{G}_u\right)(T)$$

2. It concludes the proof as

$$\mathbb{P}\left(\mathcal{B}\left(\mathsf{G}_u\right) = \text{“pk''}\right) = \mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{m}\right) \leq \max_{|\mathcal{A}| \leq T}\left\{\mathbb{P}\left(\mathcal{A}\left(\mathsf{G}_u, \mathsf{mG}_u + \mathsf{e}\right) = \mathsf{e}\right)\right\} \quad \square$$

PLENTY OF INSTANTIATION, AND ATTACKS...

Can we distinguish the public code from a random one?

Be extremely careful. . .

We believe in the hardness of DP,

but, when instantiating McEliece with codes equipped with a decoding algorithm:

these codes have to be indistinguishable from random codes

$\longrightarrow$ Many family of codes were historically proposed to instantiate McEliece's encryption

Many were broken. . .

- Generalized Reed-Solomon codes (see Exercise Session)
- Convolutional codes
- Polar codes
- LDPC codes
- etc. . .

Original McEliece's instantiation: Goppa codes (I)

Alternant Codes:

Let k, n, m such that $n \leq q^m$, $\mathbf{x} \in \mathbb{F}_{q^m}^n$ where the x_i's are distinct.

We define the alternant code $\mathcal{A}(\mathbf{x}, \mathbf{y})$ as,

$$\mathcal{A}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \mathrm{GRS}_k(\mathbf{x}, \mathbf{y}) \cap \mathbb{F}_q^n$$

$\longrightarrow$ Alternant codes belong to the class of algebraic codes!

Decoding Alternant Codes:

Given $\mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{A}(\mathbf{x}, \mathbf{y})$, we can decode using Berlekamp-Welsh algorithm

by considering them as vector of $\mathbb{F}_{q^m}$

$\longrightarrow$ Goppa codes are a particular case of alternant codes!

Original McEliece's instantiation: Goppa codes (I)

Goppa Codes:

Given some polynomial $G \in \mathbb{F}_{q^m}[X]$, its associated Goppa codes is defined as

$$\Gamma(G, \mathbf{x}) \stackrel{\text{def}}{=} \mathcal{A}(\mathbf{x}, \mathbf{y}) \quad \text{where } \forall i \in [1, n], y_i = \frac{G(x_i)}{\prod_{i \neq j}(x_i - x_j)}$$

$\longrightarrow$ Goppa Codes enjoy many interesting properties!

- We can decode them at twice the Berlekamp-Welsh decoding distance
- They behave as random codes for their weight distribution and minimum distance

Goppa Codes in Cryptography:

Since their introduction by McEliece for cryptography in '78, we did not succeed to design an efficient algorithm to distinguish between Goppa and random codes

$\Big($except when Goppa codes have rate very close to 1$\Big)$

McEliece with parity-check matrices (I)

McEliece historically described his scheme with generator matrices

Does it work with parity-check matrices?

McEliece with parity-check matrices (I)

McEliece historically described his scheme with generator matrices

Does it work with parity-check matrices?

$\longrightarrow$ Yes! It is often referred to as Niederreiter's approach

McEliece with parity-check matrices (II)

Key Generation:

- $(\mathbf{H}_{pk}, t, T) \leftarrow$ Trappdoor() where $\mathbf{H}_{pk}$ represents a code such that,

$$(\mathbf{H}_{pk}\mathbf{e}^\top, T) \xrightarrow{\text{easy}} \mathbf{e} \quad \Big(\text{if } |\mathbf{e}| \leq t\Big)$$

- Secret Key: T
- Public Key: $\Big(\mathbf{H}_{pk}, t\Big)$

Encryption of m with Public-Key:

We first encode $\mathbf{m}$ as $\mathbf{e}(\mathbf{m}) \in \{\mathbf{z} \ : \ |\mathbf{z}| = t\}$ via a public encoding and then output as ciphertext

$$\mathbf{H}_{pk}\mathbf{e}(\mathbf{m})^\top$$

Decryption of $\mathbf{H}_{pk}\mathbf{e}^\top$:

Use T to compute

$$(\mathbf{H}_{pk}\mathbf{e}(\mathbf{m})^\top, T) \longrightarrow \mathbf{e}(\mathbf{m}) \longrightarrow \mathbf{m}$$

A bad (but original) presentation of McEliece

https://en.wikipedia.org/wiki/McEliece_cryptosystem

https://en.wikipedia.org/wiki/McEliece_cryptosystem

There are no permutations in McEliece cryptosystem

About the Security:

- Both representations of the scheme have the same security
  - The decoding problem with parity-check of generator matrices is the same problem
  - If you can distinguish between $\mathbf{G}_{\text{pk}}$ and $\mathbf{G}_u$, then you can distinguish between $\mathbf{H}_{\text{pk}}$ and $\mathbf{H}_u$ because you can compute a generator from a parity-check matrix and reciprocally

About the Public-Key Size:

- Public-keys in generator or parity-check matrix can generically be represented with the same amount of bits. We can put them in "systematic form"

$$\mathbf{G}_{\text{pk}} = (\mathbf{I}_k \mid \mathbf{A}) \text{ where } \mathbf{A} \in \mathbb{F}_q^{k \times (n-k)} \quad \text{and} \quad \mathbf{H}_{\text{pk}} = (\mathbf{I}_{n-k} \mid \mathbf{B}) \text{ where } \mathbf{A} \in \mathbb{F}_q^{(n-k) \times k}$$

$$\longrightarrow \text{It requires } k \times (n-k) \times \log_2 q \text{ bits!}$$

Ciphertext Length

- With generator matrices, a ciphertext has length $n \times \log_2 q$ while with parity-check matrices it has length $(n-k) \times \log_2 q$

In McEliece scheme, keys are large: $k \times (n-k) \times \log_2 q$ bits

How to reduce key-sizes?

$\longrightarrow$ use a quasi-cyclic structure!

Quasi-Cyclic Codes:

1. First we identify $\mathbf{h} \in \mathbb{F}_2^n \cong \sum_{i=1}^{n} h_i X^{i-1} \in \mathbb{F}[X]/(X^n+1)$
2. A Quasi-Cyclic code with ℓ-blocks is a code admitting as basis

$$\begin{pmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \dots & \mathbf{g}_\ell \\ \circlearrowleft & \circlearrowleft & \dots & \circlearrowleft \end{pmatrix}$$

where we often identify vectors $\mathbf{g}_i \in \mathbb{F}_2^n$ as polynomials in $\mathbb{F}_2[X]/(X^n+1)$

$\longrightarrow$ Only one vector is necessary to represent a quasi-cyclic code!

Issue:

When using these codes in McEliece, we don't reduce the security to the hardness of the decoding problem of a random code, but the hardness of decoding a random quasi cyclic code and less is known about this problem. . .

CONCLUSION ABOUT MCELIECE APPROACH

- Two equivalent representations of McEliece's encryption: using parity-check or generator matrices

  $\longrightarrow$ Ciphers are shorter with the parity-check representation

- Security relies on decoding a random code and distinguishing the public code from a random code

  $\longrightarrow$ Be cautious when choosing a code that you know how to decode: the basis that you give as public-key as be random looking

- Public-keys are huge instead you add some algebraic structure like quasi-cyclic

  $\longrightarrow$ But it decreases the security by not relying on the hardness of decoding a random code

Two McEliece encryptions on the verge for standardization by American government (NIST)

- Classic McEliece: it uses Goppa codes as originally proposed by McEliece
- Bike: its uses quasi-cyclic MDPC codes

Alekhnovich's encryption

McEliece security is not only relying on the hardness of decoding a random code

Do we know code-based encryptions whose security only relies on the hardness of a decoding a random code?

Key Generation $\left(\mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i\right)$**:**

1. Choose $\mathbf{A} \in \{0,1\}^{k \times n}$ at random and $\mathbf{e} \in \{0,1\}^n$ with few one
2. Compute the space $\mathcal{C}_{\text{pk}} = \text{Span}\,(\mathcal{C}, \mathbf{e})^{\perp}$
3. Secret Key: $\mathbf{e}$; Public Key: $\mathcal{C}_{\text{pk}}$

To encrypt a bit $b \in \{0,1\}$ with the public key:

- If $b = 0$, choose $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ and $\mathbf{e}' \in \{0,1\}^n$ with few one. Send: $\mathbf{c} + \mathbf{e}'$
- If $b = 1$, choose $\mathbf{u} \in \{0,1\}^n$ uniformly. Send: $\mathbf{u}$

To decrypt a received word $y \in \{0,1\}^n$:

Compute

$$\mathbf{y} \cdot \mathbf{e} = \begin{cases} (\mathbf{c} + \mathbf{e}') \cdot \mathbf{e} = \underbrace{\mathbf{c} \cdot \mathbf{e}}_{=0 \text{ as } \mathbf{c} \perp \mathbf{e}} + \mathbf{e}' \cdot \mathbf{e} = 0 \text{ with prob. } \approx 1 \text{ as } \mathbf{e}, \mathbf{e}' \text{ have few one} \\ \mathbf{u} \cdot \mathbf{e} = \text{ a random bit} \end{cases}$$

$\longrightarrow$ To be successful: the encryption of a bits needs to be repeated a few number of times

WHAT ABOUT THE SECURITY?

If you want to decrypt without the secret-key

$\longrightarrow$ Distinguish between between $\mathbf{u}$ uniform and $\mathbf{c} + \mathbf{e}$ where $|\mathbf{e}|$ is small

Solving this problem does not amount to solve the average decoding problem. . .

Search-to-decision reductions

AVERAGE DECISIONAL DECODING PROBLEM

DDP(n, q, R, τ), $k \stackrel{\text{def}}{=} \lfloor Rn \rfloor$ and $t \stackrel{\text{def}}{=} \lfloor \tau n \rfloor$.

- Distributions:
  - $\mathcal{D}_0$: $(\mathbf{G}, \mathbf{u})$ be uniformly distributed over $\mathbb{F}_q^{k \times n} \times \mathbb{F}_q^n$.
  - $\mathcal{D}_1$: $(\mathbf{G}, \mathbf{mG} + \mathbf{e})$ where $\mathbf{G}$, $\mathbf{m}$, $\mathbf{e}$ being uniformly distributed over $\mathbb{F}_q^{k \times n}$, $\mathbb{F}_q^k$ and words of Hamming weight t
- **Input**: $(\mathbf{H}, \mathbf{s})$ distributed according to $\mathcal{D}_b$ where $b \in \{0, 1\}$ is uniform,
- **Decision**: $b' \in \{0, 1\}$.

Is this problem strictly easier than its search version?

$\rightarrow$ No! They are equivalent (Goldreich-Levin hardcore predicate)

How to define the "efficiency" of an algorithm solving the average decision decoding problem (DDP)?

Advantage:

The DDP(n, R, τ)-advantage of an algorithm $\mathcal{A}$ is defined as:

$$Adv^{\mathrm{DDP}(n,R,\tau)}(\mathcal{A}) \stackrel{\text{def}}{=} \frac{1}{2}\left(\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = 1 \mid b = 1\right) - \mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = 1 \mid b = 0\right)\right)$$

where the probabilities are computed over the internal randomness of $\mathcal{A}$, a uniform $b \in \{0, 1\}$ and inputs be distributed according to $\mathcal{D}_b$ which is defined in DDP(n, R, τ)

$\longrightarrow$ Any algorithm solving DDP can be turned into an algorithm solving DP?

Theorem:

Let $\mathcal{A}$ be a probabilistic algorithm running in time $T(n)$ whose DDP(n, R, τ)-advantage is given by $\varepsilon(n)$ and let $\ell(n) \stackrel{\text{def}}{=} \log(1/\varepsilon(n))$. Then it exists an algorithm $\mathcal{A}'$ that solves DDP(n, R, τ) in time $O(n^2\ell(n)^3)T(n)$ and with probability $\Omega(\varepsilon(n)^2)$

But how to prove this statement?

$\longrightarrow$ It basically relies on the following statement

Goldreich-Levin Hardcore Predicate

Let $f : \{0,1\}^* \to \{0,1\}^*$, $\mathcal{A}$ be a probabilistic algorithm running in time $T(n)$ and $\varepsilon(n) \in (0,1)$ be such that

$$\mathbb{P}\left(\mathcal{A}(f(\mathsf{x}), \mathsf{r}) = \mathsf{x} \cdot \mathsf{r}\right) = \tfrac{1}{2} + \varepsilon$$

where the probability is computed over the internal coins of $\mathcal{A}$, x and r that are uniformly distributed over $\{0,1\}^n$. Let $\ell(n) \stackrel{\text{def}}{=} \log(1/\varepsilon(n))$. Then, it exists an algorithm $\mathcal{A}'$ running in time $O\left(n^2 \ell(n)^3 T(n)\right)$ that satisfies

$$\mathbb{P}\left(\mathcal{A}'(f(\mathsf{x}) = \mathsf{x})\right) = \Omega\left(\varepsilon^2\right)$$

where the probability is computed over the internal coins of $\mathcal{A}'$ and x.

How to Prove this Statement?

The answer is not so hard! Crucial remark:

$$\mathcal{A}(f(\mathsf{x}), \mathsf{r}) = \mathsf{x} \cdot \mathsf{r} + e \text{ where } \mathbb{P}(e = 1) = 1/2 - \varepsilon(n)$$

Therefore, recovering x with the help of $\mathcal{A}$ amount to decode

$$\mathsf{x}_n \left(\mathsf{r}_{(1)}^\top \mid \ldots \mid \mathsf{r}_{(N)}^\top\right) + (e_1, \ldots, e_N) \text{ where } \mathbb{P}(e_i = 1) = 1/2 - \varepsilon < 1/2 \left(\text{not uniform noise}\right)$$

On Board

- Though Alekhnovich's cryptosystem is proved to be secure under the hardness of the average decoding problem, it is not efficient

$$\longrightarrow n \text{ (few thousand) bits to encrypt one bit...}$$

- It exists a second Alekhnovich's cryptosystem also proved to be secure under the hardness of the average decoding problem but efficient

$$\longrightarrow O(n) \text{ bits to encrypt } n \text{ bits!}$$

Many other topics:

- Code-based primitives like signatures,
- Change the Hamming metric (for instance rank metric)
- Code-based cryptography and algebra (for instance polynomial rings via function fields)
- . . .

Many other topics:

- Code-based primitives like signatures,
- Change the Hamming metric (for instance rank metric)
- Code-based cryptography and algebra (for instance polynomial rings via function fields)
- . . .

Thank You!