# Learning with Errors

## Cryptology (1)

Martin Stanek

2025

KI FMFI UK Bratislava

# Why Learning with Errors (LWE)

- introduced by Regev (2005)
- no efficient quantum algorithm is known for LWE
- variants for better efficiency: RLWE (Ring LWE), MLWE (Module LWE)
- versatile – a basis for various schemes
  - public-key encryption
  - identity-based encryption
  - fully homomorphic encryption
  - signature schemes
- can be reduced to worst-case hardness of some problems on lattices

# LWE: linear equations with some “noise”

- notation:
  - dimension $n \in \mathbb{N}$ (primary security parameter)
  - $q \in \mathbb{N}$, usually $q = \text{poly}(n)$, sometimes $q$ is a prime number
  - secret vector $\boldsymbol{s} \in_R \mathbb{Z}_q^n$, column vectors will be used
  - matrix $\boldsymbol{A} \in_R \mathbb{Z}_q^{m \times n}$, chosen uniformly random
  - error distribution $\chi$ on $\mathbb{Z}_q$
  - for odd $q$: $\mathbb{Z}_q = \{-(q-1)/2, \ldots, (q-1)/2\}$, for example $\mathbb{Z}_{29} = \{-14, \ldots, 14\}$
  - error vector $\boldsymbol{e} = (e_1, \ldots, e_m)^T \in \mathbb{Z}_q^m$, where $e_i \leftarrow \chi$ (independent) for all $i$
  - $\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$
- sometimes an oracle formulation for LWE:
  - access to oracle $\mathcal{O}_s$ that produces $(\boldsymbol{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$
  - $\boldsymbol{a} \in_R \mathbb{Z}_q^n$ (uniform random), $e \leftarrow \chi$, $b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e$
  - above: $m$ – number of samples

# LWE problems

**Search LWE** (LWE)
Find $\boldsymbol{s}$ for given $\boldsymbol{A}$ and $\boldsymbol{b}$.

**Decisional LWE** (DLWE)
Given $(\boldsymbol{A}, \boldsymbol{c})$, where $\boldsymbol{c} = \boldsymbol{b}$ or $\boldsymbol{c} \in_R \mathbb{Z}_q^m$, both with probability $1/2$, distinguish these cases with probability non-negligible better than $1/2$.

Assumption: Search/Decisional LWE is hard for suitable parameters and error distribution.

- without noise ($\boldsymbol{e}$ is zero) – system of linear equations
  - easy (Gaussian elimination)
- too much noise ($\chi$ uniform on $\mathbb{Z}_q$)
  - any $\boldsymbol{s}$ is a plausible solution
  - DLWE with identical distributions
- Gaussian elimination increases noise (up to the point where equations have no information on $\boldsymbol{s}$)

# LWE – noise selection

- various error distributions are used
  - impact security proofs, reductions, efficiency
- discrete Gaussian distribution
  - derived from a (continuos) normal distribution
  - defined for $x \in \mathbb{Z}$ and scaled accordingly
  - used with mean 0 and small standard deviation
- centered binomial distribution (easier sampling)
  - "shifted" symmetrical binomial distribution with mean 0, small standard deviation
  - defined on $\{-\beta, \ldots, \beta\}$, for a small bound $\beta$: $\Pr[X = k] = \binom{2\beta}{k+\beta} \cdot 2^{-2\beta}$
- centered uniform distribution
  - $\chi$ is uniform random on $\{-\beta, \ldots, \beta\}$, for a small bound $\beta$

# LWE – small example

$$\underbrace{\begin{pmatrix} 11 & 19 & 3 & 14 & 0 \\ 13 & 22 & 19 & 17 & 27 \\ 15 & 9 & 18 & 19 & 28 \\ 19 & 19 & 12 & 12 & 28 \\ 24 & 26 & 9 & 28 & 3 \\ 18 & 6 & 25 & 28 & 0 \\ 23 & 18 & 21 & 17 & 11 \\ 13 & 16 & 19 & 4 & 21 \end{pmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{pmatrix} 23 \\ 0 \\ 6 \\ 6 \\ 16 \end{pmatrix}}_{\boldsymbol{s}} + \underbrace{\begin{pmatrix} -1 \\ 0 \\ -1 \\ 0 \\ -2 \\ 2 \\ 0 \\ 1 \end{pmatrix}}_{\boldsymbol{e}} = \underbrace{\begin{pmatrix} 6 \\ 19 \\ 28 \\ 14 \\ 8 \\ 9 \\ 5 \\ 20 \end{pmatrix}}_{\boldsymbol{b}} \mod 29$$

# Naive algorithm for Search LWE

- maximum likelihood approach
- try all $\boldsymbol{s} \in \mathbb{Z}_q^n$ ($q^n$ possibilities)
  - small error $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{A} \cdot \boldsymbol{s}$ indicates a possible solution
  - the smallest error ⇒ the most probable solution
  - $l_2$ norm computed in $\mathbb{R}$: $\|\boldsymbol{e}\| = \sqrt{e_1^2 + \ldots + e_n^2}$
- depending on $\chi$, usually $\mathrm{poly}(n)$ equations are sufficient for a unique solution
- running time $O(q^n \cdot n \cdot \mathrm{poly}(n)) \approx 2^{O(n \log n)}$ for typical $q$ (polynomial in $n$)

# Decisional LWE and Search LWE are equivalent

- we can efficiently solve the problem if we have an algorithm that efficiently solves the other problem
  - efficiently ≈ PPT
  - with non-negligible probabilities
- DLWE ↦ LWE reduction is trivial:
  - input: $(\boldsymbol{A}, \boldsymbol{c})$; we have to decide if $\boldsymbol{c}$ is random or input is an LWE instance

    1. run search LWE algorithm on $(\boldsymbol{A}, \boldsymbol{c})$; let $\boldsymbol{s}$ be its output
    2. if $\boldsymbol{e} = \boldsymbol{c} - \boldsymbol{A} \cdot \boldsymbol{s}$ is small (from $\chi$) then return "*LWE instance*"
    3. otherwise return "*random*"

# LWE ↦ DLWE reduction (idea)

- input: $(\boldsymbol{A}, \boldsymbol{b})$, an LWE instance, and access to a DLWE oracle
- idea: guess and test the value of a coordinate for $\boldsymbol{s} = (s_1, \ldots, s_n)^T$
- testing if $s_1 = \sigma \in \mathbb{Z}_q$:
  1. add $\boldsymbol{r} \in_R \mathbb{Z}_q^m$ to the first column of $\boldsymbol{A}$, call it $\boldsymbol{A}'$
  2. test if $(\boldsymbol{A}', \boldsymbol{b} + \sigma\boldsymbol{r})$ is an LWE instance ($s_1 = \sigma$) or random ($s_1 \neq \sigma$)
- why this works:
  - if $s_1 = \sigma$: $\boldsymbol{b} + \sigma\boldsymbol{r} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} + \sigma\boldsymbol{r} = \boldsymbol{A}'\boldsymbol{s} + \boldsymbol{e}$ (LWE instance)
  - if $s_1 \neq \sigma$: $\boldsymbol{b} + \sigma\boldsymbol{r} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} + \sigma\boldsymbol{r} = \boldsymbol{A}'\boldsymbol{s} + \boldsymbol{e} + (\sigma - s_1)\boldsymbol{r}$ (random, since $\sigma - s_1 \neq 0$, and $\boldsymbol{r}$ is random)
- similarly for other coordinates ($i$-th column in $\boldsymbol{A}$ is modified for $s_i$ testing)
- running time: $O(nq)$ iterations

# Encryption scheme

- Regev (2005)
- bit encryption

**Initialization**

- private key: $\boldsymbol{s} \in_R \mathbb{Z}_q^n$
- public key: LWE instance $(\boldsymbol{A}, \boldsymbol{b})$
  - $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$, and $\boldsymbol{e} \leftarrow \chi^m$

**Encryption**

- plaintext $\mu \in \{0, 1\}$:
  1. $\boldsymbol{r} \in_R \{0, 1\}^m$
  2. ciphertext:
     $(\boldsymbol{a}, b) = \left(\boldsymbol{r}^T \boldsymbol{A}, \langle \boldsymbol{b}, \boldsymbol{r} \rangle + \mu \cdot \lfloor q/2 \rfloor\right)$

**Decryption** (ciphertext $(\boldsymbol{a}, b)$):

- output 0 if $|b - \boldsymbol{a}\boldsymbol{s}| < q/4$ (1 otherwise)
- correctness:

$$\begin{aligned} b - \boldsymbol{a}\boldsymbol{s} &= \langle \boldsymbol{b}, \boldsymbol{r} \rangle + \mu \cdot \lfloor q/2 \rfloor - \boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s} \\ &= \langle \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}, \boldsymbol{r} \rangle + \mu \cdot \lfloor q/2 \rfloor - \boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s} \\ &= \boldsymbol{r}^T (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) + \mu \cdot \lfloor q/2 \rfloor - \boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s} \\ &= \boldsymbol{r}^T \boldsymbol{e} + \mu \cdot \lfloor q/2 \rfloor \end{aligned}$$

# Remarks

- very simple encryption and decryption
- if $\chi$ is a distribution on $\{-\beta, \ldots, \beta\}$, then $|\boldsymbol{r}^T \boldsymbol{e}| \leq m\beta$, i.e., always correct decryption if $m\beta < q/4$
  - for LWE schemes a negligible probability of incorrect decryption is often accepted
- IND-CPA secure but not IND-CCA secure (malleable)
- improving throughput and encrypting longer plaintexts
  - divide $\mathbb{Z}_q$ into $2^d$ regions to encode $d$ bits
  - use matrices for secret and error, instead of vectors

# FrodoKEM (KEM based on a standard LWE)

- part of NIST Post-Quantum Cryptography Standardization Process
  - not selected for standardization, round 3 “alternate” algorithm for KEM
- computation in ring mod $q = 2^{15}$ for 128-bit security level, and $q = 2^{16}$ for 192 and 256-bit security levels
- public-key matrix generated from a public-key seed (pseudoradom)
- sizes of various parameters (in bytes):

| level | private key | public key | ciphertext | |
|---|---|---|---|---|
| 128 | 19 888 | 9 616 | 9 720 | FrodoKEM-640 |
| 192 | 31 296 | 15 632 | 15 744 | FrodoKEM-976 |
| 256 | 43 088 | 21 520 | 21 632 | FrodoKEM-1344 |

- more size-efficient schemes – algebraic lattices (structured), Ring-LWE, Module-LWE

# Remarks on FrodoKEM

- NIST Status Report on the 2nd Round:
  *Plain LWE itself is among the most studied and analyzed cryptographic problems in existence today.*
  *The resulting potential security advantages of FrodoKEM are paid for with far worse performance in all metrics than other lattice schemes. …*
  *Use of FrodoKEM would have a noticeable performance impact on high traffic TLS servers …*
  *FrodoKEM may be suitable for use cases where the high confidence in the security of unstructured lattice-based schemes is much more important than performance.*
- selected algorithms for KEM:
  - Crystals-Kyber: Module LWE (MLWE) problem
  - HQC: Quasi-Cyclic Syndrome Decoding (QCSD) problem

# Some performance numbers

| Algorithm | Public key (bytes) | Ciphertext (bytes) | Key gen. (ms) | Encaps. (ms) | Decaps. (ms) |
|---|---|---|---|---|---|
| ECDH NIST P-256 | 64 | 64 | 0.072 | 0.072 | 0.072 |
| ~~SIKE p434~~ | 330 | 346 | 13.763 | 22.120 | 23.734 |
| Kyber512-90s | 800 | 736 | 0.007 | 0.009 | 0.006 |
| FrodoKEM-640-AES | 9,616 | 9,720 | 1.929 | 1.048 | 1.064 |

Paquin et al.: Benchmarking Post-Quantum Cryptography in TLS, PQCrypto 2020

# Encryption scheme (inspired by FrodoKEM)

- IND-CPA secure scheme
- $q = 2^D$ for some $D \leq 16$ ($D = 15$ for Frodo-640)
- $0 \leq B < D$ (e.g. $B = 2$), $n = 640$, $n' = 8$
- support of $\chi = \{-12, \ldots, 12\}$
  - discrete Gaussian distribution with standard deviation $\sigma = 2.8$
  - sampling error matrices

**Initialization**

- pseudorandom matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times n}$ generated from a random $\text{seed}_A$
- sample error matrices $\boldsymbol{S}, \boldsymbol{E}$ $(n \times n')$
- compute $\boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{E}$ $(n \times n')$
- public key: $\text{seed}_A, \boldsymbol{B}$
- private key: $\boldsymbol{S}$

# Encryption and encode function

**Encryption** (plaintext $\mu = \{0,1\}^{B \cdot n' \cdot n'}$, 128-bit string for our parameters):

1. compute $\boldsymbol{A}$ from $\text{seed}_A$
2. sample error matrices: $\boldsymbol{S}'$ and $\boldsymbol{E}'$ $(n' \times n)$, and $\boldsymbol{E}''$ $(n' \times n')$
3. compute $\boldsymbol{B}' = \boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}'$ $(n' \times n)$, $\boldsymbol{V} = \boldsymbol{S}'\boldsymbol{B} + \boldsymbol{E}''$ $(n' \times n')$
4. ciphertext: $(\boldsymbol{C}_1, \boldsymbol{C}_2) = (\boldsymbol{B}', \boldsymbol{V} + \text{Encode}(\mu))$

Encode (transform $\{0,1\}^{B \cdot n' \cdot n'}$ into an $n' \times n'$ matrix):

1. each $B$-bit chunk $k$ is transformed into $k \cdot 2^{D-B} \in \mathbb{Z}_q$ (set the most significant bits)
2. return $n' \times n'$ matrix comprised of these elements

# Decryption

Decryption (two matrices $(\boldsymbol{C}_1, \boldsymbol{C}_2)$):

1. compute $\boldsymbol{M} = \boldsymbol{C}_2 - \boldsymbol{C}_1\boldsymbol{S}$ $\quad (n' \times n')$
2. decode: for each element $c$ of $\boldsymbol{M}$ decode $B$ bits as follows:

$$\lfloor c \cdot 2^{B-D} \rceil \bmod 2^B \quad \left(\text{divided by } \frac{q}{2^B} \text{ and rounded}\right)$$

Correctness:

$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{C}_2 - \boldsymbol{C}_1\boldsymbol{S} = \boldsymbol{V} + \text{Encode}(\mu) - \boldsymbol{B}'\boldsymbol{S} \\
&= \boldsymbol{S}'\boldsymbol{B} + \boldsymbol{E}'' + \text{Encode}(\mu) - (\boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}')\boldsymbol{S} \\
&= \text{Encode}(\mu) + \boldsymbol{S}'(\boldsymbol{A}\boldsymbol{S} + \boldsymbol{E}) + \boldsymbol{E}'' - (\boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}')\boldsymbol{S} \\
&= \text{Encode}(\mu) + \boldsymbol{S}'\boldsymbol{E} + \boldsymbol{E}'' - \boldsymbol{E}'\boldsymbol{S} \\
&= \text{Encode}(\mu) + \boldsymbol{E}^* \quad \text{where } \boldsymbol{E}^* \text{ should be small}
\end{aligned}$$

# Remarks

- failure rate (wrong decryption) for presented parameters: $2^{-138.7}$
- LWE security (in bits): classical 145, quantum 104
- the scheme can be transformed into IND-CCA KEM scheme
  - various transforms exist

# PKE and KEM

**PKE**

- Gen $\rightarrow$ (pk, sk)
  - public key and private key
- $\mathrm{Enc}_{\mathrm{pk}}(m) \rightarrow c$
- $\mathrm{Dec}_{\mathrm{sk}}(c) \rightarrow m$

**KEM**

- Gen $\rightarrow$ (pk, sk)
- $\mathrm{Encaps}_{\mathrm{pk}} \rightarrow (c, k)$
  - ciphertext and key
- $\mathrm{Decaps}_{\mathrm{sk}}(c) \rightarrow k$

---

- both schemes use public key and private key
- PKE encrypts arbitrary message from a plaintext space
  - plaintext space determined by the scheme's instance
  - anything else: padding, mapping into/from plaintext space
  - PKE often used just for symmetric key transport
- KEM is a dedicated scheme for symmetric key agreement

# IND-CCA security of KEM

**Ind-KEM$^{\mathbf{cca}}_{A,\Pi}(t)$**

$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}(1^t)$

$(c, k_0) \leftarrow \mathrm{Encaps}_{\mathrm{pk}}$

$b \in_R \{0,1\}, k_1 \in_R \mathcal{K}$

$b_A \leftarrow A^{\mathrm{Decaps}_{\mathrm{sk}}(\cdot)}(\mathrm{pk}, k_b, c)$

return $b_A \stackrel{?}{=} b$

- PPT attacker $A$
- $A$ has access to pk and Decaps oracle, but cannot ask to decapsulate $c$
- $\mathcal{K}$ – set of keys generated by the KEM scheme
- $A$ has to distinguish if the $c$ contains given key ($k_b$), or this key is a random one
- $\Pr\left[\text{Ind-KEM}^{\mathrm{cca}}_{A,\Pi}(k) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(k)$

# Fujisaki-Okamoto transform (PKE ↦ KEM)

**$\mathbf{Encaps_{pk}}$ :**

$m \in_R \{0,1\}^{256}$

$(k, \mathrm{rnd}) = G(H(\mathrm{pk}), m)$

$c = (u, v) \leftarrow \mathrm{Enc}_{\mathrm{pk}}(m, \mathrm{rnd})$

return $(c, k)$

**$\mathbf{Decaps_{sk}}(c)$ :**

$m' \leftarrow \mathrm{Dec}_{\mathrm{sk}}(c)$

$(k', \mathrm{rnd}') = G(H(\mathrm{pk}), m')$

$c' = (u', v') \leftarrow \mathrm{Enc}_{\mathrm{pk}}(m', \mathrm{rnd}')$

if $c \neq c'$ return $H(z, c)$

return $k'$

- Gen is the same in both schemes
- $G$, $H$ are hash functions (random oracles) with suitable range
- rnd – pseudorandom string, seed for all randomness in encryption; $\mathrm{Enc}_{\mathrm{pk}}(\cdot, \mathrm{rnd})$ is deterministic
- implicit rejection for incorrect $c$ ($z$ is a secret value)

# Exercises

1. *Show that a centered binomial distribution is a probability distribution. Generate a histogram for* $\beta = 8$.
2. *Show that Regev's scheme is malleable. Is the PKE scheme inspired by FrodoKEM also malleable?*
3. *Give a lower bound for* $q$ *in the PKE scheme inspired by FrodoKEM, such that the probability of wrong decryption is 0 (all other parameters remain intact).*