Background: Lattices and the Learning-with-Errors problem
Starting Point: Linear Equations

• Easy to solve a linear system of equations

\[
\begin{bmatrix}
A & s \\
\end{bmatrix}
= \begin{bmatrix}
b \\
\end{bmatrix} \quad (mod \ q)
\]

• Given \( A, b \), find \( s \)
• Solved using Gaussian elimination, Cramer rule, etc.

• [Regev 2005] Hard if we add a little noise

\[
\begin{bmatrix}
A & s \\
\end{bmatrix}
+ \begin{bmatrix}
e \\
\end{bmatrix}
= \begin{bmatrix}
b \\
\end{bmatrix} \quad (mod \ q)
\]

• \( e \) is a noise vector, \( |e| \ll q \)
• Given \( A, b \), find \( s \) and/or \( e \)
Learning with Errors (LWE) [R’05]

- **Parameters:**
  - \(q\) (modulus), \(n\) (dimension), \(m>n\) (# of samples)

- **Secret:** uniformly random vector \(s \in \mathbb{Z}_q^n\)

- **Input:** random matrix \(A \in \mathbb{Z}_q^{m \times n}\), vector \(b \in \mathbb{Z}_q^m\)
  - Computed as \(b = A \times s + e \pmod{q}\)
  - \(e\) chosen from some distribution s.t. \(|e| \ll q\) whp
  - \(b\) is close to the columns space of \(A\)

- **Goal:** discover \(s\)
Learning with Errors (LWE) [R’05]

1. Is it really hard to solve LWE?
   - How hard?
   - For what range of parameters?

2. Is it useful?
   - Can we design cryptosystems with security based on the hardness of LWE

- We’ll do #2 first, then #1
Using LWE in Cryptography
The Decision-LWE Problem

- A more useful variant of LWE:
- Same parameters $q, n, m$
- Input: same and $A$
  - $A$ is still a uniform random matrix in $Z_q^{m \times n}$
  - Either $b = A \times s + e \pmod{q}$, or $b$ is uniform in $Z_q^{m \times n}$
- Goal: distinguish $A \times s + e$ from uniform
  - I.e., given $A, b$, decide if $b$ is “unusually close” to the column space of $A$
Search vs. Decision LWE

- Clearly, if we can solve the search problem then we can also solve the decision problem
  - Try to solve the search problem on \( A, b \)
  - If successful then \( b \) is close to the column space of \( A \), otherwise \( b \) is random

- More interesting: If we can solve decision, then we can also solve the search problem
  - But the complexity grows by a factor of \( q \cdot n \)
  - So this reduction only works for small (polynomial) \( q \)
Assume that we have a distinguisher $D$ that can tell if $b = As + e \ (mod \ q)$ or $b$ is random.

Say for now that $D$ succeeds with probability close to 1.

We construct a solver $S$ that finds $s$.

For every index $i \in \{1..n\}$ and every value $v \in \mathbb{Z}_q$, $S$ will use $D$ to determine if $s_i = v$. 
Reducing Search to Decision LWE

Given $A$ and $b = As + e$, test if $s_i = \nu$:

- Choose $r = (r_1, r_2, ..., r_m) \in \mathbb{Z}_q^m$ uniformly at random
- Add $r$ to the $i$'th column of $A$, this gives a matrix $A'$
  - $A'$ is uniformly random because $A$ is
  $$A' = A + \begin{bmatrix} 0 & \ldots & r_1 \ldots & 0 \\ 0 & \ldots & r_2 \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & r_m \ldots & 0 \end{bmatrix}$$
- Note that $A's = As + s_i \cdot r$
Reducing Search to Decision LWE

**Given** $A$ and $b = As + e$, test if $s_i = \nu$:

- Choose $r = (r_1, r_2, \ldots, r_m) \in \mathbb{Z}_q^m$ uniformly at random
- Add $r$ to the $i$’th column of $A$, this gives a matrix $A'$
  - $A'$ is uniformly random because $A$ is
- Add $\nu \cdot r$ to $b$, this gives the vector $b'$

\[
\begin{align*}
b' &= b + \nu \cdot (r_1, r_2, \ldots, r_m)
\end{align*}
\]
Reducing Search to Decision LWE

Given $A$ and $b = As + e$, test if $s_i = v$:

- Choose $r = (r_1, r_2, ..., r_m) \in \mathbb{Z}_q^m$ uniformly at random.
- Add $r$ to the $i$’th column of $A$, this gives a matrix $A'$
  - $A'$ is uniformly random because $A$ is.
- Add $v \cdot r$ to $b$, this gives the vector $b'$

$$b' = b + v \cdot r$$
$$= As + e + v \cdot r = (A's - s_i \cdot r) + e + v \cdot r$$
$$= A's + e + (v - s_i) \cdot r$$
Reducing Search to Decision LWE

Given $A$ and $b = As + e$, test if $s_i = v$:

- Choose random $r$, compute $A', b'$
  - Such that $b' = A's + e + (v - s_i) \cdot r$
  - If $s_i = v$ then we get $b' = A's + e$
  - Otherwise $b'$ is uniform (because $r$ is uniform)
- Use the distinguisher $D(A', b')$ to tell which case it is
  - This will tell us if $s_i = v$
Reducing Search to Decision LWE

- The reduction assumes that D is always right
  - Can be extended to distinguisher D with polynomially small advantage

- The reduction works in time linear in $n \cdot q$
  - Can be refined to work in time $n \cdot \sum p_i$
  - The $p_i$’s are the prime factors of $q$
  - So the reduction can be efficient even for large $q$, as long as it is smooth
A Useful Variant of LWE

- Instead of choosing the secret $s$ uniformly, choose it from the same distribution as $e$
  - So $s$ is small

Theorem [ACPS’09]: Uniform-secret LWE is equivalent to small-secret LWE

- Solving one $\leftrightarrow$ solving the other
Uniform- vs. Small-secret LWE

Easy direction: If we can solve uniform-secret LWE then we can solve small-secret LWE

- We are given $A$ and $b = As + e$
  - $s$ is a small secret
- Choose a uniform random $r$
- Set $b' = Ar + b = A(r + s) + e \pmod{q}$
  - $s' = r + s$ is uniform (because $r$ is uniform)
- $A, b' = As' + e$ is instance of uniform-secret LWE
- Solving it, we get $s'$ and can compute $s = s' - r$
Uniform- vs. Small-secret LWE

Hard direction: If we can solve small-secret LWE then we can solve uniform-secret LWE

• But the parameter \( m \) changes
• For solving \( m \times n \) uniform-secret LWE, we would need to solve \( m' \times n \) small-secret LWE with \( m' = m - n \)

• We are given \( A \) and \( b = As + e \)
  • \( s \) is a uniform secret

• Find \( n \) linearly-independent rows of \( A \)
  • Such rows exist with high probability
  • Assume that these are the first \( n \) rows
Uniform- vs. Small-secret LWE

- Set \( A' = -A_2A_1^{-1} \) and \( b' = b_2 + A'b_1 \)
  - \( b' = (A_2s + e_2) + A'(A_1s + e_1) \)
    \[ = A_2s + A'A_1s + A'e_1 + e_2 = A'e_1 + e_2 \]
    - Because \( A'A_1 = -A_2A_1^{-1}A_1 = -A_2 \)
  - So \((A', b' = A'e_1 + e_2)\) is instance of short-secret LWE
    - The secret is \( e_1 \), drawn from the error distribution

- Solving it we get \( e_1 \)
  - Then compute \( s = A_1^{-1}(b_1 - e_1) \)
Regev’s Cryptosystem [R’05]

Secret key: vector \( s' \)
Public key: Matrix \( A' \), vector \( b = A's' + e \)

- Denote \( A = (b|A') \)
- If decision-LWE is hard then \( A \) is pseudorandom
- Denote \( s = (1, -s') \), then \( As = b - A's' = e \)

Encrypt \(_A(\sigma \in \{0,1\})\)

- Choose a random small vector \( r \in \{0,1\}^m \)
- Output the ciphertext \( c = rA + \frac{q}{2} \cdot (\sigma, 0, ..., 0) \in Z_q^n \)

Decrypt \(_s(c)\)

- Compute the inner product \( y = \langle c, s \rangle \) (mod q)
- Output 0 if \(|y| < q/4\), else output 1
Regev’s Cryptosystem [R’05]

- **Correctness:**
  
  \[ y = \langle c, s \rangle = \left\langle \left( rA + \frac{q}{2} \sigma \right), s \right\rangle = rAs + \frac{q}{2} \langle (\sigma0 \ldots 0), (1, -s') \rangle = \langle r, e \rangle + \frac{q}{2} \cdot \sigma \]

  \[ |\langle r, e \rangle| < \frac{q}{4} \text{ (since } r \text{ is 0-1 vector and } |e|_{\infty} < \frac{q}{4} \text{)} \]

  \[ \Rightarrow \text{If } \sigma = 0 \text{ then } |y| < \frac{q}{4}, \text{ if } \sigma = 1 \text{ then } |y| > \frac{q}{4} \]

- **Security:**
  
  - Recall that A is pseudo-random
  
  - We show that if A was random then \( c \) was statistically close to uniform, regardless of \( \sigma \)
Regev’s Cryptosystem [R’05]

The Leftover Hash Lemma [HILL’99] implies the following corollary:

- If \( m > 3n \log q \) then the two distributions
  
  \[
  < (A, rA) : A \in_U Z_q^{m \times n}, r \in_U \{0,1\}^m > \\
  < (A, u) : A \in_U Z_q^{m \times n}, u \in_U Z_q^n >
  \]

  are statistically close (upto \( q^{-\Omega(m)} \))

\[\Rightarrow\] For a random \( A \), \( rA \) is close to uniform

- Even conditioned on \( A \)
- And therefore so is \( rA + \frac{q}{2} \hat{\sigma} \)

\[\Rightarrow\] If \( A \) is pseudorandom, so is \( rA + \frac{q}{2} \hat{\sigma} \)
A Useful Variant of the Cryptosystem

Encrypt: \( c = 2rA + \tilde{\sigma} \)
- instead of \( c = rA + \frac{q}{2} \tilde{\sigma} \) from before
- Plaintext encoded in the LSB rather than MSB

Decrypt: \( y = \langle c, s \rangle \pmod{q} \), then \( \sigma = y \mod 2 \)
- \( y = 2\langle r, e \rangle + \sigma \pmod{q} \)
- \(|2\langle r, e \rangle| < q/2\), so no mod-\( q \) reduction
  \[ \Rightarrow y \mod 2 = \sigma \]
The Hardness of LWE
A lattice is just an additive subgroup of $\mathbb{R}^n$. 
Lattices

Lattice of rank $n$ = set of all integer linear combinations of $n$ linearly independent basis vectors.
Lattices

- A Lattice has infinitely many bases
  - They are related by unimodular matrices, $B' = BU$
  - $U$ is an integer matrix with $\det(U) = \pm 1$
  - All bases have the same determinant (upto sign)
  - This quantity is the determinant of the lattice

- Given any set of vectors that span the lattice, can compute a canonical basis
  - Hermite normal form (HNF)
Lattices

• A “good basis” has all small vectors
  • “close to orthogonal” to each other
  • Typically the HNF is a “bad” basis

• Minkowsky’s theorems:
  A rank-\( n \) lattice with determinant \( d \) has
  • A non-zero vector of length \( |v| \leq \sqrt{n} \cdot d^{1/n} \)
  • \( n \) linearly independent \( v_i \)’s s.t. \( \prod_{i=1}^{n} |v_i| \leq n^{n/2} \cdot d \)
    • Also a basis of vectors of similar sizes

• Lattice reduction: Given a “bad” basis, find a “good” one for the same lattice
Lattices and Hard Problems

Given some basis of $L$, may be hard to find good basis of $L$. Hard to solve the (approx) shortest/closest vector problems.
Hard Problems

Given a basis $B$ for a lattice $L(B)$:

- **Shortest-Vector Problem (SVP)**
  - Find the shortest nonzero vector in $L(B)$
  - Or maybe just compute the size of such vector ($\lambda_1(L)$)

- **Shortest Independent-Set Problem (SIVP)**
  - Find $n$ linearly independent $v_1, \ldots, v_n$ minimize $\max_i |v_i|$
  - Or maybe just the quantity $\lambda_n(L) = \max_{i=1}^n |v_i|$

- Also approximation versions
  - Find $v$ such that $|v| \leq \gamma \cdot$ shortest
  - Find $v_i$’s such that $\max_i |v_i| \leq \gamma \cdot$ smallest-possible
Hard Problems: What’s Known?

- The [LLL’82] algorithm and its variants can approximate SVP upto $\gamma = 2^{O(n)}$
- NP-hard to approx. SVP upto $\gamma = 2^{\log^{1-\epsilon} n}$
  - $\gamma = \omega(1)$ but $\gamma < n^\epsilon$ for any $\epsilon$
- Roughly: approximate upto $2^{n/k}$ takes time $2^{O(k)}$
  - Practically we can perhaps approximate SVP upto $\gamma = 2^{n/100}$ but not upto $\gamma = 2^{n/200}$
  - At least for moderate $n$’s (say $n < 500$)
- Similar for SIVP
LWE and Lattices

- Consider the matrix $A = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}$

- The column space mod-$q$ is a rank-$m$ lattice, spanned by the columns of $B = \begin{bmatrix} \ldots \end{bmatrix}$
  - A discrete additive subgroup of $\mathbb{R}^m$
  - Can compute its HNF basis

- $b = As + e \ (mod\ q)$ is close to this lattice
  - $v = b - e$ is in the lattice, at distance $|e|$ from $b$
    - We have a bound $\beta \ll q$ on $|e|$ whp (say $\beta = \sqrt{q}$)
  - If we find $v$, we can solve for $s$
Bounded Distance Decoding (BDD)

- Input: a basis $B$, another point $x$, a bound $\beta$
- Goal: find $v \in L(B)$ such that $|x - v| \leq \beta$

- Solving BDD $\Rightarrow$ Solving LWE

*Thm [Babai’86,GPV’08]:*
- Solving SIVP $\Rightarrow$ Solving BDD
- Given a basis for $L$ with $\max_i |v_i| = \alpha$, can solve BDD upto distance $\beta \approx \alpha \cdot \text{poly}(n)$
Thm [Reg’05, Pei’09]:

- Solving LWE $\Rightarrow$ Solving SIVP, SVP
- LWE-solver with error-bound $\beta < q/O(\sqrt{n})$ implies quantum approximation of SIVP upto a factor poly$(n)$
- Or a classical algorithm for approximating $\lambda_1(L)$ upto a factor poly$(n)$
Summary

- Learning with Errors: \( b = As + e \)
- This is a hard problem
  - For some parameters, can be shown to be as hard as some well-known lattice problems
  - Even for other settings, we don’t know how to solve it
- Only known attacks use lattice reduction
  - These only work when \( q/|e| = \exp(n) \)
- LWE is useful for cryptography
  - For example for public-key encryption
  - Decryption formula \( \langle s, c \rangle \mod q \mod 2 \)