CS 6501 Week 11: Lattice-Based Cryptography

So far in this course: Foundations of modern cryptography, pairing-based cryptography, zero-knowledge proof systems and cryptographic protocols

Final major topic in this course: post-quantum cryptography and the next generation of cryptography

We will not have time to cover quantum computing in this course. We will just state the implications:

Grover's algorithm: Given black-box access to a function $f : [N] \to \{0,1\}$, Grover's algorithm finds an $x \in [N]$ such that $f(x) = 1$ by making $O(\sqrt{N})$ queries to f.

"Searching an unsorted database of size N in time $O(\sqrt{N})$."

- Classically: Searching an unstructured database of size N requires time $\Omega(N)$ — cannot do better than a linear scan.
- Quantum: Grover's algorithm is tight for unstructured search. Any quantum algorithm for the unstructured search problem requires making $\Omega(\sqrt{N})$ queries (to the function / database).
  - $\Rightarrow$ Quantum computes provide a quadratic speedup for unstructured search, and more broadly, function inversion.

Implications in cryptography: Consider a one-way function over a 128-bit domain. The task of inverting a one-way function is to find $x \in \{0,1\}^{128}$ such that $f(x) = y$ for some fixed target value f. Exhaustive search would take time $\approx 2^{128}$ on a classical computer, but using Grover's algorithm, can perform in time $\approx \sqrt{2^{128}} = 2^{64}$.

$\Rightarrow$ For symmetric cryptography, need to double key-sizes to maintain same level of security (unless there are new quantum attacks on the underlying construction itself.

$\Rightarrow$ Use AES-256 instead of AES-128 (not a significant change!)

Similar algorithm can be applied to obtain a quantum collision-finding algorithm that runs in time $\sqrt[3]{N}$ where N is the size of the domain (compare to $\sqrt{N}$ for the best classic algorithm)

$\hookrightarrow$ Instead of using SHA-256, use SHA-384 (not a significant change)

$\hookrightarrow$ The quantum algorithm require a large amount of space, so not clear that this is a significant threat, but even if it were, using hash functions with 384-bits of output suffices for security

Main takeaway: Symmetric cryptography mostly unaffected by quantum computers ~ generally just require a modest increase in key size

$\hookrightarrow$ e.g., symmetric encryption, MACs, authenticated encryption

Story more complicated for public-key primitives:

- Simon's algorithm and Shor's algorithm provide polynomial-time algorithms for solving discrete log (in any group with an efficiently-computable group operation) and for factoring
- Both algorithms rely on period finding (and more broadly, on solving the hidden subgroup problem)

Intuition for discrete log algorith (as a period finding problem):

- Let $(g, h = g^{\alpha})$ be the discrete log instance in a group of prime order p
- Let $f : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{G}$ be the function

$$f(x,y) = g^x h^{-y}$$

- By construction,

$$f(x+\alpha, y+1) = g^{x+\alpha} h^{-y-1} = g^x h^{-y} g^{\alpha} h^{-1} = g^x h^{-y} = f(x,y)$$

- Thus, the element $(\alpha, -1)$ is the period of f, so using Shor's algorithm, we can efficient compute $(\alpha, -1)$ from (g, h), which yields the discrete log of h

Thus, if large scale quantum computers come online, we will need new cryptographic assumptions for our public-key primitives

↳ All the algebraic assumptions we have considered so far (e.g., discrete log, factoring, pairings) are broken

How realistic is this threat? - Lots of progress in building quantum computers recently by both academia and industry (e.g., see initiatives by Google, IBM, etc.)

- To run Shor's algorithm to factor a 2048-bit RSA modulus, estimated to need a quantum computer with $\approx$ 10000 logical qubits (analog of a bit in classical computers)
  - ↳ With quantum error correction, this requires $\geq$ 10 million physical qubits to realize
  - ↳ Today: machines with 10s of physical qubits, so still very far from being able to run Shor's algorithm
- Optimistic estimate: At least 20-30 years away (and some say never...)

Should we be concerned? Quantum computers would break existing key-exchange and signature schemes

- Signatures: Future adversaries would be able to forge signatures under today's public keys, so if quantum computers come online, we can switch to and only use post-quantum schemes
- Key-Exchange: Future adversaries can break confidentiality of today's messages (i.e., we lose forward secrecy) — this is problematic in many scenarios (e.g., businesses want trade secrets to remain hidden for 50 years)

Reasons to study post-quantum cryptography:

1. Protect confidentiality of today's computations against potential future threat
2. Standards take a long time to develop and deploy, so should start now
   - ↳ NIST has initiated a multi-year initiative to develop and standardize post-quantum key-exchange and signatures (currently in 2nd year of 6-year initiative)
   - ↳ Google recently piloted an experiment involving post-quantum key exchange in Chrome (using a "best of both worlds" approach where key derived from mix of classic key exchange and post-quantum key exchange)
3. New kinds of mathematical structures and assumptions — opportunity to build cryptography up from scratch again!

Candidates for post-quantum hardness: many classes of assumptions, many different tradeoffs, will survey several below:

- Hash-based cryptography: - Use hash functions (symmetric primitives)
  - Suffices for signatures, but not for key exchange (black box separations)
  - Assumption seems very safe (not based on algebraic / structured hardness assumptions)
  - Signatures built from hash functions are very large (e.g., SPHINCS signatures are 40 KB long)
    ↳ Could be good choice where large signatures are acceptable (e.g., signing software updates)

- Isogeny-based cryptography: - More recent class of cryptographic assumptions based on hard problems related to computing mappings between elliptic curves
  - Gives a simple key-exchange protocol that is analogous to Diffie-Hellman and has compact communication (e.g., a few hundred bytes)
  - Signatures also possible, but longer compared to Schnorr / ECDSA, shorter compared to hash-based and lattices [Open: Schnorr-style signatures from isogenies?]
  - Relatively new type of hardness assumption – needs more cryptanalysis
  - Has interesting algebraic structure (can be viewed as computing a hard group action) and provides promising avenues for developing new types of cryptographic primitives [lots of interesting research problems!]

- Code-based cryptography: - Based on hard problems from coding theory (e.g., hardness in decoding a random linear code)
  - Dates back to the late 1970s (e.g., McEliece family of cryptographic schemes)
  - Many variants (e.g., using codes with additional algebraic structure are broken, but original candidate by McEliesce remains a plausible candidate
  - Schemes have large parameter (key-sizes) - needed to resist best-known attacks

- Multivariate Cryptography: - Based on conjectured hardness of solving systems of multivariate polynomials over finite fields
  - Many schemes based on these types of assumptions have been broken, and to date, there has been (relatively) limited study on these assumptions
  - Typically schemes have large parameter sizes, so there is no clear advantage compared to many of the other leading contenders

Our focus: lattice-based cryptography

Before defining lattices, a few motivating reasons to study lattices (beyond its conjectured post-quantum resilience)

- Hardness assumptions in lattice-based cryptography can be based on worst-case hardness (rather than the more traditional notion of average-case hardness that we have encountered throughout this course so far)
- Worst-case problems over lattices (as well as the specific computational problems we work with) have been extensively studied (so we have good confidence in their security)
- Lattices have a lot of useful algebraic structure, which has enabled many powerful cryptographic applications that we did not have before (most notably: fully homomorphic encryption — enables computing on encrypted data)
  ↳ Breakthrough result of FHE in 2009 has led to a dramatic expansion to the landscape of cryptography and demonstrated power + potential of lattice-based cryptography

Definition. An n-dimensional lattice $\mathcal{L}$ is a "discrete additive subspace" of $\mathbb{R}^n$:

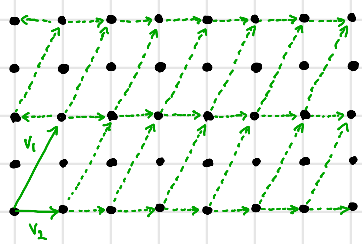
1. Discrete: every $x \in \mathbb{R}^n$ has a neighborhood in $\mathbb{R}^n$ where it is the only point
2. Additive subspace: $0^n \in \mathcal{L}$ and for all $x, y \in \mathcal{L}$, $-x \in \mathcal{L}$ and $x+y \in \mathcal{L}$

Example: the integer lattice $\mathbb{Z}^n$, the "q-ary" lattice $q\mathbb{Z}^n$ (i.e., the set of vectors where each entry is an integer multiple of q)

While most (non-trivial) lattices are infinite, they are finitely-generated by taking integer linear combinations of a finite collection of basis vectors $B = \{b_1, \ldots, b_k\}$:

$$\mathcal{L} = \mathcal{L}(B) = B \cdot \mathbb{Z}^k = \left\{ \sum_{i \in [k]} \alpha_i b_i : \alpha_i \in \mathbb{Z} \text{ for all } i \in [k] \right\}$$

Example over $\mathbb{R}^2$:



Computational problems:

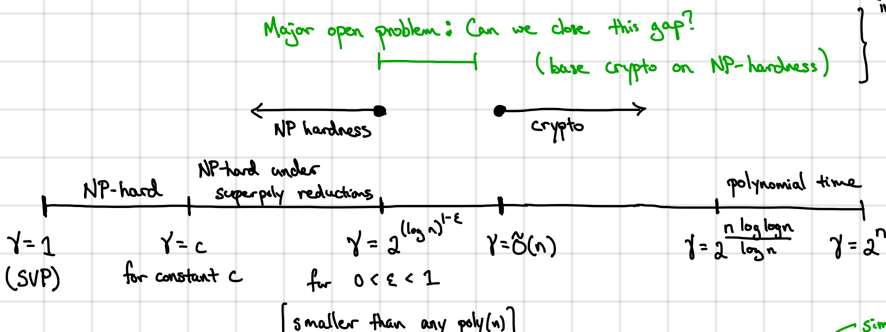
- Shortest vector problem (SVP): Given a basis B for a lattice $\mathcal{L} = \mathcal{L}(B)$, find a shortest non-zero vector $v \in \mathcal{L}$
- Approximate SVP (SVP_γ): Given a basis B for a lattice $\mathcal{L} = \mathcal{L}(B)$, find a non-zero vector $v \in \mathcal{L}$ such that $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L})$, where $\lambda_1(\mathcal{L})$ denotes the norm of the shortest non-zero vector in $\mathcal{L}$

  for simplicity, we will use the ℓ_2-norm

  approx factor typically function of lattice dimension n
- Decisional approximate SVP ($GapSVP_{d,\gamma}$): Given a basis B for a lattice $\mathcal{L} = \mathcal{L}(B)$ where either $\lambda_1(\mathcal{L}) \leq d$ or $\lambda_1(\mathcal{L}) \geq \gamma \cdot d$, decide which is the case

Many other lattice problems, but these should provide a flavor for what lattice problems look like

Hardness results: Many lattice problems are known to be NP-hard (possibly under randomized reductions)



intermediate results:
$\gamma = \sqrt{n / \log n}$: NP $\cap$ coAM
$\gamma = \sqrt{n}$: NP $\cap$ coNP

Hardness of GapSVP for different approximation factors γ [under the ℓ_2-norm]

similar results under the ℓ_∞ norm (since $\|v\|_\infty \leq \|v\|_2 \leq \sqrt{n}\|v\|_\infty$)

For cryptographic constructions, it is oftentimes more convenient to use average-case problems (which admit reductions from GapSVP)

- Specifically, we rely on the short integer solutions (SIS) or the learning with errors (LWE) problems, which are average-case problems
- Both the SIS and the LWE problems can be based on the hardness of the GapSVP problem (e.g., an adversary that solves SIS or LWE can be used to solve GapSVP in the worst-case)

Short Integer Solutions (SIS): The SIS problem is defined with respect to lattice parameters n, m, q and a norm bound β. The $SIS_{n,m,q,\beta}$ problem says that for $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, no efficient adversary can find a non-zero vector $x \in \mathbb{Z}^m$ where

$$Ax = 0 \in \mathbb{Z}_q^n \text{ and } \|x\| \leq \beta$$

In lattice-based cryptography, the lattice dimension n will be the primary security parameter.

Notes: - The norm bound β should satisfy $\beta \le q$. Otherwise, a trivial solution is to set $x = (q, 0, 0, \dots, 0)^T$.

- We need to choose m, β to be large enough so that a solution does exist.

  ↳ When $m = \Omega(n \log q)$ and $\beta \ge 1$, a solution always exists. In particular, when $m \ge \lceil n \log q \rceil$, there always exists $x \in \{-1,0,1\}^m$ such that $Ax = 0$:

  recall that we are using the ℓ_∞ norm (unless otherwise noted)

  - There are $2^m \ge 2^{n \log q} = q^n$ vectors $y \in \{0,1\}^m$
  - Since $Ay \in \mathbb{Z}_q^n$, there are at <u>most</u> q^n possible outputs of Ay

  By a counting argument, there exist $y_1 \ne y_2 \in \{0,1\}^m$ such that $Ay_1 = Ay_2$

  - Thus, if we set $x = y_1 - y_2 \in \{-1,0,1\}^m$, then $Ax = A(y_1 - y_2) = Ay_1 - Ay_2 = 0 \in \mathbb{Z}_q^n$

In fact, the above argument shows that SIS gives a <u>collision-resistant hash function</u> (CRHF).

<u>Definition</u>. A keyed hash family $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is collision-resistant if the following properties hold:

- <u>Compressing</u>: $|\mathcal{Y}| < |\mathcal{X}|$
- <u>Collision-Resistant</u>: For all efficient adversaries $\mathcal{A}$:

$$\Pr[k \xleftarrow{R} \mathcal{K} \,;\, (x, x') \leftarrow \mathcal{A}(1^\lambda, k) : H(k,x) = H(k,x') \text{ and } x \ne x'] = \text{negl}(\lambda).$$

We can directly appeal to SIS to obtain a CRHF:

$$H : \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \longrightarrow \mathbb{Z}_q^n$$

where we set $m > \lceil n \log q \rceil$. In this case, domain has size $2^m > 2^{n \log q} = q^n$, which is the size of the output space. Collision resistance follows assuming $\text{SIS}_{n,m,q,\beta}$ for any $\beta \ge \sqrt{\lceil n \log q \rceil}$

The SIS hash function supports efficient local updates:

Suppose you have a public hash $h = H(x)$ of a bit-string $x \in \{0,1\}^m$. Later, you want to update $x \mapsto x'$ where x and x' only differ on a <u>few</u> indices (e.g., updating an entry in an address book). For instance, suppose x and x' differ only on the first bit (e.g., $x_1 = 0$ and $x_1' = 1$). Then observe the following

$$h = H(k,x) = A \cdot x$$

$$= \begin{pmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_m \\ | & | & & | \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \sum_{i \in [m]} x_i a_i = \sum_{i=2}^{m} x_i a_i \qquad \text{since } x_1 = 0$$

$$h' = H(k, x') = A \cdot x'$$

$$= \sum_{i \in [m]} x_i' a_i = x_1' a_1 + \sum_{i=2}^{m} x_i' a_i = a_1 + \sum_{i=2}^{m} x_i' a_i = a_1 + h \qquad \text{since } x_i' = x_i \text{ for all } i \ge 2$$

Thus, we can easily update h to h' by just adding to it the first column of A without (re)computing the <u>full</u> hash function.

<u>Variant</u>: Inhomogeneous SIS. Given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $u \xleftarrow{R} \mathbb{Z}_q^n$, find a short $x \in \mathbb{Z}_q^m$ (i.e., $\|x\| \le \beta$) such that $Ax = u \in \mathbb{Z}_q^n$.

<u>Implication</u>: Can be used to get an OWF. Take $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and define the function $f_A : \{0,1\}^m \to \mathbb{Z}_q^n$ where $f_A(x) := Ax \in \mathbb{Z}_q^n$.

Not quite immediate. OWF security: sample $x \xleftarrow{R} \{0,1\}^m$, compute $y = f_A(x)$ and give (A, y) to the adversary.

Inhomogeneous SIS: sample $y \xleftarrow{R} \mathbb{Z}_q^n$ and give (A, y) to the adversary.

Are these distributions the same?

[When $m = \Omega(n \log q)$, these two distributions are statistically indistinguishable]

<u>Definition</u>. A keyed hash function $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is pairwise independent if for all $x_1 \ne x_2 \in \mathcal{X}$ and $y_1, y_2 \in \mathcal{Y}$,

$$\Pr[k \xleftarrow{R} \mathcal{K} : H(k, x_1) = y_1 \text{ and } H(k, x_2) = y_2] = \frac{1}{|\mathcal{Y}|^2}.$$

Definition. Let Ω be a finite set and X be a random variable over Ω. Then, the guessing probability $\gamma(X)$ is defined as

$$\gamma(X) = \max_{x \in \Omega} \Pr[X = x]$$

[The probability of the most likely value of X]

The min-entropy of X, denoted $H_\infty(X)$ is defined to be

$$H_\infty(X) = -\log \max_{x \in \Omega} \Pr[X = x]$$

[Number of bits of randomness in X]

Leftover Hash Lemma (LHL): Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a pairwise-independent hash family. Let X be a random variable over $\mathcal{X}$ with guessing probability γ. Then, for $k \xleftarrow{R} \mathcal{K}$,

$$\Delta[(k, H(k, X)), (k, Y)] \leq \frac{1}{2}\sqrt{\gamma |\mathcal{Y}|}$$

where Y is the uniform distribution over $\mathcal{Y}$.

In words: pairwise independent hash functions are good randomness extractors.

Example: Suppose we use a group-based PRF, and we want to extract a 128-bit AES key. Suppose we have a pairwise-independent hash function $H : \mathcal{K} \times \mathbb{G} \to \{0,1\}^{128}$. If we have a group element σ with 256 bits of min-entropy, then $\gamma = 2^{-256}$. In this case, $H(\sigma)$ is δ-close to uniform where $\delta = \frac{1}{2}\sqrt{2^{-256} \cdot 2^{128}} \leq 2^{-64}$.

And now back to Inhomogeneous SIS... the family $H : \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \setminus \{0^m\} \to \mathbb{Z}_q^n$ is pairwise independent whenever q is prime.

Take any $x_1 \neq x_2 \in \{0,1\}^m \setminus \{0^m\}$ and $y_1, y_2 \in \mathbb{Z}_q^n$. Suppose $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$. Then,

$$\begin{aligned}\Pr[Ax_1 = y_1 \text{ and } Ax_2 = y_2] &= \Pr[Ax_1 = y_1] \cdot \Pr[Ax_2 = y_2 \mid Ax_1 = y_1] \\ &= \Pr[Ax_1 = y_1] \cdot \Pr[A(x_2 - x_1) = y_2 - y_1]\end{aligned}$$

Since $x_1 \neq 0$, Ax_1 is taking a subset-sum of the columns of A. Since A is uniformly random, $\Pr[Ax_1 = y_1] = \frac{1}{q^n}$ (can see this by sampling all but one column of A, corresponding to an entry in x that is set to 1 — probability that this column satisfies the relation is $\frac{1}{q^n}$). Likewise for $\Pr[A(x_2 - x_1) = y_2 - y_1]$.

Consider the distributions in the inhomogeneous SIS problem and the OWF security game:

OWF security: sample $x \xleftarrow{R} \{0,1\}^m$, compute $y = f_A(x)$ and give (A, y) to the adversary.

Inhomogeneous SIS: sample $y \xleftarrow{R} \mathbb{Z}_q^n$ and give (A, y) to the adversary.

From above, $H(A, x) = f_A(x)$ is a pairwise-independent hash function so sampling $x \xleftarrow{R} \{0,1\}^m$ and computing $f_A(x) = Ax$ yields a value that is statistically close to uniform over $\mathbb{Z}_q^n$. [Statistical distance is $\frac{1}{2}\sqrt{2^{-m} \cdot q^n} = \frac{1}{2}\sqrt{q^{-3n} \cdot q^n} = \frac{1}{2} q^{-n} = \mathrm{negl}(n)$]

Here, we will take $m \geq 3n \log q$. [Smaller values also suffice for argument.]

The LHL will be a very useful tool in lattice-based cryptography (and more generally in cryptography!)

SIS as a lattice problem: given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, find non-zero $x \in \mathbb{Z}_q^m$ such that $Ax = 0 \in \mathbb{Z}_q^n$ and $\|x\| \leq \beta$.

↳ Can be viewed as an average-case version of finding short vectors in a "q-ary" lattice:

$$\mathcal{L}^\perp(A) = \{ z \in \mathbb{Z}^m : Az = 0 \pmod{q} \}$$

Notice that by construction, $q\mathbb{Z}^m \subseteq \mathcal{L}^\perp(A)$

"q-ary" lattice (e.g., vectors where all entries are integer multiples of q)

Inhomogeneous SIS: given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $y \xleftarrow{R} \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ such that $Ax = y \in \mathbb{Z}_q^n$ and $\|x\| \leq \beta$

↳ This is problem of finding short vectors in lattice $\mathcal{L}_y^\perp(A) = c + \mathcal{L}^\perp(A)$ where $c \in \mathbb{Z}_q^m$ is an arbitrary vector where $Ac = y$

Hardness of SIS: Ajtai first showed (in 1996) that average-case hardness of SIS can be based on worst-case hardness of certain lattice problems $\Longrightarrow$ long sequence of works understanding and improving the worst-case to average-case reductions

Typical statement: Let n be the lattice dimension. For any $m = \text{poly}(n)$, norm bound $\beta > 0$, and sufficiently large $q \geq \beta \cdot \text{poly}(n)$, Then, the $\text{SIS}_{n,m,q,\beta}$ problem is at least as hard as solving GapSVP_γ on an arbitrary n-dimensional lattice for $\gamma = \beta \cdot \text{poly}(n)$.

↳ i.e., solving SIS is as hard as approximating GapSVP in the worst case!