CSC 2419: Lattice-based Cryptography **Fall 2025**

Lecture 1: Short Integer Solutions and Collision-Resistant Hashing

Instructor: Akshayaram Srinivasan *Scribe: Sky Li*

Date: 2025-09-08

1.1 What is Cryptography?

Traditionally, cryptography focused on ensuring private communication between parties. However, the modern goal is broader. Modern cryptography can be seen as designing systems that protect information and computation from any adversarial attack.

Formally, to define a cryptographic system, we must specify the following:

1. What tasks does the system need to perform, and what information or computation does it need to protect?
2. What does the system need to protect **against**?

For (2), we generally will consider attackers to be any **polynomial-time** adversary. Specifically, we only consider computationally-bounded adversaries that can run adversarial attacks in polynomial-time with respect to a given **security parameter** for the system (which may depend on the system).

1.1.1 Security Reductions

How does one formally prove that a system is safe against **all** poly-time adversaries? One approach is through a **security reduction**:

Assume the existence of a poly-time attacker for the system. The reduction demonstrates that if such an attacker exists, it can be utilized to solve a (believed-to-be-)**hard** mathematical problem, such as factoring. Since we believe the underlying mathematical problem is **hard** to solve, this implies that breaking the security of the cryptographic system must be equivalently hard. Notably, the mathematical problem should have survived extensive cryptanalytic effort to have sufficient confidence that no polynomial-time solvers exist. (Note: if we could prove this **unconditionally**, we would have established that $P \neq NP$).

1.1.2 Choosing Hard Problems

Because the security of cryptographic systems inherently relies on the **difficulty** of a given mathematical problem, the specific mathematical problem is often of interest for cryptographers. Historically, cryptographic hardness relied on the hardness of problems like **factorization** and **discrete logarithm**. However, [S94] showed that, on a *quantum* computer, factorization and discrete log can be solved in polynomial time. This motivated the development of cryptographic systems based on different underlying problems. We focus on **lattice**-based problems; intuitively, these problems are believed to have quantum resistance than traditional problems because they are **geometric** in nature.

1.2 Lattice-Based Cryptography

Informally, a **lattice** is a set of points in m-dimensional space that has a **periodic** structure. We formally define a lattice as follows:

Definition 1.1 (Lattices) *Let $B = b_1, \ldots, b_n \in \mathbb{R}^m$ such that $B = \{b_1, \ldots, b_n\}$ is linearly independent. Then the **lattice** $\mathcal{L}(B)$ is defined as:*

$$\mathcal{L}(B) := \left\{ \sum_{i=1}^{n} x_i b_i \Big| x_i \in \mathbb{Z} \right\}$$

*We call B the **basis** of the lattice, n the **rank** of the lattice, and m the **dimension** of the lattice. If $n = m$, then we call $\mathcal{L}(B)$ a **full rank** lattice.*

In other words, a lattice is the set of all arbitrary **integer** linear combinations of a linearly independent basis set.

1.2.1 Examples

1. If $B = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$, then $\mathcal{L}(B) = \mathbb{Z}^2$.
2. If $B = \left\{ \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$, then $\mathcal{L}(B) = \mathbb{Z}^2$ also.

Note that a single lattice can have multiple bases!

1.3 Shortest Vector Problem (SVP)

We now define the Shortest Vector Problem (SVP) on lattices—this will serve as a hard problem for the purposes of constructing cryptographic systems.

We first define the notion of a **shortest vector** in a lattice:

Definition 1.2 (Shortest Vector) *The length of the shortest vector of a lattice L, denoted $\lambda_1(L)$, is defined as the length of the shortest[1] **non-zero** vector in L. Formally,*

$$\lambda_1(L) = \min_{v \in L \setminus \{\mathbf{0}\}} ||v||_2$$

We are specifically interested in the **decision** problem that involves finding the length of the shortest vector:

Definition 1.3 (Shortest Vector Decision Problem) *The Shortest Vector Decision Problem (SVP) is defined as follows:*

- *Given: A lattice $\mathcal{L}(B)$ with basis B, $r \in \mathbb{R}^+$.*

[1] Shortness can be defined using any linear norm; for the purposes of these notes we will default to the ℓ_2 norm.

- *Decide: Is* $\lambda_1(\mathcal{L}(B)) \leq r$*?*

We observe that SVP is clearly in NP: given a candidate shortest vector v, we must check that (1) $v \in \mathcal{L}(B)$, and that (2) $||v||_2 \leq r$. (1) can be checked in polynomial time by solving a system of linear equations given B and v and checking whether it has an integral solution, and (2) can be checked immediately. It is additionally known that SVP is $NP-$hard, if randomized reductions are allowed [A98].

We often call this version of SVP **Exact-SVP**, to distinguish it from a commonly-used relaxation:

Definition 1.4 (Gap-SVP) *The approximate decision version of SVP (Gap-SVP) is defined as follows:*

- *Given: A lattice* $\mathcal{L}(B)$ *with basis* B, $\gamma \in \mathbb{R}^+, \alpha \in \mathbb{N}$. *Additionally, it is promised that either* $\lambda_1(\mathcal{L}(B)) \leq \gamma$ *or* $\lambda_1(\mathcal{L}(B)) \geq \alpha\gamma$.
- *Decide: Is* $\lambda_1(\mathcal{L}(B)) \leq \gamma$ *("Yes") or* $\lambda_1(\mathcal{L}(B)) \geq \alpha\gamma$ *("No")?*

Gap-SVP can be considered parametrized by α*—in these cases, we denote the problem* α*-Gap-SVP.*

We observe that when $\alpha = 1$, Gap-SVP reduces to Exact-SVP.

We now present known hardness and computability results for Gap-SVP:

Theorem 1.5 ([K05]) α*-Gap-SVP is NP-hard [using randomized reductions] if for any* $\varepsilon > 0$, $\alpha \leq 2^{(\log n)^{\frac{1}{2}-\varepsilon}}$.

Theorem 1.6 ([LLL82]) α*-Gap-SVP is in P for* $\alpha = 2^{\Omega(n)}$.

These two results define a region where α-Gap-SVP is cryptographically interesting: in particular, when $\alpha \in [n, 2^{n^\varepsilon}]$ for some (small) $\varepsilon > 0$. We note that the lower bound on α does not match the result of [K05]—it is an **open question** whether or not α-Gap-SVP for $\alpha < n$ can be used to construct useful cryptographic systems.

We observe that Gap-SVP on its own generally only gives **worst-case** hardness; typically, cryptographic systems require **average-case** hardness to be truly secure. In particular, we need to be able to generate hard instances of a problem efficiently and we need that the average generated instance of a problem is hard. In the next section, we use Gap-SVP to construct an important cryptographic primitive.

1.4 Constructing a Cryptographic System using Gap-SVP

In this section, we leverage Gap-SVP to construct **Collision-Resistant Hash Functions** (CRHF), a standard cryptographic protocol.

In order to properly define the notion of CRHFs, we first define **negligible functions**; our goal will be to make any adversary's probability of success (i.e. probability of breaking the cryptographic system), a negligible function in the security parameter of a system:

Definition 1.7 (Negligible Functions) *A function* $f : \mathbb{N} \to [0, 1]$ *is* ***negligible*** *if for every polynomial* $p(\cdot) : \mathbb{N} \to \mathbb{R}^+$,

$$\exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) < \frac{1}{p(n)}$$

In other words, f vanishes faster than any inverse polynomial function.

We note that the sum of any polynomial number of negligible functions remains negligible. This is important, since it means that even with a polynomial number of attempts, no adversary can **boost** their success probability to significant levels.

We can now define CRHFs:

Definition 1.8 (Collision-Resistant Hash Functions) *Let $n \in \mathbb{N}$ be the **security parameter**. A **Collision-Resistant Hash Function** (CRHF) consists of two algorithms:*

1. $\textsc{Keygen}(1^n)$*: outputs a key k by randomly sampling from some distribution.*
2. $\textsc{Eval}(k, x)$*: defined such that $\textsc{Eval}(k, \cdot) : \{0,1\}^{q(n)} \to \{0,1\}^{p(n)}$ is a deterministic function where $q(n) > p(n)$.*

Additionally, Eval *must satisfy the following for every **probabilistic polynomial-time** (PPT) attacker A: there exists a negligible function μ such that*

$$\mathbb{P}\left[x = x' \wedge \textsc{Eval}(k, x) = \textsc{Eval}(k, x') \middle| k \leftarrow \textsc{Keygen}(1^n); x, x' \leftarrow A(k)\right] \leq \mu(n)$$

Note that requiring that $q(n) > p(n)$ means that Eval **compresses** the input, meaning that there *must* be hash collisions. However, the condition of collision-resistance means that it should be hard for a polynomial-time attacker to find a collision. We additionally observe that the adversary A is given the generated key k, and is allowed to perform any randomized polynomial-time process to attempt to generate a collision. The probability in the definition of collision resistance is over the randomness of Keygen and A.

By using Gap-SVP as the underlying mathematical primitive, it is in fact possible to construct CRHFs:

Theorem 1.9 ([A96, MR04]) *If α-Gap-SVP is hard in the **worst case** for $\alpha = n$, then there exist collision resistant hash functions.*

The construction of [A96, MR04] proceeds by constructing a hard instance of the *Short Integer Solution* (SIS) problem using Gap-SVP, and then using a hard instance of SIS to construct a CRHF. We first define SIS, and then show how to construct a CRHF assuming that SIS is hard.

Definition 1.10 (Short Integer Solution Problem) *Let $n \in \mathbb{N}, q \in \mathbb{N}$ be the dimension and modulus of the problem, respectively. The Short Integer Solution (SIS) problem is defined as follows:*

- *Given: a **randomly sampled** matrix $A \in \mathbb{Z}_q^{n \times m}$ such that $m > n \log(q)$.*
- *Find: $\mathbf{x} \in \{-1, 0, 1\}^m$, $\mathbf{x} \neq \mathbf{0}$ such that $A\mathbf{x} \equiv 0^n \mod q$*

*SIS is **hard** if the following holds: for every $n, \exists q, m$ such that for any PPT adversary Adv, there exists a negligible function μ such that:*

$$\mathbb{P}\left[\mathbf{x} \neq \mathbf{0} \wedge \mathbf{x} \in \{-1, 0, 1\}^m \wedge A\mathbf{x} \equiv 0^n \mod q \middle| A \leftarrow \mathbb{Z}_q^{n \times m}; \mathbf{x} \leftarrow Adv(A)\right] \leq \mu(n),$$

where the probability is over the random sampling of A and the random choices of Adv.

We note that it is equivalently hard to ask simply for a **low-norm** vector $\mathbf{x}$.

Theorem 1.11 *If SIS problem is hard, then CRHFs exist.*

Proof: Suppose we have a hard SIS instance with parameters n, q, m.

We construct a CRHF defined by:

- $\textsc{Keygen}(1^n) :=$ sample a random $A \in \mathbb{Z}_q^{n \times m}$.
- $\textsc{Eval}(A, x \in \{0,1\}^m) := Ax \mod q$.

We first observe that if $m > n \log q$, then Eval does indeed compress its input, as required. We show collision-resistance through a **security reduction**.

Suppose that this CRHF is not collision-resistant. We will show that this implies that SIS is not hard.

Because this CRHF is not collision-resistant, there exists some PPT adversary Adv such that for some non-negligible function β,

$$\mathbb{P}\left[x = x' \wedge Ax \equiv Ax' \mod q \middle| A \leftarrow \textsc{Keygen}(1^n); x, x' \leftarrow Adv(A)\right] \geq \beta(n)$$

But then this implies that $A(x - x') \equiv 0^n \mod q$, and since $x \neq x'$, $x - x' \neq \mathbf{0}$. Further, note that $x - x' \in \{-1, 0, 1\}^m$. This means that Adv can also be used to solve SIS with sampled matrix A by outputting $x - x'$. Thus, SIS cannot be hard, which completes the proof. ■

References

[A96] M. Ajtai. 1996. Generating hard instances of lattice problems (extended abstract). In Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC '96). Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/237814.237838

[A98] Miklós Ajtai. 1998. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98). Association for Computing Machinery, New York, NY, USA, 10–19. https://doi.org/10.1145/276698.276705

[K05] Subhash Khot. 2005. Hardness of approximating the shortest vector problem in lattices. J. ACM 52, 5 (September 2005), 789–808. https://doi.org/10.1145/1089023.1089027

[LLL82] Lenstra, A.K., Lenstra, H.W. & Lovász, L. Factoring polynomials with rational coefficients. Math. Ann. 261, 515–534 (1982). https://doi.org/10.1007/BF01457454

[MR04] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, Italy, 2004, pp. 372-381, doi: 10.1109/FOCS.2004.72.

[S94] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, 1994, pp. 124-134, doi: 10.1109/SFCS.1994.365700.