6.876 Advanced Topics in Cryptography: Lattices	November 9, 2015

Lecture 17

Lecturer: Vinod Vaikuntanathan | *Scribe: Alex Grinman*

Previously,

1. We discussed two notions of trapdoors for an $n \times m$ matrix A:
   - **Type 1:** a "short" full rank basis T $(m \times m)$ such that
   $$AT = 0 \mod q$$
   - **Type 2:** a "short" matrix R such that
   $$AR = G \mod q$$
   where G is the gadget matrix from last class.
2. Using trapdoors, we constructed a "candidate" Digital Signature Scheme. This scheme was broken because we did not have a secret key sampling scheme.

Today: we will fix the candidate Digital Signature Scheme with a discrete Gaussian sampling algorithm (GPV Sampling algorithm).

1 Candidate Digital Signature Scheme

Our candidate digital signature scheme is based off trapdoors and consists of three functions *Gen*, *Sign*, and *Verify*. Additionaly we will use a random oracle function $H : \mathbb{Z}_q \to \mathbb{Z}_q^n$.

- $Gen(1^n, 1^m, q)$: Pick $(A, R) \xleftarrow{\$} TrapSamp(1^n, 1^m, q)$ where *TrapSamp* is the algorithm presented in the last lecture that uniformly at random samples a matrix A and it's Type 2 trapdoor R. The $n \times m$ matrix A is the verification key PK and the trapdoor R is the signing key SK.
- $Sign(R, m)$: "Solve" the equation (using R):
$$A\vec{e} = H(m) \mod q$$
for vector $\vec{e}$. Let the signature $\sigma = \vec{e}$.
- $Verify(A, m, \sigma)$: Check that following holds:
  1. $A\sigma = H(m) \mod q$
  2. $||\sigma|| \leq poly(n)$

  if both conditions succeed, accept, otherwise reject.

The scheme is almost complete, everything has been determined except the "Solve" function. How do we construct a "good" enough solve function that computes the signature $\sigma = \vec{e}$ and can't be forged? We describe what it means for the solve function to be "good" in the next section.

2 A "Good" Solve Function

For the *Solve* function to be good it must be that for most A, all short enough trapdoors R, and polynomially many y_i the following distributions are distributions are statistically close:

$$(A, e \sim D_{\mathbb{Z}_q^m,\sigma}, y_i = Ae) \approx_s (A, e \leftarrow Solve(R, A, y_i), y_i \xleftarrow{\$} \mathbb{Z}_q^n)$$

where $(A, R) \leftarrow TrapSamp(1^n, 1^m, q)$. Hence, we define a "good" *Solve* function which on input A, R and some y_i taken uniformly at random from the range of the random oracle function H, generates a vector e in a distribution that is statistically close to a sampling from a Gaussian distribution over vectors in $\mathbb{Z}_q^m$ with standard deviation σ, such that $y_i = Ae$. Why is this definition good enough?

Claim 1. *Forging Signatures from a "good" Solve is as hard as solving ISIS*

Proof. Suppose there is an adversary α that can forge signatures $e \sim D_{\mathbb{Z}_q^m,\sigma}$ of messages m where $Ae = H(m) = y$. We can use α construct an ISIS solver β as follows. β on input $A \in \mathbb{Z}_q^{n\times m}$, $u \in \mathbb{Z}_q^n$:

1. Generate large values $y, y' \in \mathbb{Z}_q^n$ such that $y - y' = u$.
2. Query α, the solve adversary and ask for a forgeries: $e, e' \leftarrow \alpha(A, y), \alpha(A, y')$ where $Ae = y$ and $Ae' = y'$
3. Output $e - e'$.

Note that $A(e-e') = Ae - Ae' = y - y' = u$, hence $x = e - e'$ is a solution to $Ax = u$, and x is small because both e, e' are statistically small since the solve adversary must produce forgeries that are statistically from a gaussian distribution on $D_{\mathbb{Z}_q^m,\sigma}$. Thus, forging signatures from a "good" *Solve* function can be used to solve ISIS. □

3 "Solve" using GPV Sampling

3.1 What is a discrete Gaussian Distribution over a lattice?

For any $s > 0$, a Gaussian function on $\mathbb{R}^n$ centered c with parameter s is

$$\forall x \in \mathbb{R}^n, \rho_{s,c}(x) = e^{-\frac{\pi||x-c||^2}{s^2}}$$

The *discrete Gaussian distribution* over a lattice $\mathcal{L}(\mathbf{B})$, for any $c \in \mathbb{R}^n$ and $s > 0$ is defined as

$$D_{\perp(\mathbf{B}),s,c} = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\mathcal{L}(\mathbf{B}))} = k \cdot e^{-\frac{\pi||x-c||^2}{s^2}}$$

where the denominator is just normalizing factor for the lattice, making $D_{\perp(\mathbf{B}),s,c}$ proportional to $e^{-\frac{\pi||x-c||^2}{s^2}}$, and s is the *smoothing parameter* for the lattice.

3.2 "Solve" is *GaussSamp*

Theorem 2 ([GPV08])**.** *For all* $\mathbf{B}$ *basis, where* $s > ||B|| \cdot \omega(\log n)$*, the algorithm* $GaussSamp(B, s) \approx D_{\mathcal{L}(\mathbf{B}),s,c}$ *meaning*

1. *Sampling* $e \leftarrow GaussSamp(B, \sigma)$ *is exactly the same as sampling* e *from a discrete Gaussian distribution.*
2. *GaussSamp operates independently of the basis* $\mathbf{B}$*, no matter what* $\mathbf{B}$ *is the result will still be from the distribution.*

In section 4.3 we will prove our construction of *GaussSamp* satisifies this first requirement Theorem 2.

3.3 Converting Trapdoors from Type 2 $\rightarrow$ Type 1

In order to use the our new solve function, *GaussSamp*(B, s), we need to convert the signing key, trapdoor R (type 2), into a trapdoor T (type 1) which is a full-rank "short" basis T such that $AT = 0 \mod q$. Let $G = AR$ be the gadget matrix, and T_G be the type 1 trapdoor for G. To convert a type 2 to a type 1 trapdoor we observe that $A \cdot (RT_G) = GT_G = 0$, where $R \cdot T_G = T$ is a full rank, short basis type 1 trapdoor for A.

4 *GaussSamp*

How do we construct the *GaussSamp* algorithm?

4.1 Idea 1: "Round Off"

A simple starting point is the following algorithm:

1. Sample a vector e^* from a continuous Gaussian distribution with paramter s and center c
2. Use $\mathbf{B}$ to round off e^* to the nearest lattice vector e.

Why does this not work? While this algorithm is well defined, it is not basis independent and therefore does not produce samples from discrete Gaussian over the lattice.

4.2 Idea 2: Rejection Sampling over 1-Dimensional Integer Lattice

Instead of sampling from a continuous Gaussian distribution we construct an algorithm to sample vectors directly from a lattice using a discrete Gaussian distribution. This approach is not obvious since the distribution is infinite and is unclear how to sample succinctly, so we will first start with a simpler algorithm to sample integers which uses rejection sampling on a 1-dimensional lattice $\mathbb{Z}$, and then lift this scheme to an n-dimensional lattice.

Given smoothing parameter s, center c, and a fixed function $t(n)$ define the integer lattice sampling function *GaussSamp1D* as follows:

1. choose $x \leftarrow \mathbb{Z} \cap [c - s \cdot t(n), c + s \cdot t(n)]$ uniformly at random
2. with probability $\rho_{s,c}(x)$ output x, with probability $1 - \rho_{s,c}(x)$ go to step 2.

We now show that *GaussSamp1D* samples from a discrete Gaussian.

Claim 3. *GaussSamp1D for $t(n) = o(\sqrt{\log n})$ samples from a distribution statistically close to $D_{\mathbb{Z},s,c}$*

Proof. From the proofs of the tail inequality on the distribution $D_{\mathbb{Z},s,c}$ in [GPV08] and [Ban95] we have that

$$\Pr_{x \sim D_{\mathbb{Z},s,c}}[|x - c| > s \cdot t(n)] \leq e^{-t(n)^2 \alpha}$$

and therefore for $t > \omega(\sqrt{(}\log n))$, the probability that an x is selected that is outside of the interval $[c - s \cdot t(n), c + s \cdot t(n)]$ is negligible. Hence, the distribution of samples produced in *GaussSamp1D*, since they are kept with probability $\rho_{s,c}(x)$, is statistically close to $D_{\mathbb{Z},s,c}$. $\square$

It is also easy to see that *GaussSamp1D* terminates in polynomial time in n,by substituting the probability density function $\rho_{s,c}(x) = e^{\frac{-\pi||x-c||^2}{s^2}}$ for $s > ||B|| \cdot \omega(\log n)$. Hence, the expected number of iterations is less than $t(n) \cdot \omega(\log n)$.

4.3 Sampling over n-Dimensional Lattices

We can now describe an algorithm to sample vectors from an n-dimensional lattice using *GaussSamp1D* as a subroutine. We will use Babai's nearest plane algorithm to solve CVP, except instead of rounding, we will use *GaussSamp1D* to sample integers from $\approx_s D_{\mathbb{Z},s,c}$. The overall idea of the algorithm is to find the $n-1$ dimension hyperplane that's closest to the target vector, project c onto this hyperplane, and recurse. More formally, the algorithm works as follows:

GaussSamp$(\mathbf{B}, s, c)$:

1. for $i = n \dots 1$:
2. Compute $c'_i = \frac{\langle c_i, \tilde{b_n} \rangle}{\langle \tilde{b_n}, \tilde{b_n} \rangle}$
3. Sample $z_i \leftarrow GaussSamp1D(\frac{s_i}{||\tilde{b_i}||}, c'_i)$
4. Project onto $\mathsf{Span}(b_1, \dots, b_{i-1})$: $c_{i-1} \leftarrow c_i - z_i \cdot b_i$
5. output $\sum_{i=1}^n z_i \cdot b_i$

Now we will prove correctness of *GaussSamp*.

Claim 4. *On any input* $(\mathbf{B}, s, c)$ *where* $s > ||\mathbf{B}||\omega(\log n)$*, and for any output* $v_n = \sum_{i=1}^n z_i^* b_i \in \mathcal{L}(\mathbf{B})$*:*

$$\Pr[v \leftarrow GaussSamp(\mathbf{B}, s, c)] = \frac{\rho_{s,c}(v)}{\rho_s(\mathcal{L}(\mathbf{B}))}$$

Proof. *GaussSamp*$(\mathbf{B}, s, c)]$ outputs $v = \sum_{i=1}^n z_i^* b_i$ if and only if for every sampling $z_i = z_i^*, \forall i = n \dots 1$. Hence,

$$\begin{aligned}
\Pr[v_n = \sum_{i=1}^n z_i b_i] &= \Pr[z_n = z_n^*] \cdot \Pr[v_{n-1} = \sum_{i=1}^{n-1} z_i b_i \mid z_n = z_n^*] \\
&= \frac{\rho_{\frac{s}{||\tilde{b_n}||}, c'_i}(z_n^*)}{\rho_{\frac{s}{||\tilde{b_n}||}, c'_n}(\mathbb{Z})} \cdot \frac{\rho_{\frac{s}{||\tilde{b}_{n-1}||}, c'_{n-1}}(z_{n-1})}{\rho_{\frac{s}{||\tilde{b}_{n-1}||}, c'_{n-1}}(\mathbb{Z})} \\
&= \prod_{i=n}^{1} \frac{\rho_{\frac{s}{||\tilde{b_i}||}, c'_i}(z_i^*)}{\rho_{\frac{s}{||\tilde{b_i}||}, c'_i}(\mathbb{Z})} \\
&= \frac{\prod_{i=n}^1 \rho_s((z_i^* - c'_i) \cdot ||\tilde{b_i}||)}{\rho_s(\mathcal{L}(\mathbf{B}))} = \frac{\rho_s(\sum_{i=n}^1 (z_i^* - c'_i) \cdot \tilde{b_i})}{\rho_s(\mathcal{L}(\mathbf{B}))} \\
&= \frac{\rho_s(v - c)}{\rho_s(\mathcal{L}(\mathbf{B}))} \\
&= \frac{\rho_{s,c}(v)}{\rho_s(\mathcal{L}(\mathbf{B}))}
\end{aligned} \tag{1}$$

where c'_i is as is defined in the *GaussSamp* algorithm above, $\rho_s(\mathcal{L}(\mathbf{B})) = \prod_{i=n}^1 \rho_{\frac{s}{||\tilde{b_i}||}, c'_i}(\mathbb{Z})$, the third-to-last equality follows from the orthonogonality of Gram-Schmidt vectors, and the second to last equality follows from [GPV08] Lemma 4.4. Note that the size requirement for $s > ||\mathbf{B}||\omega(\log n)$ is required for the statistical closeness of the 1-dimensional *GaussSamp1D* to a discrete Gaussian distribution over $\mathbb{Z}$. ☐

References

[Ban95] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices inr n. *Discrete & Computational Geometry*, 13(1):217–231, 1995.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.