Lecture 23: Trapdoored Matrices

Lecturer: Vinod Vaikuntanathan **Date:** April 30, 2026
Scribes: Edward Yu, Sam Zhang

1. Trapdoored matrices

In today's lecture we discuss *trapdoored matrices* and their applications.

Loosely speaking, a trapdoored matrix $A \in \mathbb{F}^{n \times n}$ over field $\mathbb{F}$ is a random-looking matrix for which the linear map $x \to A(x)$ may be computed much faster than the usual quadratic time.

We may imagine some trivial examples of matrices for which $x \to Ax$ may be computed quickly, which are decidely *not* random:

- the matrix of all zeros;
- any low-rank matrix;
- any sparse matrix;
- the identity matrix;
- the Vandermonde matrix over $\mathbb{C}$, used in the FFT, and so on.

Definition 1.1. A *trapdoored matrix* is based on a (nondeterministic) procedure Sample which outputs a pair (M, T_M), where $M \in \mathbb{F}^{d \times n}$ and T_M is a circuit $\mathbb{F}^m \to \mathbb{F}^d$ with size $\tilde{O}(n + d)$. We impose the following requirements:

- **(Correctness)** For all $x \in \mathbb{F}^n$, we have $T_M(x) = Mx$.
- **(Security)** Given $(M, T_M) \leftarrow$ Sample, the distribution of $M \in \mathbb{F}^{d \times n}$ is computationally indistinguishable from some reference distribution $\mathcal{D}$.

The following intuition is useful—given an arbitrary $n \times n$ matrix $M \in \mathbb{F}^{n \times n}$, and a vector $v \in \mathbb{F}^n$, one must take $O(n^2)$ time to compute the product Mv, because the entire information content of M and v must be known. Thus we will have to develop trapdoored matrices which have "lower" information content, but appear to be in our desired random distribution; this is exactly the concept of pseudorandomness.

2. Constructing trapdoored matrices

In this section we give two constructions of trapdoored matrices using hardness assumptions, over two different domains. Note that in both cases we will be working with $n \times n$ square matrices;

we can construct non-square trapdoor matrices by partitioning them into squares in block-matrix form.

2.1. Over $\mathbb{F}_2$, using LPN

We wish to construct trapdoored matrix $m \in \mathbb{F}_2^{n\times n}$; the reference distribution is uniform.

The key idea is that if two matrices A and B admit efficient matrix-vector multiplication, then their sum and product also admit (somewhat slower) efficient matrix-vector multiplication. Note that

- if matrix AB is low-rank, we can perform matrix-vector multiplications Bv efficiently, by multiplying $(AB)x = A(Bx)$.
- if matrix E is sparse, we can perform matrix-vector multiplications Ev efficiently, by simply only perfoming the calculations on the nonzero entries of E;

The assumption we will use in this setting is *learning parity with noise*: Let $A \leftarrow \mathbb{F}_2^{n\times k}$ and $B \leftarrow \mathbb{F}_2^{k\times n}$ be uniformly random, and let E be a noise matrix in $\mathbb{F}_2^{n\times n}$ with each entry set to 1 with probability p independently at random for some p. We assume that if $p \gg 1/k$, the distribution of $AB + E$ is computationally indistinguishable from uniform on $\mathbb{F}_2^{n\times n}$.

We claim that $M = AB+E$ will be the desired trapdoor matrix. To compute Mx, we can individually compute ABx and Ex, then sum. With $p \approx 1/k$ this takes $\sim nk + n^2/k$ time; setting $k \sim \sqrt{n}$, we find that the time needed to compute Mx is $\sim O(n^{3/2})$. This runtime can be improved to $\tilde{O}(n)$ by turning A and B themselves into trapdoored matrices (which involves, as alluded to earlier, partitioning them into square trapdoored-matrix blocks).

2.2. Over the real numbers

Our previous method for constructing trapdoored matrices over $\mathbb{F}_2$ fails if $\mathbb{F}_2$ is replaced with the reals, because the LPN assumption no longer works. We will use an alternative hardness assumption, based on what is called *Kac's walk*.

Let $R_{i,j,\theta}$ be the rotation by θ in the ij-plane, i.e.

$$R_{i,j,\theta} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\theta & \sin\theta & & & \\ & & -\sin\theta & \cos\theta & & & \\ & & & & 1 & & \\ & & & & & \ddots & \end{bmatrix},$$

where the $\cos\theta$ entries appear at positions (i,i) and (j,j).

Now consider a matrix M_k given by $M_k = R_{i_1,j_1,\theta_1} \cdots R_{i_k,j_k,\theta_k}$, for some step count k and random $i, j \leftarrow [n]$ and $\theta \leftarrow [0, 2\pi)$. It is known (in very recent work) that if $k = \Omega(n^2 \log n)$, the distribution of M_k is close to that of a random orthogonal matrix, under the Haar measure. But $n^2 \log n$ steps is too many for us; instead, we will *assume* that by setting $k \sim n \operatorname{polylog} n$, the distribution of M_k is computationally indistinguishable from that of a Haar random orthogonal

matrix. This assumption is likely valid because it can be shown that some types of distinguishers, such as ones that use fewer than k columns of the matrix or ones that compute only low-degree polynomials of the matrix entries, are guaranteed unable to break it.

Given this assumption, one can easily show that M_k for such $k \sim n \operatorname{polylog} n$ is a trapdoored orthogonal matrix, since we can multiply by a sequence of k matrices of the form $R_{i,j,\theta}$ in time $O(n+k)$. It is possible to use this construction to obtain trapdoored Gaussian matrices and other distributions; details omitted.

3. Two applications of trapdoored matrices

3.1. Dimensionality reduction

Recall that the Johnson-Lindenstrauss Lemma allows us to reduce the dimensionality of a point set while (approximately) preserving their pairwise distances:

Lemma 3.1 (Johnson-Lindenstrauss)**.** *Given a finite set of points $S \subset \mathbb{R}^n$ and some $\epsilon > 0$, there exists a linear map $f : \mathbb{R}^n \to \mathbb{R}^d$, such that $d = O(\epsilon^{-2} \log |S|)$, and such that for all $x, y \in S$, we have*

$$(1-\epsilon)\|x-y\| \le \|f(x)-f(y)\| \le (1+\epsilon)\|x-y\|.$$

The proof is relatively straightforward: we can sample $A \sim N(0, 1/d)^{d \times n}$, and show that A satisfies the desired property with high probability.

Now, usually after choosing such A it would take $O(dn)$ time to compute each transform Ax for $x \in S$. However, now suppose we were able to take a trapdoored matrix A instead; then, each transform is now computable in $\tilde{O}(n)$. Furthermore, A still must preserve pairwise distances with high probability, because if it does not, we would be able to distinguish the trapdoored distribution A from the reference distribution of $N(0, 1/d)^{d \times n}$ by checking whether it preserves pairwise distances.

3.2. Secure delegation for matrix multiplication

Suppose that a client has two matrices M_1 and M_2 of size $n \times n$ over a finite field, and wants to delegate the computation of $M_1 M_2$ to a server, without revealing M_1 and M_2. Assume that the client must run in $\tilde{O}(n^2)$ time, but the server can take longer. Then the client can sample trapdoored matrices R_1 and R_2, and send the server $M_1 + R_1$ and $M_2 + R_2$. The server responds with $(M_1 + R_1)(M_2 + R_2)$; the client can recover $M_1 M_2$ from it, since it can compute $M_1 R_2 + R_1 M_2 + R_1 R_2$ in $\tilde{O}(n^2)$ time using the trapdoor, and subtract this from $(M_1 + R_1)(M_2 + R_2)$.

It is possible to do something similar with real-valued matrices; our construction only works for orthogonal matrices which have the necessary group structure over $\mathbb{R}^{n \times n}$. A client who wishes to multiply M_1 and M_2, they can generate orthogonal trapdoored matrices R_1, R_2, R_3, ask the server to multiply $R_1 M_1 R_2^{-1}$ and $R_2 M_2 R_3^{-1}$, receive $R_1 M_1 R_3^{-1}$, and apply R_1^{-1} and R_3 to get their result. (Note that, for this to work, conveniently the Kac walk procedure generates trapdoored matrices R such that $R^{-1} = R^\intercal$ is also a trapdoored matrix.)

4. Thoughts on the future of ML and crypto

- **verification:** if a model can guarantee that its outputs are correct, or (weaker) a model can come with a verifier that checks whether its outputs are correct, then we should be able to use these ideas to raise accuracy of models, maybe train self-improving models, etc.
- **alignment/interpretability:** inspired by research on subliminal learning or communication, all of the safety research being done on looking at model internals or looking at chain-of-thought traces seems like it will ultimately be a dead end; there are too many ways for an adversary to establish communication with a model that is not readable by third-parties. Instead it may be interesting to try to design models with provable guarantees that its behavior gets scrambled when it faces out-of-distribution behavior, a la watermark removal.