# Indian Institute of Technology, Kanpur
# Computer Science and Engineering
# Assignment 3 and 4
# CS670: Cryptographic Techniques for Privacy Preservation

Instructor: Adithya Vadapalli

06/10/2025

This assignment is long! Please start early. Doing this assignment correctly is very important, as every other assignment will be built on this.
**Deadline:** November 14th, 2025, EOD on Hello IITK. No extensions will be given.

---

**Academic Integrity** You have to do the assignment individually. You are encouraged to ask the instructor for help. Sign up on Piazza to ask for help: `https://piazza.com/iitk.ac.in/firstsemester2025/cs670`.
Students should come to the instructor's office hours for help in coding. However, copying code from others or using AI tools without understanding the solution is not allowed and will result in 0 marks for the assignment. Students may be asked to explain their code and solution during evaluation.

## Assignment 3

## Updating Item Profiles

When a user issues a query, the *item profiles* are updated according to:

$$v_j \leftarrow v_j + u_i \left(1 - \langle u_i, v_j \rangle\right),$$

where $u_i$ is the user's profile and $v_j$ is the profile of the item that was queried.

## Challenge

Updating item profiles securely is non-trivial because:

- The servers, who hold secret shares of the item profiles, do not know which item $j$ should be updated.
- The user, on the other hand, knows which item was queried, but does not know the update value
$$M = u_i \left(1 - \langle u_i, v_j \rangle\right),$$
since it depends on $v_j$, which is shared between the servers.

## Ideal (Infeasible) Approach

If the user somehow knew $M$, they could directly perform a private update using a Distributed Point Function (DPF):
$$(k_0, k_1) \leftarrow \mathsf{Gen}(j, M).$$
Each server $P_b$ ($b \in \{0, 1\}$) could then locally update its share of the item profiles as:
$$V_b \leftarrow V_b + \mathsf{EvalFull}(k_b),$$
which adds $M$ to the $j^{\text{th}}$ position of the shared item profile vector without revealing $j$. However, since the user does not know $M$, this approach is not feasible.

## Actual Protocol

To overcome this, we proceed as follows.

**Step 1: User-side DPF generation.** The user generates a DPF that points to the desired index $i^*$ but carries a zero value:
$$(k_0, k_1) \leftarrow \mathsf{Gen}(i^*, 0).$$
Let each key have the structure
$$k_b = (s_b, cw_0, \ldots, cw_h, \mathsf{FCW}),$$
where $\mathsf{FCW}_b$ denotes the final correction word.

The user sends:
$$k_0' = (s_0, cw_0, \ldots, cw_h, \mathsf{FCW}_0) \text{ to } P_0, \quad k_1' = (s_1, cw_0, \ldots, cw_h, \mathsf{FCW}_1) \text{ to } P_1,$$
such that
$$\mathsf{FCW}_0 + \mathsf{FCW}_1 = \mathsf{FCW}.$$

**Step 2: Server-side computation of the update value.** Each server locally computes its share of the update term:

$$M = u_i \left(1 - \langle u_i, v_j \rangle\right).$$

Denote these shares as $M_0$ and $M_1$, such that:

$$M = M_0 + M_1.$$

**Step 3: Adjusting the DPF final correction word.** Each server modifies the final correction word of its DPF key so that the combined DPF now encodes $M$ (instead of 0) at index $i^*$.

Each server sends the masked difference:

$$P_0 : M_0 - \mathsf{FCW}_0, \qquad P_1 : M_1 - \mathsf{FCW}_1.$$

After exchanging these values, both servers compute:

$$\mathsf{FCW}_m = (M_0 - \mathsf{FCW}_0) + (M_1 - \mathsf{FCW}_1).$$

The updated DPF keys are thus:

$$k_b = (s_b, cw_0, \ldots, cw_h, \mathsf{FCW}_m).$$

**Step 4: Applying the update.** Each server evaluates its DPF key and adds the result to its share of the item profiles:

$$V_b \leftarrow V_b + \mathsf{EvalFull}(k_b).$$

This procedure correctly and privately applies the update $M$ to the item profile $v_j$, without revealing the target index $j$ or the update value $M$ to either server or the user.

**Conversion from XOR to Additive Shares.**

1. Note that the output of EvalFull will be in the form of XOR shares, whereas the item profiles are stored as additive shares. To make the update compatible, we must convert the XOR shares into additive shares.

2. This can be done by having one of the parties multiply its vector by $-1$. However, the challenge lies in determining which party should perform this negation. If the wrong party multiplies by $-1$, the result will have a sign error.

3. For the purposes of this assignment, you may assume an insecure method for determining which party negates its vector. Implementing the conversion securely will earn you bonus points.

**Grading**

- Correctness and Security: 80%
- Code clarity and documentation: 20%

# Assignment 4

In this assignment, you will evaluate your protocol. Vary the number of queries, items, and users, and plot the time taken for one user profile update and one item profile update.