CIS 552: Database Design

Final Project Report

Oblivious Transfer in Financial Transactions

Aditya Sahu

Group 18

UmassD ID: 02129089

**Overview**

Oblivious transfer helps keep financial transactions secure by encrypting and storing them in a way that maintains privacy. When someone needs to access specific information, they can do so without exposing unnecessary details. It's a way to balance data accessibility with the need for privacy and security. OT has many practical applications. Private Financial Transactions: In a scenario where individuals want to make financial transactions without revealing specific details to the other party, oblivious transfer can be applied. The payer engages in an oblivious transfer with the payee, allowing them to transmit the encrypted transaction details.

**Problem**

In today's world, where digital transactions and data sharing are prevalent, privacy and security are of utmost importance. Oblivious transfer is needed in today's world to address the growing concerns related to privacy, security, and user control over personal data. It provides a way to harness the benefits of digital transactions and collaborations while minimizing the risks associated with unauthorized access and data exposure.

**Goals**

1. Enhance Data Privacy: The primary goal is to enhance the privacy of financial transactions stored in the database. This involves implementing robust encryption techniques, such as column-level encryption and homomorphic encryption, to ensure that sensitive details remain confidential. The application of oblivious transfer during querying and retrieval further strengthens data privacy by allowing selective disclosure of information

2. Facilitate Secure Collaborations: Another goal could be to facilitate secure collaborations between entities or parties involved in financial transactions. This involves implementing secure multi-party computation (SMPC) or similar protocols that enable collaborative analysis without exposing the raw transaction data. The project aims to balance data sharing for joint insights and maintaining the privacy of individual transactions.

**SQL Queries and C Code**

The project assumes a database that stores financial records. For the sake of simplicity I have created a naive database in which I ran some queries to create table, insert data, drop table.

create database projectdb;

use projectdb;

create table financial\_transactions (

transaction\_id int primary key,

Amount Decimal(10,2) not null,

timestamp int NOT NULL,

sender\_account varchar(50) not null,

reciever\_account varchar(50) not null,

status VARCHAR(20) DEFAULT 'Pending'

);

select \* from financial\_transactions;

insert into financial\_transactions values

( 158, 59768.92, 169, '29667X', '99667X', 'Completed'),

( 161, 69765.92, 152, '89667Y', '79667X', 'Completed'),

( 172, 79755.92, 189, '59667Y', '69667X', 'Completed'),

( 175, 89762.92, 195, '39669X', '49667X', 'Pending');

select \* from financial\_transactions;

However, in a real-world scenario, we will require a DBMS that uses OT on the fly when a remote computer tries to access data. Basically, the C code that it lies in my repository should be injected in the database management systems so that whenever a person is querying data that C code should be executed before the query is run.

**Limitations**

Due to time constraints, the lack of DBMS integrated with OT is realized. Future scope would be to integrate the C code and the DBMS to create a full fledged application for querying database using OT.