Privacy-Preserving Logging and Recovery for Growing Databases

Maria Fanelle, Johes Bater

1. Introduction:

- Database recovery logs record every transaction that occurred in the database.
- These transactions could contain **sensitive information**, such as names, addresses, and financial information.
- The **number of transactions** at any given time can reveal the behavior patterns of database users.
- Commonly used recovery algorithms are not designed with privacy in mind; instead, they focus primarily on achieving optimal recovery speed, efficiency, and scalability.
- A database can provide **privacy guarantees** with a minimal performance tradeoff.

2. Problem

- Problem: Database recovery logs **reveal** information about transactions occurring.
- An adversary can analyze the log and use it to **re-identify** individual transactions. The details from individual transactions and the total number of transactions executed should be kept private.
- Without a log, a database cannot recover from failure.
- A standard database log will show the data being inserted and reveal when certain transactions occur.
- A simple implementation of a privacy-enhanced database log grows at a uniform rate, with a constant number of transactions being executed against the database.
- Differential privacy is a statistical technique that provides a strong privacy guarantee. A database system can generate dummy transactions to **obscure** usage patterns and **protect** sensitive information.
- **Question:** How can we recover from database failure while ensuring individual privacy?

3. Proposed Solution:

We propose the following system:

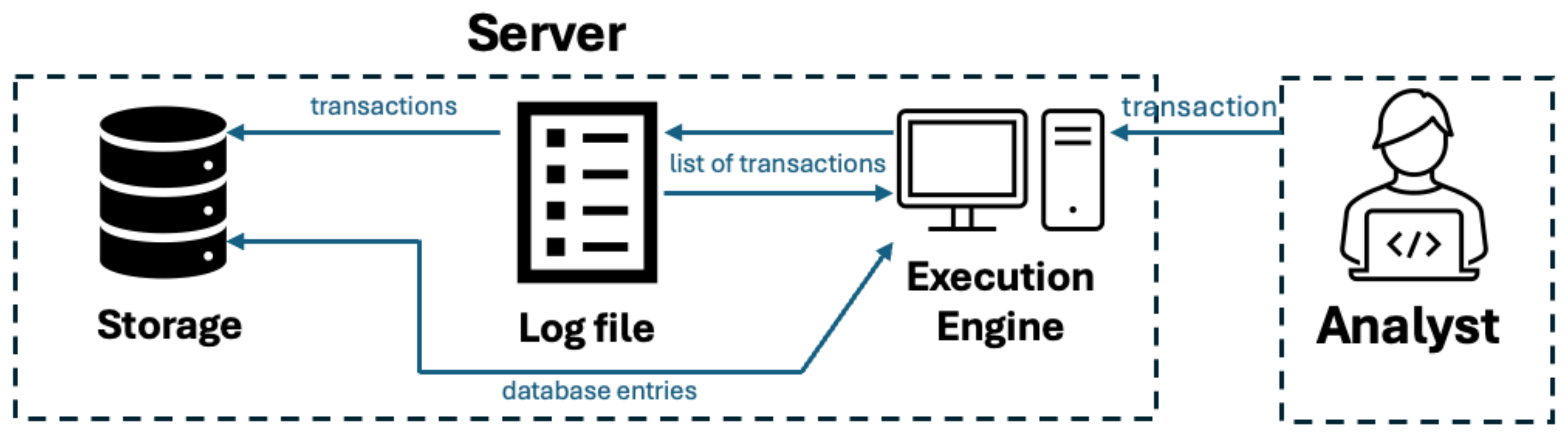
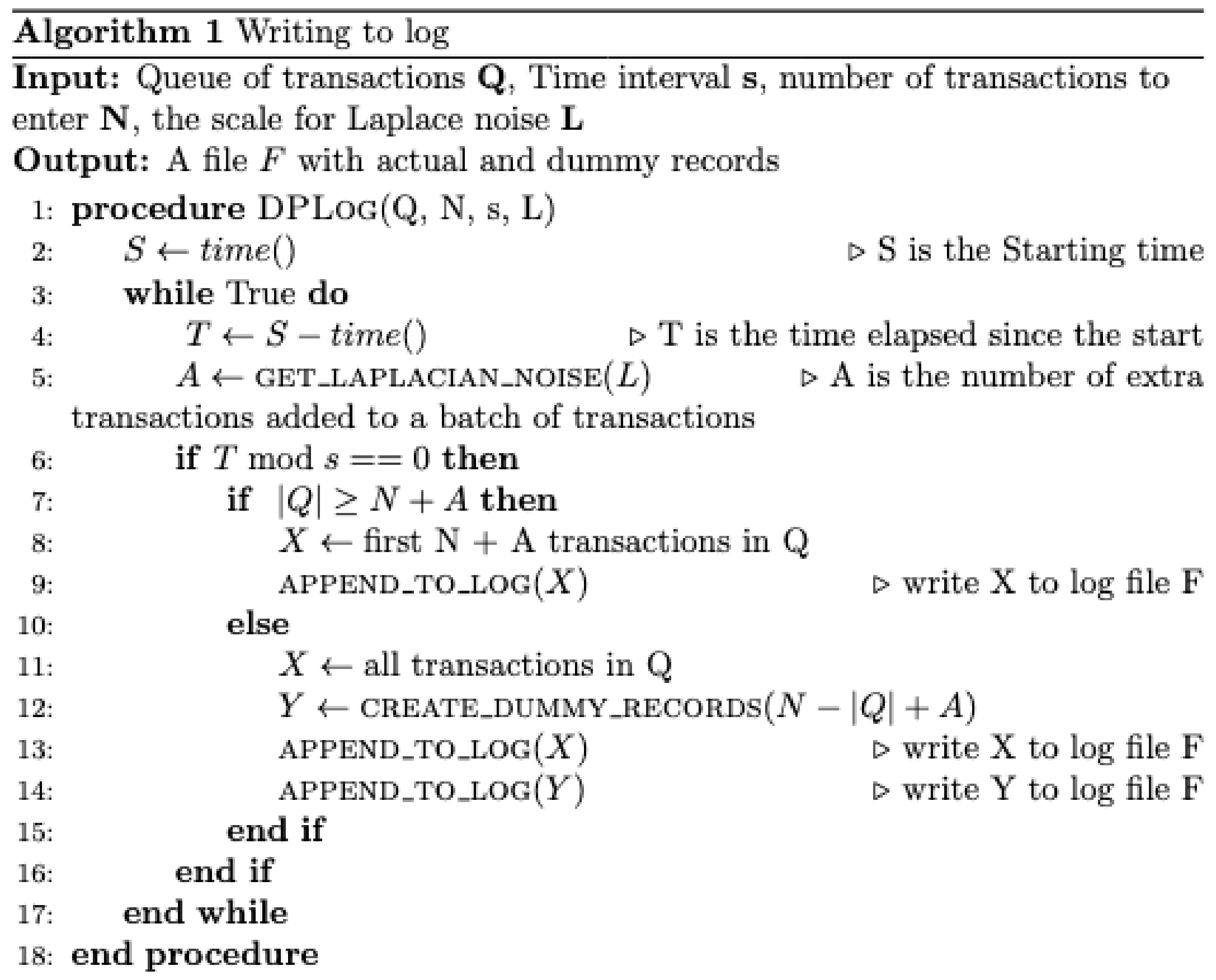


Figure 1: System Design

- The analyst sends a transaction to **an untrusted server**.
- The execution engine batches the transactions and sends them to the log within an assigned timespan. It appends a differentially private number of transactions to each batch.
- If there are not enough real transactions to meet the minimum required, artificial transactions will complete the batch. These artificial transactions mask the content of the real transactions and obscure the behavioral patterns of the analyst.
- Upon system failure, the database **recovers** both real and dummy transactions in the order they were appended.



Algorithm 1 Writing to log

Input: Queue of transactions **Q**, Time interval **s**, number of transactions to enter **N**, the scale for Laplace noise **L**

Output: A file F with actual and dummy records

```
procedure DPLog(Q, N, s, L)
    $S \leftarrow time()$                                ▷ S is the Starting time
    while True do
        $T \leftarrow S - time()$                        ▷ T is the time elapsed since the start
        $A \leftarrow$ GET_LAPLACIAN_NOISE($L$)          ▷ A is the number of extra transactions added to a batch of transactions
        if $T$ mod $s == 0$ then
            if $|Q| \geq N + A$ then
                $X \leftarrow$ first N + A transactions in Q
                APPEND_TO_LOG($X$)                       ▷ write X to log file F
            else
                $X \leftarrow$ all transactions in Q
                $Y \leftarrow$ CREATE_DUMMY_RECORDS($N - |Q| + A$)
                APPEND_TO_LOG($X$)                       ▷ write X to log file F
                APPEND_TO_LOG($Y$)                       ▷ write Y to log file F
            end if
        end if
    end while
end procedure
```

4. Preliminary Results:

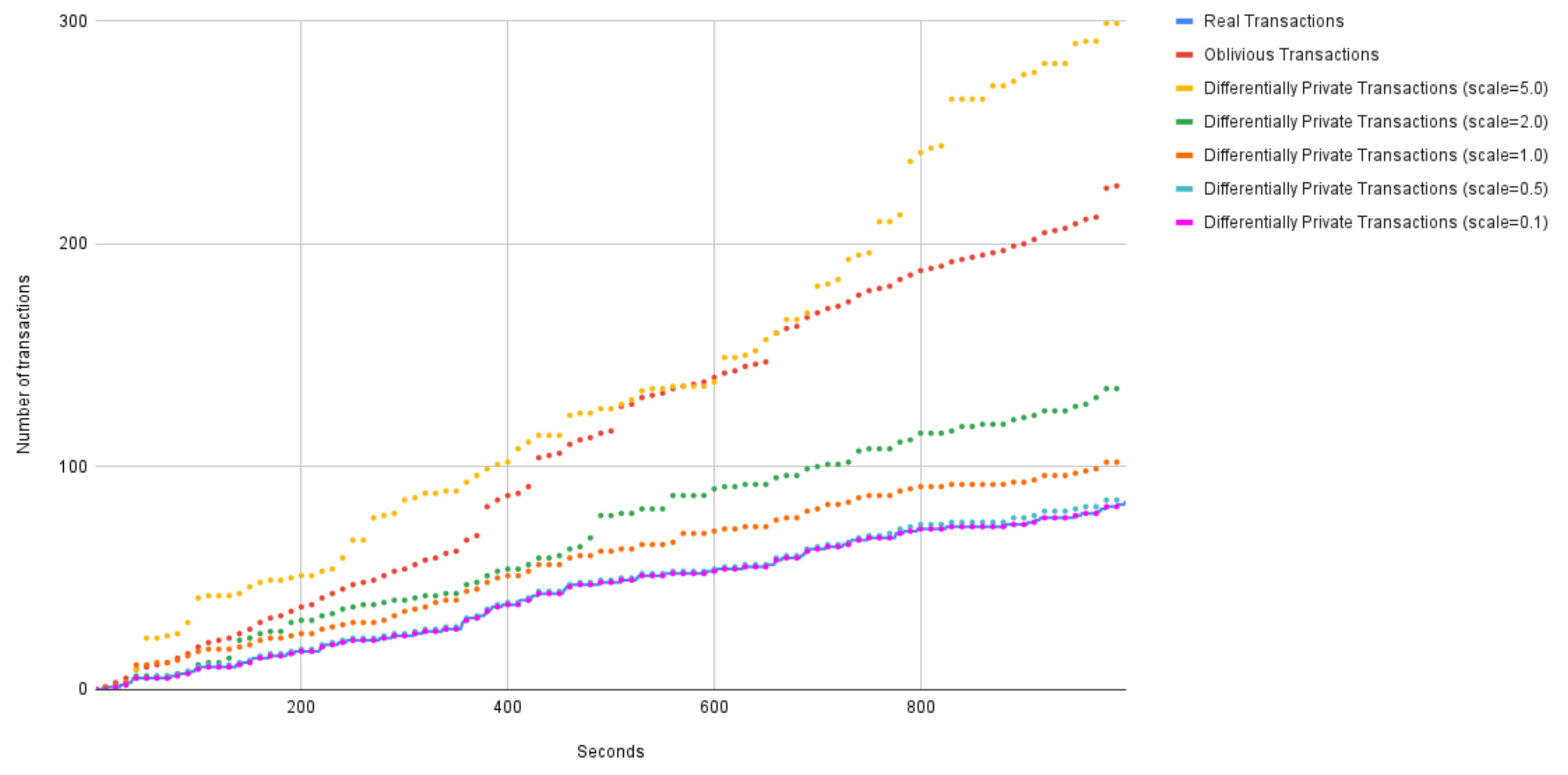


Chart 1: Number of transactions per logging technique

- The approach with a scale of 0.1 or 0.5 was close to the real transaction count, adding only a small number of dummy transactions. It was the most optimal for obfuscating the number of entries.
- The differentially private approaches with scales of 1.0, 2.0, and 5.0, along with the oblivious approach, caused the number of appended transactions to increase rapidly, leading to significantly higher memory consumption.
- Although recovery time was not measured in these results, it is likely that a database would take significantly longer to recover from a larger log file. This is an important consideration for privacy-preserving recovery performance optimizations.
- The scale for each differentially private approach is calculated by dividing the sensitivity, which refers to how much a result can change when a data point is altered, by epsilon, the level of privacy guaranteed. The noise added is proportional to the change in the result based on the scale value. A smaller scale results in less noise being added, providing higher accuracy but weaker privacy protection.