Introduction

- objective:
  - given a database with some disclosure policy
    - e.g., attribute X is private, but aggregated values of X over different subsets of the records may be available
    - detect or prevent violations of the disclosure policy (i.e., disclosure of private information)
  
- detection = off-line query auditing
  - the process of examining queries that were answered in the past to determine whether answers to these queries could have been used to obtain confidential information forbidden by the disclosure policy

- prevention = on-line query auditing
  - examine current query in real-time, and deny queries that could potentially cause a breach of privacy

- an alternative approach to disclosure prevention:
  - add noise or otherwise perturb the query results supplied to the user
  - drawback: may introduce bias in the result
**Notation**

- $n$ denotes the total number of records in the DB
- $X = \{x_1, x_2, \ldots, x_n\}$ are the private attribute values in the records
- $q = (X_q, f)$ is an aggregate query, where
  - $X_q$ specifies a subset of records, called the query set
  - $f$ is aggregation function such as MAX, MIN, SUM, AVG, MEDIAN
- $a = f(X_q)$ is the result of applying $f$ to $X_q$

**The full disclosure model**

- given
  - a set of private values $X = \{x_1, x_2, \ldots, x_n\}$
  - a set of queries $Q = \{q_1, q_2, \ldots, q_t\}$ and corresponding answers $A = \{a_1, a_2, \ldots, a_t\}$
  - an element $x_i$ is fully disclosed by $(Q, A)$ if it can be uniquely determined
    - i.e., $x_i$ is the same in all possible data sets $X$ consistent with the answers $A$ to the queries $Q$

- example:
  - let $n = 3$
  - let $Q = \{(\text{ALL}, \text{MAX}), (\text{ALL}, \text{SUM})\}$
  - assume that $A = \{5, 15\}$
    - $\text{MAX}(x_1, x_2, x_3) = 5$ and $\text{SUM}(x_1, x_2, x_3) = 15$
    - then one can deduce that $x_1 = x_2 = x_3 = 5$, i.e., $x_i$ is fully disclosed for all $i$
Off-line query auditing

- given
  - a set of private values $X = \{x_1, x_2, \ldots, x_n\}$
  - a set of queries $Q = \{q_1, q_2, \ldots, q_t\}$ and corresponding answers $A = \{a_1, a_2, \ldots, a_t\}$
- determine if any $x_i$ is fully disclosed

- example:
  - let the elements of $X$ be real-valued from an unbounded range
  - let $Q$ be a set of SUM queries
  - an auditor essentially needs to solve a system of linear equations
    - each query can be represented by a binary $n$-dimensional vector
    - find a maximal set of linearly independent query vectors (this has complexity $O(n^2 t)$)
    - these query vectors (considered as rows) form a matrix
    - if the matrix has size $n \times n$, then it can be inverted and all $x_i$’s are disclosed (this has complexity $O(n^3)$)
    - otherwise the matrix can be diagonalized (this has complexity at most $O(n^3)$), and if the resulting matrix has a row with a single non-zero element, then some element of $X$ is disclosed
    - overall complexity is at most $O(n^3 + n^2 t)$
  - polynomial time (efficient) off-line query auditing is possible in case of SUM queries over real-valued data

State-of-the-art

- efficient off-line query auditors exist for
  - SUM, MEDIAN, and AVG queries
  - combinations of MAX and MIN queries over real-valued data
- no significant progress has been made in auditing arbitrary combinations of aggregate queries
- some hardness results
  - e.g., there is no polynomial time full-disclosure auditing algorithm for SUM and MAX queries unless $P=NP$
- full-disclosure auditing of sum queries over boolean data is coNP-hard
- there exists an efficient polynomial time algorithm, however, in the special case where the queries are 1-dimensional
  - i.e., for some ordering of the elements in $X$, the query set for each query involves a consecutive sequence of $x_i$’s
  - e.g., number of HIV-positive persons in age groups
On-line query auditing

- given
  - a set of private values $X = \{x_1, x_2, \ldots, x_n\}$
  - a set of queries $Q = \{q_1, q_2, \ldots, q_{t-1}\}$ and corresponding answers $A = \{a_1, a_2, \ldots, a_{t-1}\}$ already returned
  - a new query $q_t$
- determine if $q_t$ can be answered or it should be denied in order to prevent disclosure of private data
- note: any previous answer in $A$ can be true response or denial

A bad approach

- can we apply an off-line auditor directly to solve the on-line auditing problem?
  - let $Q'$ be the subset of queries in $Q$ that has been responded
  - the corresponding answer set is $A'$
  - run the off-line auditor with $(Q' \cup \{q_t\}, A' \cup \{a_t\})$
  - if some data is disclosed, then deny response, otherwise return $a_t$
- this does not work in general, because denials also leak information!
- example:
  - let $n = 3$ and $X = \{5, 5, 5\}$
  - let $Q = \{\{\text{ALL, SUM}\}\}$, then $A = \{15\}$
  - let $q_2 = \{\text{ALL, MAX}\}$
  - this is denied (see previous example on slide 33)
  - however, one can still figure out that all $x_i$'s are 5
  - $\text{MAX}(x_1, x_2, x_3)$ cannot be smaller than 5, otherwise the SUM cannot be 15
  - if $\text{MAX}(x_1, x_2, x_3) > 5$ then the query would have been answered
    $\Rightarrow \text{MAX}(x_1, x_2, x_3)$ must be 5
Another bad idea

- deny whenever the off-line auditor does, and in addition, randomly deny some queries that would normally be answered
  - now denials leak less information, but leakage is not generally prevented
  - the auditing algorithm needs to remember which queries were randomly denied, since otherwise an attacker can repeatedly pose the same query until it is answered
  - a difficulty is then to determine whether two queries are equivalent
    - the computational hardness of this problem depends on the query language, and may be intractable, or even undecidable

Simulatable on-line auditing

- crucial observation:
  - query denials have the potential to leak information if in choosing to deny, the auditor uses information that is unavailable to the attacker (i.e., the answer to the current query)

- the idea behind simulatable auditors:
  - the attacker is able to simulate or mimic the auditors decisions to answer or deny a query
  - as the attacker can equivalently determine for himself when his queries will be denied, denials provably leak no information

- formal definition:
  - an online auditor $B$ is simulatable, if there exists another auditor $B'$ that is a function of only $Q \cup \{q_t\} = \{q_1, q_2, \ldots, q_t\}$ and $A = \{a_1, a_2, \ldots, a_{t-1}\}$, and whose output on $q_t$ is always equal to that of $B$
A sufficient condition for simulatability

- with each new query, the auditor should determine if there is any possible data set, consistent with all past responses, in which the answer to the current query would cause some element to be fully disclosed
- if so, the query should be denied, else it can be answered
- note that this is a condition that an attacker could check for himself and predict denials

- example revisited:
  - let $n = 3$ and $X = \{x_1, x_2, x_3\}$
  - $q_1 = (\text{ALL, SUM})$ can be responded
  - then $q_2 = (\text{ALL, MAX})$ should always be denied (even if for the particular values of $x_1, x_2, x_3$, it would be safe to respond)

The partial disclosure model

- motivation:
  - even if a private value cannot be uniquely determined, it might still be determined to lie in a tiny interval, or in a large interval with a heavily skewed distribution
  - one might consider this to be sufficient disclosure

- in the partial disclosure model, the data is assumed to be drawn from some distribution $D$ on $(-\infty, \infty)^n$ that is known to both the attacker and the auditor

- in addition, we allow the auditor to be randomized
  - i.e., it’s decision to answer or deny a query need not be deterministic
Definition of partial disclosure

- a sequence of queries and answers, \( q_1, q_2, \ldots, q_t \) and \( a_1, a_2, \ldots, a_t \) is said to be \( \lambda \)-safe with respect to a data element \( x_i \) and an interval \( I \subseteq (\infty, \infty) \) if

\[
\frac{1}{1 + \lambda} \leq \frac{\Pr_D(x_i \in I | q_1, \ldots, q_t, a_1, \ldots, a_t)}{\Pr_D(x_i \in I)} \leq (1 + \lambda)
\]

intuitively: the attacker's confidence that \( x_i \) is in \( I \) does not change significantly upon seeing the queries and answers

- let us define the predicate \( \text{AllSafe} \) as follows:

\[
\text{AllSafe}(q_1, q_2, \ldots, q_t, a_1, a_2, \ldots, a_t) = \begin{cases} 
1, & \text{if } q_1, q_2, \ldots, q_t, a_1, a_2, \ldots, a_t \text{ is } \lambda \text{-safe for all } i \text{ and every } I \\
0, & \text{otherwise}
\end{cases}
\]

\( (\lambda, \delta, T) \)-privacy game:
- there are up to \( T \) rounds
- in each round \( t \):
  - the attacker (adaptively) poses a query \( q_t = (X_t, f_t) \)
  - the auditor determines whether \( q_t \) should be answered; the auditor responds with \( a_t = f_t(X_t) \) if \( q_t \) is allowed and with \( a_t = \text{"denied"} \) otherwise
  - the attacker wins if \( \text{AllSafe}(q_1, q_2, \ldots, q_t, a_1, a_2, \ldots, a_t) = 0 \)

- an auditor is \( (\lambda, \delta, T) \)-private if for any attacker \( A \)

\[
\Pr(A \text{ wins the } (\lambda, T) \text{-privacy game}) \leq \delta
\]

where the probability is taken over the distribution \( D \) that the data comes from and the coin tosses of the randomized auditor and the attacker

- note: simulatability can and should be imposed on auditors as before
State-of-the-art

- randomized auditors have been developed for
  - SUM queries,
  - MAX queries, and
  - combinations of MAX and MIN queries

- efficiency?
- hardness results?

Challenges in query auditing

- Privacy definitions
  - full disclosure, partial disclosure, perfect privacy, differential privacy, …
  - the assumption is that there is one probability distribution $D$ from which the data is generated and which is known to both the attacker and the auditor
  - in reality, there are two other distributions, the attacker’s prior and the auditor’s prior, and these three distributions may be different

- Algorithmic limitations
  - on-line simulatable algorithms for auditing aggregate queries require sampling a data set consistent with a given set of queries and answers → this procedure may be computationally prohibitive
  - while there has been some investigation into auditing SUM, MAX, MIN, MEDIAN queries, intermingling these queries has proven to be a greater challenge
  - auditing Select-Project-Join queries
Challenges in query auditing

- **Collusion**
  - collusion is a largely unaddressed issue in most interactive data sharing mechanisms today
  - in the absence of any obstacles to collusion, multiple users can pool together answers that are individually safe but together may leak information

- **Utility**
  - while there have been some initial analyses on the utility of online auditors, utility is a dimension that is not well understood
  - how should we even define utility?
    - e.g., expected number of denials in a random sequence of aggregate queries?
    - but in reality, queries are likely to come from a non-uniform distribution
  - there might be some important, fairly generic queries, that should always be answered
  - in general, we would like to ensure that a database will not be rendered useless with too many denials, and to this end, it might well be worthwhile to sacrifice some privacy for greater utility

---

Private Information Retrieval

*Foundations of Secure e-Commerce (bmevihim219)*

Dr. Levente Buttyán
Associate Professor
BME Híradástechnikai Tanszék
Lab of Cryptography and System Security (CrySyS)
butyan@hit.bme.hu, buttyan@crys.hu
Problem formulation

- Alice wants to obtain information from a database, but she does not want the database to learn which information she wanted
  - e.g., Alice is an investor querying a stock-market database
  - e.g., Alice is a company querying a patent database
- a trivial solution is for Alice to download the entire database
- Can the problem be solved with less communications?

- typical model:
  - the database is an n-bit string: \( X = x_1 x_2 \ldots x_n \)
  - Alice is interested in \( x_i \)
  - the database should not be able to learn \( i \)

Some negative results

- if Alice uses a deterministic scheme then \( n \) bits must be transferred (even if there are multiple non-communicating copies of the database)
  \( \rightarrow \) Alice should use coin flips (a randomized algorithm)

- if the database has unlimited computational power and there’s only a single copy of the database then \( n \) bits must be transferred
  \( \rightarrow \) there’s hope if the database can only perform efficient computations (i.e., it is computationally bounded)
  \( \rightarrow \) there’s hope if the database has unlimited computational power but there are multiple non-communicating copies of the database
An example PIR protocol

- assume that there are 4 copies of the database
- the bits of X are arranged in a $n^{1/2} \times n^{1/2}$ matrix
- Alice wants to retrieve $x_{ij}$ (1 <= i, j <= $n^{1/2}$)

protocol:
- Alice generates two random bit strings s and t of length $n^{1/2}$
- let s' be the same as s but with the i-th bit flipped, and let t' be the same as t but with the j-th bit flipped
- Alice sends
  - s and t to DB0
  - s and t' to DB1
  - s' and t to DB2
  - s' and t' to DB4
- each DB returns a single bit computed as the XOR of bits $x_{ab}$ where the a-th bit of s (or s') and the b-th bit of t (or t') are both equal to 1
- Alice XORs the received bits, and the result gives $x_{ij}$

Why does it work?

\[
\begin{array}{c}
\text{s} = \begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\end{bmatrix} \\
\text{t} = \begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\end{bmatrix} \\
\text{i} \\
\text{j}
\end{array}
\quad
\begin{array}{c}
\text{s'} = \begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\end{bmatrix} \\
\text{t'} = \begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\end{bmatrix} \\
\text{i} \\
\text{j}
\end{array}
\]
Why is it private?

- each database receives two random vectors that are independent of $i$ and $j$
- no information on $i$ and $j$ is leaked to the database

Inf. theoretic vs. computational PIR

- information theoretic PIR protocols leak no information (in information theoretic sense) about the index requested by Alice
  - they withstand attacks even from a database with unlimited computational power

- computational PIR (CPIR) protocols provide weaker guarantees: they ensure only that the database cannot get any information unless it solves a computationally hard problem (reduction)

- information theoretic PIR protocols require more than one non-communicating copies of the database, while CPIR protocols with low communication overhead exist even for the single database case
An example CPIR protocol

- Preliminaries
  - Let $m$ be a positive integer
  - A number $a$ is a quadratic residue (QR) mod $m$, if there’s an integer $x$ such that $x^2 \mod m = a$
  - Otherwise, $a$ is quadratic non-residue (QNR) mod $m$
  - It is computationally hard to distinguish numbers that are QRs mod $m$ from numbers that are QNRs mod $m$, unless one knows the factorization of $m$

- Setup
  - The bits of $X$ are arranged in a $n^{1/2} \times n^{1/2}$ matrix
  - Alice wants to retrieve $x_{ij}$ ($1 \leq i, j \leq n^{1/2}$)

- Protocol:
  - Alice chooses at random a large integer $m$ (together with its factorization)
  - She generates $n^{1/2}$-1 random QRs mod $m$: $a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots$
  - She generates a random QNR mod $m$: $b_i$
  - Alice sends $a_1, a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots$ to the database
  - The server cannot make the difference between QRs and QNRs mod $m$, so from the server’s point of view, the received vector is just an array of random numbers: $u_1, u_2, \ldots$
  - For each column $c$ of $X$, the database computes
    \[ v_c = u_1 x_1 c u_2 x_2 c \ldots \mod m \]
  - The database responds with $v_1, v_2, \ldots$
  - Alice verifies if $v_j$ is a QR or a QNR mod $m$
    - If QR, then $x_{ij} = 0$
    - If QNR then $x_{ij} = 1$
Why does it work?

- if \( x_{ij} = 0 \) then only QRs are multiplied, otherwise QRs are multiplied with a single QNR
- it is known that \( QR \times QR = QR \) and \( QR \times QNR = QNR \)

\[
X = \begin{bmatrix}
x_{11} & \cdots & x_{1j} & \cdots \\
\vdots & \ddots & \vdots & \ddots \\
x_{i1} & \cdots & x_{ij} & \cdots \\
\vdots & \ddots & \vdots & \ddots \\
\end{bmatrix}
\]

\[
U = \begin{bmatrix}
a_1 \\
\vdots \\
\vdots \\
a_i \\
\vdots \\
\vdots \\
b_i \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

\[
v_j = a_1^{x_{1j}} \cdots b_i^{x_{ij}} \cdots \text{mod} \ m
\]

State-of-the-art

- best known information theoretic PIR protocol is based on representing the database as a polynomial, and requires the transmission of \( n^{O(\log \log k / k \log k)} \) bits (where \( k \) is the number of copies of the database)
- CPIR schemes have been constructed based on the difficulty of the Quadratic Residue Problem \( (O(n^2)) \) and the \( \phi \)-hiding problem \( (O((\log n)^a)) \), and based one-way permutations \( (n-o(n)) \)
- connections of CPIR to oblivious transfer, collision resistant hash functions, function hiding public key crypto, complexity theory in general have been studied
Variants of PIR and CPIR

- block PIR
  - what if Alice wants a block of bits (of size m)?
  - can we do better than invoking a PIR protocol m times?
- robust PIR
  - what if some of the database copies break down or return false answers (Byzantine failure model)?
- t-private PIR
  - how to ensure that even t colluding databases cannot figure out in which bit Alice is interested in?
- symmetric PIR
  - how to prevent Alice to learn more than just the bit she is interested in?
- PIR with preprocessing
  - the database usually has to do O(n) computations
  - can this be cut down?

Locally decodable codes (LDCs)

- error correcting codes
  - add redundancy to a message → codeword
  - send over noisy channel
  - recover message even if some fraction of the codeword bits are corrupted

  - in practice, longer messages are partitioned into smaller blocks and each block is coded separately
    - this allows efficient random access to message bits (one must decode only a fraction of the received codewords)
    - however, even if a single codeword is lost (unrecoverable), then the message cannot be recovered

  - if the entire message would be encoded as a single large block
    - this would improve robustness
    - but random access would require decoding the entire message (typically prohibitively expensive)
Locally decodable codes (LDCs)

- LDCs simultaneously provide random access retrieval and high noise resistance
- this is achieved by allowing the reliable reconstruction of any bit of the message from a small number of randomly chosen codeword bits

**Definition:** A \((k, \delta, \epsilon)-LDC\) encodes \(n\) bit messages into \(N\) bit codewords such that every bit \(x_i\) of the message can be recovered with probability \(1-\epsilon\) by a randomized decoding procedure that reads only \(k\) codeword bits, even if at most \(\delta N\) bits of the codeword are corrupted

- local decodability comes at a price of loss in terms of code efficiency \((N >> n)\)
- finding more efficient (optimal) LDCs is an active research area and a major challenge

**Example: \((2, \delta, 2\delta)\)-Hadamard**

- encodes \(n\) bit messages into \(2^n\) bit codewords
- let \(H\) be a binary matrix that contains in its columns all the possible \(n\) bit vectors \((H\) is an \(n \times 2^n\) matrix)
- encoding: \(y = C(x) = xH\)
- decoding (of the \(i\)-th bit of \(x\)):
  - pick a random \(n\)-bit vector \(t\), and let \(t'\) be the same as \(t\) but with the \(i\)-th bit flipped
  - \(x_i = y_t \text{ XOR } y_{t'}\)
- probability of successful decoding
  - at most \(\delta N\) bits of \(y\) are corrupted \~ each bit in \(y\) is corrupted with probability \(\delta\) (independently from the other bits)
  - the probability that \(y_t\) or \(y_{t'}\) is corrupted is \(2\delta\)
  - the probability that both \(y_t\) and \(y_{t'}\) are intact (and hence the decoding of \(x_i\) is successful) is \(1-2\delta\)
LDCs and the PIR problem

- LDCs yield efficient PIR schemes and vice versa
- all recent construction of information theoretic PIR schemes work by first constructing LDCs and then converting them into PIR protocols
- general procedure to obtain a k-server PIR scheme from a (perfectly smooth) k-query LDC:
  - each of the k database servers encodes the database X with the LDC and stores C(X)
  - if Alice is interested in \(x_i\), she generates k random queries \(q_1, q_2, \ldots, q_k\), such that \(x_i\) can be recovered from \(C(X)q_1, \ldots, C(X)q_k\), and sends \(q_j\) to DB\(_j\)
  - each server DB\(_j\) responds with one bit \(C(X)q_j\)
  - Alice combines the responses to obtain \(x_i\)
- privacy
  - perfect smoothness of the LDC means that individual queries are distributed perfectly uniformly over the codeword bits
  - thus, in the PIR scheme, every query \(q_j\) is independent from i, and hence, reveals no information on i

Summary

- privacy problems in statistical databases
  - Query Auditing for preventing private information disclosure
  - off-line vs. on-line query auditing
  - simulatable on-line query auditors
  - disclosure models: full disclosure, partial disclosure

- privacy for users accessing public databases
  - Private Information Retrieval
  - information theoretic and computational PIR schemes
  - locally decodable codes and PIR