

Privacy Enhancing Techniques

Information Security (bmevihim102)

Dr. Levente Buttyán

associate professor

BME Híradástechnikai Tanszék

Lab of Cryptography and System Security (CrySys)

buttyan@hit.bme.hu, buttyan@crysys.hu



Introduction

- privacy is the right of an individual to control how information about him/her is collected, stored, and shared
 - people don't like if their personal information such as their religion, sexual orientation, political affiliations, or personal activities are revealed
 - this may be to avoid discrimination, personal embarrassment, or damage to one's professional reputation
- new technologies also create new ways to gather and process private information
- today, as large scale information systems become more common, there is a lot of information stored in many databases worldwide and an individual has no way of controlling (or even knowing) of the information about themselves that others may have access to
 - such information could potentially be sold to others for profit and/or be used for purposes not known to the individual of which the information is about
 - consequences of a violation of privacy can be more severe



Introduction (cont'd)

- privacy is not really a technical issue: it needs law and law enforcement
- however, technical solutions can help to retain control over private information in some cases
 - encryption and access control techniques
 - anonymous communication techniques
 - in general: privacy enhancing technologies



Objectives of this lecture

- see some examples on how technical approaches can help to retain privacy in IT systems
- get an introduction into anonymous communication techniques used over the Internet
 - understand how a MIX network works
 - get some more insight into the engineering details of a practical anonymity system (Tor)
- get an idea on how anonymity can be measured
- get an overview on privacy problems and solutions in databases
 - preventing private information disclosure in statistical databases by query auditing
 - private information retrieval from public databases



Outline

- introduction
- **anonymous communication techniques**
- anonymity metrics
- Query Auditing
- Private Information Retrieval (PIR)
- summary



Basic anonymity concepts

- What do we want to hide?
 - sender anonymity
 - attacker cannot determine who the sender of a particular message is
 - receiver anonymity
 - attacker cannot determine who the intended receiver of a particular message is
 - unlinkability
 - attacker may determine senders and receivers but not the associations between them (attacker doesn't know who communicates with whom)
- From whom do we want to hide this?
 - external attackers
 - local eavesdropper (sniffing on a particular link (e.g., LAN))
 - global eavesdropper (observing traffic in the whole network)
 - internal attackers
 - (colluding) compromised elements of the anonymity system
 - communication partner



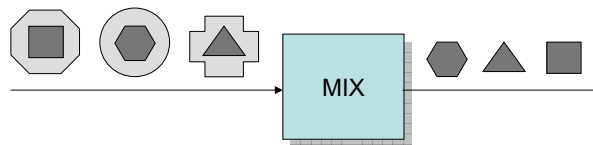
Anonymizing proxy

- application level proxy that relays messages back and forth between a user and a service provider
- properties:
 - ensures only sender anonymity with respect to the communicating partner (service provider does not know who the real user is)
 - a local eavesdropper near the proxy and a global eavesdropper can see both the sender and the receiver information
 - proxy needs to be trusted for not leaking information (it may be coerced by law enforcement agencies!)
 - even if the communication between the user and the proxy, as well as between the proxy and the server is encrypted, a naïve implementation would have the same properties (weaknesses)



MIXes

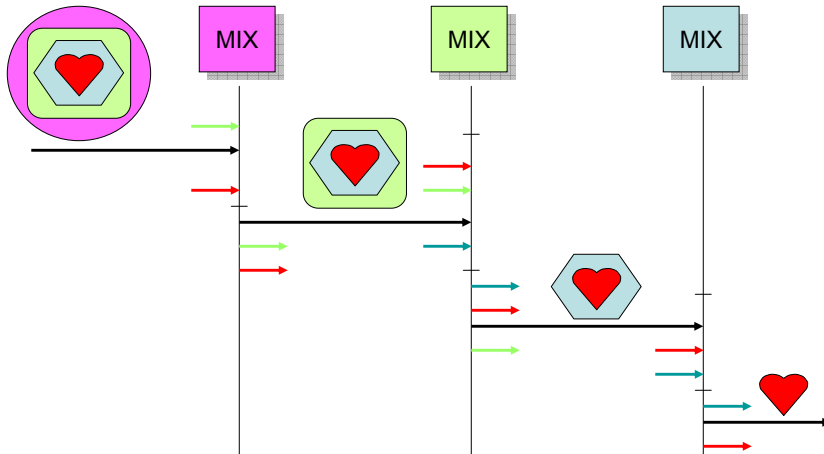
- a MIX is a proxy that relays messages between communicating partners such that it
 - changes encoding of messages
 - batches incoming messages before outputting them
 - changes order of messages when outputting them
 - (may output dummy messages)



- properties:
 - sender anonymity w.r.t. communication partner
 - unlinkability w.r.t. global (and hence local) eavesdroppers
 - the MIX still needs to be trusted

MIX cascade (network)

- defense against colluding compromised MIXes
 - if a single MIX behaves correctly, unlinkability is still achieved



Implementation of MIXes

- each MIX has an RSA key pair
- sender of a message m selects a path through the MIX network and encodes the message iteratively for each MIX on the path

$$\text{MIX}_1 \left| E_{\text{MIX}_1} \left(\text{MIX}_2 \left| E_{\text{MIX}_2} \left(\text{MIX}_3 \left| \dots \left| E_{\text{MIX}_n} (m \mid \text{rnd}_n) \right| \dots \right| \text{rnd}_2 \right) \right| \text{rnd}_1 \right) \right|$$

where $E_{\text{MIX}_i}(\cdot)$ is RSA encryption with the public key of MIX_i and rnd_i is some sufficiently large random number (salt)

- alternatively, a probabilistic encryption scheme (e.g., ElGamal or RSA with PKCS#1 formatting) could be used
- each MIX decodes the message with its private key and forwards it to the next MIX indicated in the decoded message
- message m may contain a “return address”
 - an iteratively encrypted structure, where layer i is encrypted with the public key of the i -th MIX on the return path and contains
 - the identifier of the next MIX on the return path
 - a secret key to be used for encrypting the content of the reply
 - layer $i-1$ of the return address

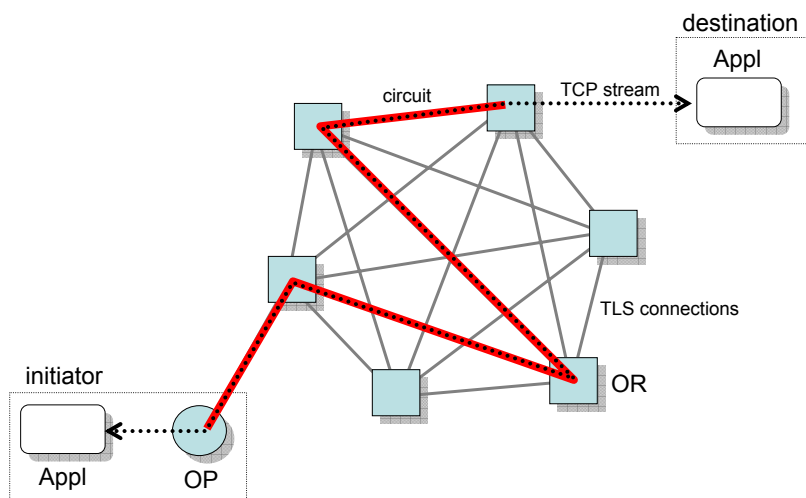


The Tor network

- Tor is a low-latency (real-time) mix-based anonymous communication service
- the Tor network is an overlay network consisting of onion routers (OR)
- ORs are user-level processes operated by volunteers in the Internet
 - a few special directory servers keep track of the ORs in the network
 - each OR has a descriptor (keys, address, bandwidth, exit policy, etc.)
 - each OR maintains a TLS connection to all other ORs
- users run an onion proxy (OP) locally, which establishes virtual circuits across the Tor network, and multiplexes TCP streams coming from applications over those virtual circuits
- the last OR in a circuit connects to the requested destination and behaves as if it was the originator of the traffic



The Tor network illustrated





Cells

- data within the Tor network are carried in fixed sized cells (512 bytes)
- cell types
 - control cells
 - used to manage (set up and destroy) circuits

2	1	509
CircID	Cmd	DATA

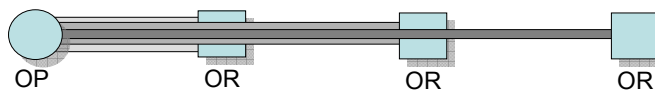
- relay cells
 - used to manage (extend and truncate) circuits, to manage (open and close) streams, and to carry end-to-end stream data

2	1	2	6	2	1	498
CircID	Rly	StreamID	Digest	Length	Cmd	DATA



Setting up a circuit

- circuits are shared by multiple TCP streams
- they are established in the background
 - OPs can recover from failed circuit creation attempts without harming user experience
- OPs rotate to a new circuit once a minute
- a circuit is established incrementally, in a “telescoping” manner
 - a circuit is established to the first OR on the selected path by setting up a shared key between the OP and that OR
 - this circuit is extended to the next OR by setting up a shared key with that OR; this already uses the circuit established in the previous step
 - and so on...





Establishment of shared keys

- Diffie-Hellman based protocol:

OP → OR: $E_{PK_{OR}}(g^x)$

OR → OP: $g^y | H(K | \text{"handshake"})$

where K is the established key g^{xy}

- properties:
 - unilateral entity authentication (OP knows that it is talking to OR, but not vice versa)
 - unilateral key authentication (OP knows that only OR knows the key)
 - key freshness (due to the fresh DH contributions of the parties)
 - perfect forward secrecy
 - (assuming that OR deletes the shared key K when it is no longer used)
 - if OR is later compromised, it cannot be used to decrypt old (recorded) traffic



Relaying cells on circuits

- application data is sent in relay cells
- OP encrypts the cell iteratively with all the keys that it shares with the ORs on the path (onion-like layered encryption)
- each OR peels off one layer of encryption
- last OR sends cleartext data to the destination
- on the way back, each OR encrypts the cell (adds one layer), and the OP removes all encryptions
- AES is used in CTR mode (stream cipher) → encryption does not change the length

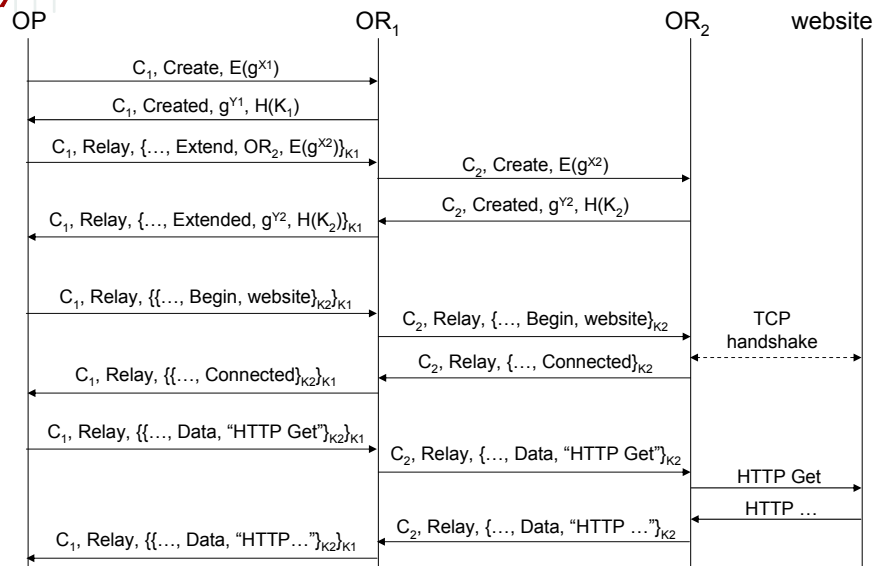


Opening and closing streams

- opening:
 - the TCP connection request from the application is re-directed to the local OP (via SOCKS)
 - OP chooses an open circuit (the newest one), and an appropriate OR to be the exit node (usually the last OR, but maybe another due to exit policy conflicts)
 - OP opens the stream by sending a “relay begin” cell to the exit OR
 - the exit OR connects to the given destination host, and responds with a “relay connected” cell
 - the OP informs the application (via SOCKS) that it is now ready to accept the TCP stream
 - OP receives the TCP stream, packages it into “relay data” cells, and sends those cells through the circuit
- closing:
 - OP or exit OR sends a “relay end” cell to the other party, which responds with its own “relay end” cell



Operation illustrated





Exit policies

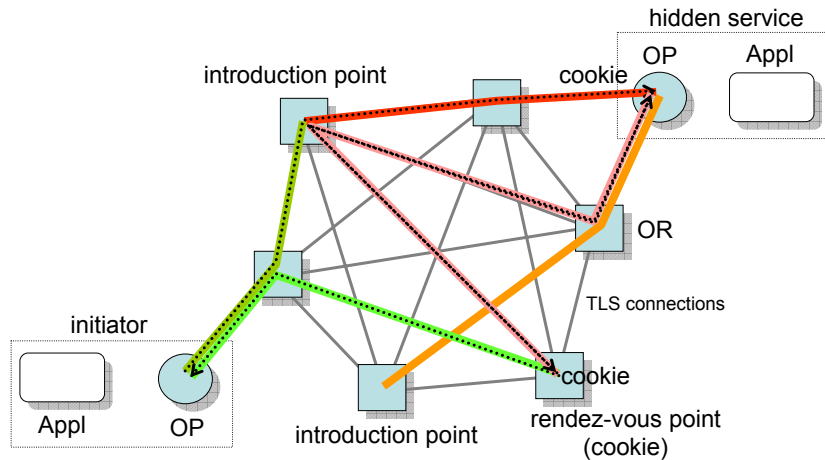
- hackers can launch their attacks via the Tor network
 - no easy way to identify the real origin of the attacks
 - exit nodes can be accused
 - this can discourage volunteers to participate in the Tor network
 - fewer ORs means lower level of anonymity
- each OR has an exit policy
 - specifies to which external addresses and ports the node will connect
 - examples:
 - open exit – such nodes will connect anywhere
 - middleman – such nodes only relay traffic to other Tor nodes
 - private exit – only connect to the local host or network
 - restricted exit – prevent access to certain abuse-prone addresses and services (e.g., SMTP)



Rendez-vous and hidden services

- Tor allows someone to offer a TCP-based service anonymously (without revealing his IP address to the world)
- the server's OP chooses some ORs as introduction points and builds a circuit to each of these introduction points
- the service provider advertises his service and the related introduction points by some anonymous directory service
- the client's OP chooses an OR as the rendez-vous point, builds a circuit to the rendez-vous point, and gives it a random rendez-vous cookie
- the client OP builds a circuit to one of the service introduction points, opens an anonymous stream to the server, and sends the cookie and a reference to the rendez-vous point
- the server OP builds a circuit to the given rendez-vous point and sends the cookie
- the rendez-vous point verifies the cookie and connects the client circuit to the server circuit
- the client establishes an anonymous stream through the new circuit

Rendez-vous illustrated



Some attacks

- end-to-end timing (or size) correlation
 - an attacker watching traffic patterns at the initiator and the responder will be able to confirm the correspondence with high probability
 - it was not the goal of Tor to prevent this
- website fingerprinting
 - an attacker can build up a database containing file sizes and access patterns for targeted websites
 - he can later confirm a user's connection to a given website by observing the traffic at the user's side and consulting the database
 - in case of Tor, granularity of fingerprinting is limited by the cell size
- tagging attacks
 - an attacker can "tag" a cell by altering it, and observing where the garbled content comes out of the network
 - integrity protection of cells prevent this



What is Tor good for?

- anonymous web browsing
- anonymous e-mail
- anonymous remote electronic voting
- ...
- carrying out malicious activity anonymously
 - we installed a Tor OR and ran it for 72 hours
 - we could observe 2216 sessions in clear (where our OR was the exit router) where a password was transmitted (most probably on-line password guessing attempts)
 - some statistics:
 - allotracker.com: 1285
 - yahoo.com 59
 - games.amil.ru 18
 - tracker.zamunda.net 140
 - nntp: 25
 - http: 1900
 - pop3: 286
 - other: 5



Outline

- introduction
- anonymous communication techniques
- **anonymity metrics**
- Query Auditing
- Private Information Retrieval (PIR)
- summary



Measuring anonymity

- anonymity set size
 - anonymity set: set of entities that could potential be associated with a given action (e.g., sending or receiving a message) by a given type of attacker
 - the larger this set is the more difficult is for an attacker to identify the relationship between the observed action and an entity
- entropy based anonymity metric
 - let p_i be the probability that entity i is responsible for an observed action (sender or receiver of a message)
 - uncertainty of the attacker is $-\sum p_i \log p_i$

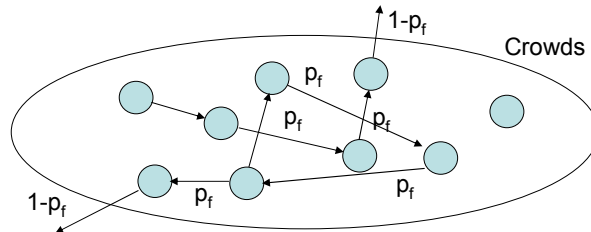


Degrees of (sender) anonymity

- beyond suspicion:
 - attacker can see evidence of a sent message, but ...
 - the sender appears no more likely to be the originator than any other potential sender in the system
- probable innocence:
 - the sender may be more likely the originator than any other potential sender, but
 - the sender appears no more likely to be the originator than not to be the originator
- possible innocence:
 - the sender appears more likely to be the originator than not to be the originator, but
 - there's still a non-trivial probability that the originator is someone else

Example: Crowds

- Crowds is a set of volunteers who run a special process called *jondo* on their computers
- each user sets his browser to use his jondo as a web proxy
- when the jondo receives the first request from the browser, it initiates the establishment of a random path of jondos in the crowd
 - the jondo picks a jondo (possibly itself) in the crowd at random, and forwards the request to it (after sanitizing it)
 - when this jondo receives the request it forwards it with probability p_f (to a randomly selected jondo again) or submits the request to the destination server with probability $1-p_f$



Sender anonymity wrt the end server?

- α – originator of request
- ω – jondo that submits the request to the end server

$$\begin{aligned} \Pr\{\alpha = x | \omega = y\} &= \frac{\Pr\{\alpha = x, \omega = y\}}{\Pr\{\omega = y\}} \\ &= \frac{\Pr\{\omega = y | \alpha = x\} \Pr\{\alpha = x\}}{\sum_z \Pr\{\omega = y | \alpha = z\} \Pr\{\alpha = z\}} & \Pr\{\alpha = x\} = \frac{1}{n} \\ &= \frac{\Pr\{\omega = y | \alpha = x\}}{\sum_z \Pr\{\omega = y | \alpha = z\}} \end{aligned}$$

$$\Pr\{\omega = y | \alpha = x\} = ???$$

$$\Pr\{x \rightarrow y \rightarrow SRV\} = \frac{1}{n}(1-p_f)$$

$$\Pr\{x \rightarrow i \rightarrow y \rightarrow SRV\} = \sum_i \frac{1}{n} p_f \frac{1}{n} (1-p_f) = \frac{1}{n} p_f (1-p_f)$$

$$\Pr\{x \rightarrow i \rightarrow j \rightarrow y \rightarrow SRV\} = \sum_{i,j} \frac{1}{n} p_f \frac{1}{n} p_f \frac{1}{n} (1-p_f) = \frac{1}{n} p_f^2 (1-p_f)$$

$$\Pr\{\omega = y | \alpha = x\} = \Pr\{x \rightarrow * \rightarrow y \rightarrow SRV\} = \frac{1}{n}(1-p_f) \sum_{k=0}^{\infty} p_f^k = \frac{1}{n}$$

$$\Pr\{\alpha = x | \omega = y\} = \frac{\frac{1}{n}}{\frac{1}{n}} = \frac{1}{n}$$



If user could submit the req immediately

$$\Pr\{\omega = y | \alpha = x\} = ???$$

$$\text{if } x = y, \text{ then: } \Pr\{x \rightarrow SRV\} + \Pr\{x \rightarrow * \rightarrow x \rightarrow SRV\} = (1 - p_f) + p_f \frac{1}{n}$$

$$\text{if } x \neq y, \text{ then: } \Pr\{x \rightarrow * \rightarrow y \rightarrow SRV\} = p_f \frac{1}{n}$$

$$\sum_z \Pr\{\omega = y | \alpha = x\} = 1$$

$$\Pr\{\alpha = x | \omega = y\} = \frac{\Pr\{\omega = y | \alpha = x\}}{\sum_z \Pr\{\omega = y | \alpha = z\}} = \Pr\{\omega = y | \alpha = x\}$$

→ sender is more likely to be the jondo from which the request was received, than any other jondo !



Outline

- introduction
- anonymous communication techniques
- anonymity metrics
- **Query Auditing**
- Private Information Retrieval (PIR)
- summary



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
- $X = \{x_1, x_2, \dots, x_n\}$ 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 $X = \{x_1, x_2, \dots, x_n\}$
 - a set of queries $Q = \{q_1, q_2, \dots, q_t\}$ and corresponding answers $A = \{a_1, a_2, \dots, a_t\}$
- 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 $Q = \{(ALL, MAX), (ALL, SUM)\}$
 - assume that $A = \{5, 15\}$
 - $MAX(x_1, x_2, x_3) = 5$ and $SUM(x_1, x_2, x_3) = 15$
 - 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 $X = \{x_1, x_2, \dots, x_n\}$
 - a set of queries $Q = \{q_1, q_2, \dots, q_t\}$ and corresponding answers $A = \{a_1, a_2, \dots, a_t\}$
- 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 $O(n^2|Q|)$)
 - 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 $O(n^3 + n^2|Q|)$
- 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 queriesover 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 $X = \{x_1, x_2, \dots, x_n\}$
 - a set of queries $Q = \{q_1, q_2, \dots, q_{t-1}\}$ and corresponding answers $A = \{a_1, a_2, \dots, a_{t-1}\}$ 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 $(Q' \cup \{q_i\}, A' \cup \{a_i\})$
 - if some data is disclosed, then deny response, otherwise return a_i
- this does not work in general, because denials also leak information!
- example:
 - let $n = 3$ and $X = \{5, 5, 5\}$
 - let $Q = \{(ALL, SUM)\}$, then $A = \{15\}$
 - let $q_2 = (ALL, MAX)$
 - this is denied (see previous example on slide 4)
 - however, one can still figure out that all x_i 's are 5
 - $MAX(x_1, x_2, x_3)$ cannot be smaller than 5, otherwise the SUM cannot be 15
 - if $MAX(x_1, x_2, x_3) > 5$ then the query would have been answered
 - $MAX(x_1, x_2, x_3)$ 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 $Q \cup \{q_t\} = \{q_1, q_2, \dots, q_t\}$ and $A = \{a_1, a_2, \dots, a_{t-1}\}$, 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 $X = \{x_1, x_2, x_3\}$
 - $q_1 = (\text{ALL}, \text{SUM})$ can be responded
 - then $q_2 = (\text{ALL}, \text{MAX})$ 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, \dots, q_t and a_1, a_2, \dots, a_t is said to be λ -safe with respect to a data element x_i and an interval $I \subseteq (-\infty, \infty)$ if

$$1/(1 + \lambda) \leq \frac{\Pr_D(x_i \in I | q_1, \dots, q_t, a_1, \dots, a_t)}{\Pr_D(x_i \in I)} \leq (1 + \lambda)$$

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 AllSafe as follows:

$\text{AllSafe}_\lambda(q_1, q_2, \dots, q_t, a_1, a_2, \dots, a_t) =$

1, if $q_1, q_2, \dots, q_t, a_1, a_2, \dots, a_t$ is λ -safe for all i and every I
0, otherwise



Definition of partial disclosure

- (λ, 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 $a_t =$ “denied” otherwise
 - the attacker wins if $\text{AllSafe}_\lambda(q_1, q_2, \dots, q_t, a_1, a_2, \dots, a_t) = 0$
- an auditor is (λ, δ, 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



Outline

- introduction
- anonymous communication techniques
- anonymity metrics
- Query Auditing
- Private Information Retrieval (PIR)
- summary



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: $X = x_1 x_2 \dots x_n$
 - 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)
 - 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
 - there's hope if the database can only perform efficient computations (i.e., it is computationally bounded)
 - 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} ($1 \leq i, j \leq n^{1/2}$)
- 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}



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 un-limited 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 $x^2 \bmod m = a$
 - 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} ($1 \leq i, j \leq n^{1/2}$)



An example CIPR protocol

- protocol:
 - Alice chooses at random a large integer m (together with its factorization)
 - she generates $n^{1/2}-1$ random QRs mod m : $a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots$
 - she generates a random QNR mod m : b_i
 - Alice sends $a_1, a_2, \dots, a_{i-1}, b_i, a_{i+1}, \dots$ 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, \dots
 - for each column c of X , the database computes
$$v_c = u_1^{x_{1c}} u_2^{x_{2c}} \dots \text{ mod } m$$
 - the database responds with v_1, v_2, \dots
 - 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?

$$X = \begin{array}{|c|} \hline x_{11} \dots x_{1j} \dots \\ \hline \dots \\ \hline x_{i1} \dots x_{ij} \dots \\ \hline \dots \\ \hline \end{array} \quad U = \begin{array}{|c|} \hline a_1 \\ \hline \dots \\ \hline b_i \\ \hline \dots \\ \hline \end{array} \quad v_j = a_1^{x_{1j}} \dots b_i^{x_{ij}} \dots \text{ mod } m$$

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



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 ($O(n^\epsilon)$) and the ϕ -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, δ, ϵ) -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 δN bits of the codeword are corrupted
- local decodability comes at a price of loss in terms of code efficiency ($N \gg n$)
- 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 δN bits of y are corrupted \sim each bit in y is corrupted with probability δ (independently from the other bits)
 - the probability that y_t or $y_{t'}$ is corrupted is 2δ
 - 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, \dots, q_k , such that x_i can be recovered from $C(X)_{q_1}, \dots, C(X)_{q_k}$, 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

- we saw some examples on how technical approaches can help to retain privacy in IT systems
- anonymous communication techniques used over the Internet
 - concept and general operation of MIX networks
 - engineering details of a practical anonymity system (Tor)
 - examples for anonymity metrics
- 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