Summary Cybersecurity

# Macrodata Disclosure techniques

## Table of counts or frequencies

* **Sampling**: estimate made by multiplying individual responses by a sampling weight before aggregating them.
* **Special rules**: define restriction on the level of detail that can be provided in a table.
* **Thresholds rules**: Cell is sensitive if the number of respondents is less than some specified number.
  + Table restructuring and category combination
  + Cell suppression:
    - Primary suppression
    - Complementary suppression (Audit Techniques and linear programming)
  + Rounding:
    - Random: random decision on cell values will be rounded up or down.
    - Controlled: ensure that the sum of published entries is equal to published marginal totals.
  + Confidentiality edit: Microdata files applies switching

## Table of magnitude data

* **Suppression** **rules**:
  + **p-percent rule** : celli s sensitive if lower estimates for the respondent’s value are closer to the report value than a percentage p.
  + **pq rule** : value q represents how is the prior knowledge value before publication (p < q < 100)
  + **(n,k) rule** : if a small number of cell ( <= n) of these respondents contribute a large percentage (>= k%) the cell is considered sensitive.

Primary rule: identify sensitive cell

Secondary rule: restructure the table and collapse the cell or cell suppression

# Microdata Disclosure techniques

Microdata protection techniques can be classified in two main categories:

* Masking techniques
* Synthetics data generation techniques

## Masking Techniques

Original data are transformed to produce new data that are valid for statistical analysis and preserve the confidentiality.

They are classified as:

Non-perturbative: original data are not modified, but some data suppressed

Perturbative: original data are modified.

* **Sampling**: Protected microdata table is obtained as a sample of original microdata table.
* **Local suppression**: suppression of some attribute values (sensitive cells that are likely to contribute to the disclosure risk of the tuple.
* **Global recoding**: replace values of the attribute with the label associated with the corresponding interval
* **Top-coding:** define upper limit called “top-code” for each attribute to be protected, any value > to this value is replaced with the top-code.
* **Bottom-coding:** as top but any value < to this value is replaced with “bottom-code”
* **Generalization:** representation of a value with a more general value.
* **Random Noise:** perturbs a sensitive attribute by adding or multiplying by a random variable.
* **Swapping:** non-matched values of the variables are swapped between the two records.
* **Micro-aggregation (blurring):** grouping individual tuples into small aggregates of a fixed dimension k.

## Synthetic Techniques

Original and synthetic data should present the same quality of statistical analysis.

# Privacy in data publication

Different types of disclosure:

* **Identity disclosure**: a third-party can identify a respondent from the released data
* **Attribute** **disclosure**: confidential information about a respondent is revealed (exactly revealed or closely estimated)
* **Inferential disclosure**: information can be inferred from statistical properties of the released data.

**Confidentiality** can be protected by: restricting the amount of information released (restricted data), imposing conditions on access to the data products (restricted access), combination of these two.

**Anonymity problem**: de-identification (remove explicit identifier) is not sufficient. Linking identities with de-identified information led to re-identification.

Classification of attributed in microdata:

* Identifiers: attributes uniquely identify microdata respondent
* Quasi-identifiers: attributes in combination identify or reduce uncertainty over microdata respondent
* Confidential: attributes that can contain sensitive information
* Non confidential : attributed that respondent do not consider sensitive

Factors for **disclosure risk** : Existence of high visibility record (for VIPs) and matching data with external information.

Factors **reducing risk:** microdata table are subset of population, information is not always up-to-date.

## K-anonymity

**Goal**: protect respondent identities while releasing truthful information.

**How**: released data should be indistinguishably related to no less than a certain number of respondents.

k-anonymity requires that each quasi-identifier value appearing in the released table must have at least k occurrences.

It can be used together with generalisation (values substituted with a more general values) and suppression (removing sensitive information).

Generalization and suppression addition info...

**Algorithms** for computing a **k-anonymous** table

Minimal k-anonymous table is NP-hard problem 🡺 use of hierarchies or heuristics.

* **Incognito Algorithm**
* **Mondrian multidimensional algorithm**
* **K-anonymity revisited**

## Attribute disclosure

All tuples with a quasi-identifier value in a k-anonymous table may have the same sensitive attribute value.

**ℓ-diversity:** an adversary needs to eliminate at least ℓ-1 possible values to infer the respondent.

Tuple T is ℓ-diverse if all blocks are ℓ-diverse:

* Homogeneity attack not possible
* Background knowledge more difficult

BUT : ℓ-diversity leave space for attack based on distribution of values inside blocks (skewness and similarity attacks)

* **Skewness attacks:** occur when the distribution in a q-block is different than the original population
* **Similarity attack :** occur when q-block has different but semantically similar values for the sensitive attribute.

# Differential privacy

Aims at preventing adversaries from being capable to detect the presence of absence of a given individual in a dataset.

**k-anonymity vs differential privacy:**

* K-anonymity:
  + Nice capturing of real word requirement
  + Not complete protection
* Differential privacy:
  + Better protection guarantees
  + Not easy to understand

**….**

# Authentication and Access Control

Security strategies:

* Prevention: take measure that prevent damage
* Detection: take measure that detect
* Reaction : take measure to recover your system

Security objective:

* Confidentiality: prevent unauthorized disclosure of info
* Integrity: prevent unauthorized modification of info
* Availability: guarantee info are available

## Authentication

Provide the system to:

* Identification of the user
* Authentication to be certain of the identity presented (necessary for access control and security logging)

Cryptography: transform cleartext into a non-intelligible text. It is based on the use of a key to decrypt and encrypt 🡺 symmetric and asymmetric encryption.

Authentication can be based on something user :

* knows (password)
  + Vulnerability of password: easily guessing, read by people, acquired by third-parties.
* has (token)
  + Vulnerability of token: identify only token, not the user
* is (biometric)

## Access control

It evaluate access requests to the resources by the authenticated user based on some rules: direct access or inference.

Studying access control is important:

* policy: define guidelines
* model: define the access control specification
* it implements policy via low level function

Approach for describing protection system:

* access matrix: (S, O, A)
* authorisation table: store table of non-null triples (subject, object, access)
* Access control list (ACL): stored by column. Require auth of sub, provide access control and revocation per object
* Capability list (tickets): stored by row. Not require auth of sub, provide access control and revocation per subject

**Policies**: define who can access the resources

* **Discretionary (DAC)**

Access using: Identity of the requestors and explicit access rule. User can be given the ability of passing on their rights to the user.

CONS: vulnerable from Trojan horses (hidden code performs function not know to the caller)

* **Mandatory (MAC)**

Impose restriction on the information flow which cannot be bypassed by Trojan Horses. Makes distinction between user (human) and subject (process in the system).

CONS: controls only overt channels (covert channels are vulnerable)

* **Role-based (RBAC)**

Role named set of privileges related to execution of a particular activity. Role grant auth for access object, user to activate roles.

PRO: easy management, role hierarchy, separation of duty.

* **Credential-based:** Attributes proved by presenting certificates
* **Attribute-based (ABAC):** Authorisation defined on attributed of the requester

# Privacy in Data Outsourcing

## Smart devices

PRO: better protection, business continuity and disaster recovery, prevention

CONS: more complexity, explosion of damage and violation, loss of control over data and processes.

## Cloud Computing

Allow user to rely on external provider for storing, processing, and accessing their data.

CONS: user do not control their own data.

## Technical challenges

Security properties:

* Confidentiality
* Integrity
* SLA compliance (certification and assurance)

Access requirement:

* Data archival
* Data retrieval
* Data update

Architectures:

* 1 user -1 provider
* N users - \* providers
* \* users – n providers

Challenges in data protection:

* Secure and private data computation
* Privacy and integrity of queries
* Privacy and integrity of data storage
* Privacy of user

# Privacy and Data Protection in Emerging Scenarios

Solution for protecting data can be based on:

* Encryption
* Encryption and fragmentation
* Fragmentation

## Encryption

Server can be host but curious and should not have access to resource content.

* **Keyword-based searches**
* **Homomorphic**: operation directly on encrypted data
* **Onion Encryption**: onion layer each of which support a specific SQL operation
* **Indexes:** indexes associate with attributes are sued by the server to select the data to return as response.

Indexes for queries:

* Direct(1:1)
* Bucket (n:1)
* Flattened (1:n)
* Partition-based: map to a certain partition
* Hash-based index: one-way hash function
* Interval-based queries: use B+-tree

**Inference exposure**

Two conflict requirement in indexing data: index should provide effective query execution, should not open door to inference attacks. It is important to measure the exposure coefficient.

Possible Inference:

* Freq+ DBk = correspondence between plaintext values and indexes based on the number of occurrences.
* DB + DBk = determine correspondence between plaintext and indexes.

**Bloom Filter**

At the basis of the construction of some indexing techniques:

* N elements
* Vector of l bits
* h independent hash function Hi : {0,1} 🡪 [1,l]

Insert and search element.

PRO: space efficient, acceptable in practical application

CONS: false positive

**Data integrity**

We need integrity in storage and in query computation. Relies on digital signatures that are usually computed at tuple level.

Verification cost grows linearly with the number of tuples in the query result.

**Selective Encryption**

Different user might need different views of data. Access control policy require data owner to mediate access 🡺 impractical

* Attribute based on encryption (BAE): allow key only user with certain attribute
* Selective encryption: auth policy defined data owner is translated into encryption policy

**Over-Encryption**

Resources are encrypted twice by the owner (Base layer) and the provider (surface Layer). It need support by the server.

**Mix for policy Revocation**

Mixing resources using different rounds of encryption: unavailability of a small portion of the encrypted resource prevents its construction.

**Slice for policy Revocation**

Slice the resource into fragments and re-encrypt a randomly chosen fragment: lack of a fragment prevents resource decryption.

## Encryption and Fragmentation

Find minimal number of fragments is NP-hard problem: select attribute with the highest number of non-solved constraints and insert it in an existing fragments that violate constraint.

**Data fragmentation:**

* **Non-communicating pair of servers**
  + Confidentiality are enforced by splitting information into two independent servers that cannot communicate (sensitive info are protected by distributing attributes among the two servers)
* **Multiple non-linkable fragments**
  + Allow for more than two non-linkable fragments
* **Departing from encryption**
  + Involve the owner as a trusted party to maintain a limited amount of data
* **Fragmentation and inferences** 
  + Find minimal fragmentation using metrics such storage or computation traffic.

## Combining Indexes, Selective Encryption and Fragmentation

Combining all these technique may open door to inferences by users.

* Access control and indexes :
  + Selective encryption for access control with indexes for query execution.

Cell with same plaintext values are exposed

* Indexes and Fragmentation
  + Provide effectiveness and efficiency in query execution but can cause info leakage of confidential information

Open issues: protection against observation of accesses to fragments, protection against the release of multiple indexes …

## Protecting data accesses

* Access confidentiality:
  + Confidentiality access aim at a specific data
  + Pattern confidentiality: confidentiality of two accesses aim at the same data
* **Path ORAM:** 
  + Server side tree structure with L level and N blocks. Each node is a bucket that contains Z real blocks
  + Client side: locally stores a small number of blocks in a stash and store a position map. Position map changed every time blocks are accessed and remapped.
* **Ring ORAM**
  + Path ORAM that reduces the online access bandwidth to O(1).
  + Same server of classic ORAM but each node has additional dummy block, small map of the offset a counter of access.

**Ring ORAM vs Path ORAM:**

* Same protection as path ORAM
* Much more efficient
* Limited access time
* Shuffle Index

CONS: range queries not supported, accesses by multiple client are not supported, vulnerable to failure of the client.

**Shuffle index**

Data are indexed over a candidate key K and organized as an unchained B+-tree with fan out F.

Logical representation: set of pair <id, n> with id the node identifier and n the node content. Each node on the server in encrypted form.

Client at each iteration decrypts the retrieved block and determine the block to be retrieved. Process end when a leaf block is retrieved.

Server received a set of block to store. Server receive request to access the block that translate into observations. Server should not be able to infer data stored (content confidentiality), data which access request are aimed (access confidentiality), aim at the same node (patter confidentiality).

Is encryption enough?

PRO: it protects content confidentiality and access confidentiality

CONS: access and patter confidentiality is not provided 🡺 frequency-based attacks allow the server to reconstruct the correspondence between plaintext values and block.

Combine three strategies:

* Cover searches: provide confusion on accesses
* Cached searches: allow protection of accesses to the same values
* Shuffling : change node location at every access.

# Integrity in query computation

Integrity for query result:

* Correctness
* Completeness
* Freshness

Two approaches:

* Determinist: uses auth data structures
* Probabilistic: insertion of fake tuples.
* Merkle hash tree:
  + Binary tree where each leaf contains the hash of the tuple
  + Each internal node contain the result of the hash of the concatenation of its children.

Probabilistic approach:

* Encryption: makes data unintelligible.
  + On-the-fly encryption
* Markers: fake tuples not recognizable
* Twins: replication of existing tuples.
  + Twice as effective as marker but lose effectiveness on large fraction of tuples omitted.
* Salt: replication with salt
  + Map different occurrences of the same join value on the side of the join to a different encrypted value sing different salt
* Bucket: replication with bucket
  + Dummy tuples guarantee flat frequency distribution of join attribute values.

**Map reduce**

Framework supports execution of task over large amount of data in parallel by multiple nodes (worker), coordinated by a manager.

* Map function translate the input tuples in a set of pair <key, value>.
* Function f assign pair to worker
* Reduce function executed by each worker and result combined by manager
* Markers and MapReduce:
  + Marker should be distributed among all workers:
    - Random
    - At-least-n
    - Perfect balance
* Twins and MapReduce:
  + Control Twin generation like for markers is impossible
    - Server twin different tuple
    - Server observe different number of twins
    - Server cannot coordinate to regulate twin distribution