Notes un po' alla cazzo

GLV

1 Security for Cloud computing

Releasing data collected from a huge horde of people, public realse and important analysis. Public manner without restriction to access of anytype. Protection, in an "who access" prospective. How to protected data putted inside clouds. Data integrity and confidentiality. A lot of seemngly important stuff here.

1.1 Macrodata and Microdata Protection

Specific case, hospital data to sell to a pharm.co. The control over the infos will be lost after the full publication or the sell. The agent will be free to do whatever he wants with the data after the change-event, whatever it is. The importance of preventive protection to avoid a specific thing is not desirable, or even dangerous or illegal. Planning is vital, as most of time. Risk awareness, interesting application from economics here towards agents who take protection-related choices and information asymmetry. Two kinds of way to release infos, Micro and Macro. A big historic examples is statistical information released publicly, Istat, Eurostat, whatever. Disclosure and its risk. Behind the collected data there are people, and we have to avoid that someone is able to retrieve the original data, exposing people that have a disease, especially a socially problematic one, like an STD. Privacy is a right that we have to respect. if this information are available elsewhere, combined, form private issue, that is something to prevent and avoid, like eliminating potential key variables, even if this is a loss of information. DBMS or just statistical data. DBMS control using permission and queries. Statistical data, Istat like, not direct data or assets are given. Statistical DBMS, give access to perform analysis over collected data. No revealing of a single individual. Denied he queries through permission. Confidential can be reach with combination for multiple queries and external knowledge. We should avoid this, but its difficult, especially external knowledge. Independently from external knowledge, cause we have no control and is constantly updated. No clear solution here. Simple example, where the setted condition of the query reduce to just one individual, that is a problem. Result focus approach, eliminate sensitive query who reveal single individual data. Set a threshold, k related to the dataset dimension(Golden rule). Cleaver is a must, keeping track of the different queries to avoid sensitive results. Macrodata—¿results of computation performed. Micro—¿the opposite, clean data ready to use. Now is possible to analyse directly microdata, even if large. A 10 year technological lap made this possible. The importance and the flexibility of raw data, that after some cleanse from microdata. Trade-off utility (who is going to be charge) vs. privacy, a failure in this can cause long-run damage to the business, starting with reputation issue. Macrodata seems to be self protected but can be still used to get more precise and personal information. Through combination or just simple a table composed of a large sum of individual features. Reverse engineering is a possible no matter the dimension we are working with. Macrodata of two types, count/frequency, of an attribute. Magnitude data is an aggregate of some characteristics. Techniques used for Macrodata, counts and frequencies cases are protected in a different way from magnitude. Inference through peculiar features of a table, like having only one cell of a row empty. Sampling is a protection technique, quite common and easy. Just publishing a representative sample of the complete dataset. Perhaps the person we are looking to violate, is not included or represented. Not knowing the size of the sample, thus the rescaling, is a simple way to protect data. Encryption is an example of that. As long as the parameters used are kept secret we are safe. Special rules usable everywhere, but more used for whole population data rather than sample. Geographical details are an example field of application. Special rules, SSA example, no single cell in row allowed, or where a single value of a cell allow to determine earning, age and benefits within a certain interval, this interval is establish by law. To protect from this we can reorganize rows so to respect the special rule, by lesser rows or columns. Seems that laws decide unacceptable intervals of potential inference, the acceptable threshold is setted sector by sector (geographical is treated differently from income), identifying information are not allowed and dates of birth is rounded to the year only, larger if age is higher than 90. Simple works well enough surprisingly. Threshold rules, more general rules, that makes it safer. Different approach and techniques to protect sensitive cells by restructuring, suppressing, random rounding, confidentiality edit and control rounding. Confidentiality edit protect from microdata not macrodata, so the results are self protected. Columns and row approach, with only primary suppression does not resolve all of the cases, and we have more than one table the security have to imposed with further limit so o avoid different crossing relseases. Supress additional cells to solve the problem. We should also avoid that through system of equation is possible ro solve and get one by one the data that were original suppress. Audit techniques to check if we are really protected, software that try to get the suppressed cell using computation. Rounding is another technique. Random decision on whether cell is going to be rounded up or down, the consequence of this is that the sum will be different from the published marginal totals. Control rounding avoid this so that entries and marginal totals are equal. Granularity is respected of course even in rounding. Control rounding may not exist mathematically. Linear programming methods are used to identify a controlled problem. Confidentiality edit starting protection procedure on the microdata so that any (macro) table obtained from the protected micro is protected as well. Applicable both to the entire microdataset or just sample of it. The idea is switching the tuple. Sampling n records, than matching on a specific set of attributes but from a different geographic area and the switching the two tuples. (quite smart and easy). Set a % to switch and the n-attributes valuable for the matching, this depends on what we are looking for, the desired table etc etc.

Protection of Magnitude Data, these data have different peaks, based on frequency. The uncommon values, outliers are problematic for the people behind these values. Eliminating the outliers is not the solution, is changing too much the dataset, thus the infos. How to protect the sensitive outliers, the previous methods do not work, since the sensitivity is based on contribution to data not frequency or count. First of all establish when is sensitive, so when the outliers is not enough. There are three common suppression rules pp, pq and nk. PP, p-percent, disclosure of nagnitude if the inference is accurate at most 10% (p%). If the more accurate inferences are possible, that is a problem, so we have a lower p. $\sum Ni = c + 2x_i \geq \frac{p}{100}x_1$. c is the size, x_1 is our outlier. Lower range in abs term is better, protection "sinergy" if the most sensitive is protected. P is computed case by case of course setting a c arbitrary. The idea if c individual put together their knowledge and can infer the attribute of another. Another technique is pq rule. PQ is similar to the pp rule, but pp does not take into account external previous knowledge, PQ does. How much the new table increase the information and what the new information brings, in term of potential inference. $\frac{q}{100}$ is added to the previous rule. NK is another technique, way more common the pp and pq. NK is simpler. A table in which n or fewer contributes to $k\%$ or more, in sensitive it should not be published.

2 Lesson N.2

Secondary suppression, restructure, collapse cells or cell suppression, blanking them. Marginal total have to be respected otherwise our datable will not be considered useful or trustworthy. limit the ability of obtaining through inference personal data of a respondent, if not total blocking the possibility, just limiting up to a certain level, p or one of the other methods discuss before. Union of suppressed cell is still sensitive. Secondary suppression, using automating technique, usually is not done automatic, cause otherwise fundamental cells will be eliminated. The machine aim to maximise protection with as few cells as possible, and mostly this cell are fundamental for our buyer or whatever the final user is. Audit technique to check if the protection is sufficient for the table, basically roleplaying the attacker, like a white hat hacker. Once released there is no control over them. Suppressing, combining, whatever is removed infos, so is information loss, and is measurable in many different ways according to the related theory. Keep confidential the parameter values used in the suppression rule applied. Solving a system of linear eqs basically if we have the parameters. Magnitude table include counts and frequencies, so that has to be taken into account when applying protection tables technique.

2.1 protection of microdata

Being able to protect microdata will protect also whatever macrotable we build from it of course. Is a recent think since there was not enough computational power to make this approach mainstream. High flexibility of micro, availability is also a positive quality. Anonymity is just removing explicit identifiers. Is also necessary to avoid the possibility to build and infer the identity. Micro has much higher risk than macro of course. Quasi-identifying attributes are a think to control, since you can restrict of total identification of subjects, with enough quasi-identifying data. Putting together, income, race, zip code, whatever and you get the the individual. Combining two or more source of infos with many quasi-identifying can get you to get the identity of a specific individual. To solve this reducing and changing the data is a necessity. A join with some public available infos and a microdata table, you get the identity of a % of the people part of the microtable. There is a strong difference in sensitivity between micro and macro, it's crystal clear. Sampling is the first method to apply, especially in microdata, so to give a representative sample to the public, thus useful, but controlling it at the same time, reducing the risk greatly. Of course identifiers are also removed, limiting geographic details and limit the number of vars itself. Too much vars increase the risk the error chance and the possibility of more possible join with already public infos. Geographic details often appears in microdata, since they are used a lot to collect data. But is also useful to identify people, quite powerful in that regard. (Logically) From the 80s' there are limitation to avoid excessive geographic details, so to avoid inferences and recognition. (Person per area limit). Also contextual vars are a thing to take into account, extra steps but still useful to recognize the geographic area, like temperature, presence of mountains, etc etc. Classification of techniques use for microdata protection. Masking techniques and synthetic data generation techniques. Masking reduce details or noise-injecting so to increase protection. Synthetic generate a model based on the data collected, and then the model generate samples that will be published. There is no correspondence with people living in the real word. (quite smart, I like this). They work in total different manner. Another classification is based of the var type. Continuous and categorical require different approach. The domain here is the difference. Masking techniques there is another difference between perturbative and non-perturbative. perturbative modify infos a bit, non-p does not. Perturbative seems extreme and a easy solution.

Sampling is the first approach, the most common technique in absolute term. Of course is necessary to keep the sample representative of the population. Of course is not enough to protect everyone, also outliers exist for a reason, they are useful. Local suppression of attributes that are risky. Sensitive cells means infos loss, so we should beware of how much we are decreasing the infos. The ideal is an outliers that is also useless for the final user. Of course removing attributes that are unique in a category it does not reduce as much as we should aim at the risk. Global recoding disjoint intervals and label replacing, pushing towards to an higher level of abstraction. Like dividing the income in 3 intervals and applying the label. This is not perturbative. T-coding or B-coding is defining a upper or lower limit to avoid easy gate to identification. Disclosure is something to avoid anyway is possible. More general values instead of the collected ones. Hierarchy of generalization is applied to the vars in a different way, is a substitution(Non-perturbative as well). Combining different set of values, removing digits, it varies. More frequent same values. Random noise is pertubarive, suited for continued attributes. a rule is applied, noise addition of multiplication. Noise rule can be made public, but is safer to not do so. Noise table represents what was done, the objecting is to maintain properties of the data, but the more you maintain the higher is the risk. Swapping is also a perturbative technique, is similar to micro-aggregation. Matching and swapping the different vars. Is necessary to choose the matching rule, the similarity necessary to swap, what to swap etc etc. Micro-aggregation, is still masking and perturbative. Form a group of tuples into smalls aggregate of fix k dimension. The groups are then averaged and that is data that will be public. Synthetic techniques are the different case. Build a statistical model, the model will produce data etc etc. There are fully and partially synthetic techniques, also some techniques works only on continuous vars, some only on categorical, some on both.

2.2 Privacy and data publication

Going beyond to the techniques saw until now, but when we did enough so to publish a table? Where draw the safe line is what we should clearly know. First addressing the problem of infos disclosure, inference to obtain infos about a specific subjects, identity disclosure through released data. Sensitive information (attribute disclosure) revealed through data, so associate the identity with personal and sensitive infos. Inferential disclosure is when there are no record referring the respondent, correlation between what I'm releasing and other attributes, infos in other tables, common knowledge etc etc. Putting together different infos from different tables, and correlation to get attributes behind the publish vars. Microdata table are way more harmful, in Macro identifying does not reveal attributes of the subject, while Micro does, with all the vars in the tab. Inferential disclosure, correlation is the an incredible easy and efficient weapon to achieve attribute disclosure with proxies. Statistical disclosure limitation methods depends on the nature of the data products and the how much we should protect. Explicit identifiers, first thing to cleanse. Restrict data based techniques or restrict classes techniques. Data reduce infos, the amount of it. Access-restriction is not a public/sell relationship to the final user, you limit and classify the user accordingly to which data are will be able to see and interact with. Anonymity problem, privately know records is increasing day by day. De-identification and anonymity are two different things. De-identification is just the first step towards anonymity. (GDPR data+additional knowledge should not give the identity back, otherwise the tab is not anonymous). Identity disclosure is the first to achieve, is necessary but not sufficient. Attributes have to be classify in 4 groups, id, quasi-id, confidential, n-confidential. The table set the attributes class, tab by tab, there are no abs. (Candidate key should be not eliminated basically). Existence of high visibility records, outliers are easiest way to identify something, always exposed, so if they are protected everyone is basically. How much matching is possible before publishing, and mow much information is possible to extract with those join? Fundamental questions here. External factor that increase the risk, have to be taken into account, like the number of outside sources, common attributes, the accuracy or resolution of the data themselves. Timing of the data collected has to be considerer, in time more sources will appear, increasing the risk. Also infos might change overtime, that will generate join error, cause people divorce, move, change, whatever. Noise are a thing that exist, human error, desired or not, has an impact of join possibility. Measuring the risk is fundamental. Probability approach, probability of the same individual in many different sources. The probability of matching vars in different sources. The probability of the respondent to be an outlier, unique etc etc, like being the pope, the queen, whatever, uniqueness. To measure the level of protection, a threshold, different way and concept are used and introduced. A threshould for every tuple k-individuals that can correspond to the tuple that will form the publisher tab. k2k correspondence in the real world with the various attributes. Assumption on our knowledge, worst scenario, safer manner, considering whatever is accessible right now when I'm publishing. Att have domains, generalization reduce this domain and execute a mapping. The maximal element of the domain–¿singleton. Value generalization hierarchy is a tree. Larger the tree, higher generalization steps, of course. K-anon is of course a requirement to satisfy, the generalization is done to satisfy the k-annon. Generalization through suppression (tuple suppression). The link between generalized and original still exists, and is possible to apply more than one generalization step. If this is not true, is due to suppression. Check overzip, seems to be suppression of the zip code, but the entire row is eliminated, probably based of ZIP differences. What is better between suppressing or generalization? Top most element of in the hierarchy is equivalent to suppression. So a subset representation is possible. Complete infos loss vs a reduction of infos that is still pretty useful. Unique tuple are better removed that generalize so to protect them, is the strategy that argmin the infos loss. Do not overdue, otherwise the utility is strongly reduced. Distance vector is a possible method tu apply. Putting the square brackets the level i of generalization apply to x_i. Of course there is an upper limit of potential generalization. Cross-graph generalization possibility decrease the risk. Probably having more coss generalization of a lower level is better that few of an higher level. k-min generalization. K-annon has to be respected with minimal required suppression. If k-annon is satisfy, the one with more tuple is preferred, of course have more infos then the one with less tuples. Is not possible to compare tabs from different branch not linked to one another. 1-ann cause no suppression and one tuple is enough to identify, high risk. Moving different paths of hierarchy, different generalization, more than one min, but there are differences. If we include sup, putting a limit to it. There a limit to generalization now, eliminating the singletons. First we apply generalization, the suppression is apply respecting the threshold. Logically, after gen singleton standout and are easy to see and suppress. Criteria based on what we want to achieve, higher difference, minim relative or absolute distance. Min suppression. Dependence on the table we have, the size, the dimension, and what we want to do with it. Granularity of choice set possible suppression and generalization. Uniform representation can be ignored to achieve min info loss. Computational hard problems as usual. Incognito alg, not recent, but quite used. Necessary but not sufficient is k-ann satisfy for any proper subset. Comminatory computation used to eliminate not respected k-annom. Checking the bottom of the hierarchy, if satisfy k-ann, all the general/step forward will be also ok with k-ann. Mondrian multidimensional alg, works with cells directly. We achieve so that we put in a specific region a n tuples of the same type. A split will case a point to belong to more than one region. 1 Tuple—¿ 1 respondent, always one true, more than 1 through generalization etc etc etc. Less strict condition on cell level if we respect k-ann. k-anno with cell generalization using one or more quasi-identifier. K-ann is for identity disclosure protection. Attribute disclosure is also a thing, and we should deal with it. K-annon is not enough for ATT disclosure. We can use homogeneity, all tuples x have the same values, so if we know someone belonging to the group, we will learn his/her sensitivity ATT. using l-diverse we avoid homogeneity, l-diversity is monotonic with respect to general hierarchies. Thus major step of generalization on a monotonic l-divers will keep the l-divers properties. Maximize diversity inside the tuple merging is the best case scenario. K-ann can be used to enforce l-diverse by adding additional condition inside the groups verification. L-diverse can leave to attack for ATT disclosure. Skewness attack, a distribution-based attack. t-closeness, so to fight distribution problems, to balance towards the overall population values. Also Group closeness has some issue and require action to improve and patch. There is no 100% safe solution of course. We should pick our protection tool based on legal requirements, task ahead, necessity and the tabs we are working with. Intersection attack protection. Is necessary to remove simplifying assumptions to deal with real world problem. Assumption on which k-anon, l-diversity etc etc where elaborate. Overdue of generalization and suppression is a possibility to control, to keep in check. Is still a demand to satisfy the one about cyber security and techniques. K-anon is a popular solution. Neighbourhood attack is also a thing. K-anon is possible to be used in also other scenarios, like social network etc etc. Incognito algs, uses generalization for ATT e suppression for tuples, k-ann but uses only a subsets to see if k-ann is respected as a necessary but not sufficient condition. Data mining, mining technique release are risky. Check results even if we use mining. Association rule is one way to check this. Association based on transaction. Putting together the various transaction, see how often x is bought together with y. (Elasticity). The same is applied to dataset about workers or whatever. Same hours of work, same sex, martial status etc etc etc. Support and confidence based and how much time two values for x attributes appears, frequencies basically. How much together with respect to the number of tuples (support). Confidence instead put together how much x compared together with y, over the number of tuples (confidence of saying how probable is that x –¿ is also y) the number of tuples that have only x (how is likely that x has also y values as other ATT), we should always check the complement case, that could actually violate the k-ann. Classification mining based on a decision tree. Exposure is a possibility even releasing one branch, through complement elements. K-ann in data mining. Anonymize and then mine, is the first approach. (protect micro and surely macro will be protected). Otherwise mine and then apply anonymize compatible techniques. Techniques to ke-ann compatible with location-based services, increasing the range such that more people are included basically, or just using ranges so is not possible no know where you are exactly, confuse people or laction choice. Precision is a cost if we are fogging geo-location instead of people, but if there are not enough people well, that's a problem. Syntactic, same quasi-id, make the remaining ATT not useful to recognize the single individuals. Semantic privacy def uses a mechanism to control it, like differential privacy. Presence/absence results with an analysis, if respected is semantic OK. Trade-off between protection and understanding, (perfect protection is still not reachable here). Genomic infos is a new field where privacy is surely necessary, many sensitive infos, unique perfect identifiers, and various infos about you, from the medical field and beyond. Be able to identify one specific person, risk for the relatives as well etc etc, a chain effect.

Data are invaluable, as we know. Deal with buzzword is terrible. Database, curator/Sanitizer, then released of course. Privacy, checking the results of the results, no possible exploitation are possible to identify individuals. The presence of individuals has to be presence, otherwise those data are useless. Preserve privacy, but keeping the data useful. Differential privacy, adding an individual should not change much the risk of that individual, limitation on what ca be inferred. Limiting it to what can be inferred anyway. Not single, but combination of all subject present. Risk increase based only on the analysis not on the single tuple the individual belong. Check if the difference is at most δ, we are basically bounding the risk, between output with and without a specific subject. Noise injection to perturbate data, same results overall, but difficult to inference single people. Approximation not exact results. Premium consequences of the results of the analysis. A simple choice problem, two scenarios, and see how the insurance will adapt, change the premium. (Game theory approach). Comparing the threshold and the impact of x over the results. Measuring the difference between the presence and the absence of individuals x. Too high impact, means risk release, we should not release the data. Small δ more privacy, less utility, the opposite if large, is always a trade off. $0 < \delta < 1$ Assuming always the worst case scenarios, safe in that, sea always (accounting principle here). Worst case assuming, different global sensitivity, using only the worst possible. Using Laplace distribution for sensitivity studying. 0-centred distribution. Laplace sharp peak on 0, alter less then other distribution like Gaussian. $\lambda = 1$ $\delta = 1$, high utility, low security. Softer peak, higher λ, 0 tendency of δ. Noise injection is based on sensitivity. Low sensitivity do not require much noise. Differential privacy composes will with itself. Differential Privacy, a semantic technique. The presence of an individual should no change drastically infos. ϵ differential privacy. Noise injection is useful to guarantee no high impact from the presence of a specific individual, of course ideally we don't want to perturb the population overall characteristic. To protect the subjects, as usual, worst case scenario is the one to deal with. Check for every single release of course. Cause is possible to use multiple release to check for a specific individual or disclosure specific infos. Laplace distribution to check for how much noise injection we are adding. Same results to the various release, to keep them equal. λ values, small λ high ϵ and vice versa. Outliers require much more noise, eliminate outliers is a info loss, so it's not something desirable. Closure under post-processing. Apply noise, then post-processing to make sense and guarantee reliability etc etc(like negative values with vars that should not have negative values). Differential privacy perform well with itself. Multiple dataset analysis with differential privacy, clear level of protection is computable. Sequential composition, m sequence computations over databe D with overlapping results. ϵ_m, is the sum of the various ϵ. Increasing ϵ decreases λ, lower protection. Parallel composition, disjoint based computations. Using various subsets, and it is a maximizing approach. 1 subject for 1 computation, no overlapping. Thus is easy to find the risk, just the worst of the them and they are all single case, no repetition. Parallel or sequential is an approach based on the attributes we are dealing with. Group privacy, protect groups instead of single individuals, like genes, if we have data from people belonging to the same family we have to protect all of them otherwise by disclosing one, is possible, potentially, to disclose all of them. Non interactive vs interactive model. If there is a final recipient that can ask continuous computation over the dataset, that's the interactivity. Single-shot use of the data request. Different λ, different injection, same request, different results, different people. Global and local differential privacy, global combined and inject noise, local collect data already injected. Each user runs a differential private basically, there is not much trust between user and data collector. Noise is not removable this way, but it does not sum up the way we expect, data are strongly disjointed. Same techniques basically. Differential privacy based on probability, coin tossing approach. Cause lies are a thing. Permanent randomized responds are store for ever, flipping the bloom filter, changing it. Rapport is instead an instantaneous randomized response. A significant difference can be realized, based on parameters. High difference low infos release high security, and vice versa, is the usual trade-off. Parameters make the security, not just the technique. Sensitivity of computations.

2.3 Authentication and Access Control

Prevention, Detection and Reaction. (BASICS). Prevention is the most relevant of course. Cloud overseer, that is not trustable. Confidentiality, integrity and availability. Cryptography using the encryption alg and encryption key. Asymmetric uses two key one per encrypting, one for decrypting. Symmetric uses just one key for both. Username and password is the easiest way for every point of view, but also the weakest among all of them (authentication). The user is the problem. Token are safer than password. Vulnerability of tokens, combined, steal token, the generating methods. Backup authentication factor for the biometrics base approach. Access Control case. Request to access resources. Accountability is also related to Access control. CIA, Confidentiality, Identification, Authentication. Access control DAC,MAC,RBAC,ABAC, Credentia-based access control. Access control is build on authentication system, thus is a requirement, a strong one indeed. Policy def, Model theory, Mechanism application. DAC is the basic one. Id is requested, explicit access rules. They are discretionary, as a policy, there are establish roles with powers and necessary action to enforce something, and is possible to pass those rights, delegate thus to build an hierearchy. Access Matrix Model is an example. Policy represented as a table, 2D table, so a Matrix. Is an abstract reps, in real system can be shaped in different form from table, but the logic is still there. SOA, triple is respected, S stands for set of subjects with privileges, O stands for object over which privileges are enforce. A stands for access matrix, rows for subjects, columns for object. $A = [s, o]$ report subject s over object o. Primary operation are of course a thing, like create new s or o, delete s or o, enter etc etc. Matrix is usually large and sparse. Waste of memory space, thus huge matrices are problematic. Alternatives approaches, authorization table, access control and capability lists. better storing is authorization table, just saving ONLY non-empty cells/tuple/table. ACLs store by columns, or store by row (Capability lists). It depends on what we want (the best solution). Centralized vs N-centralized. Centralized prefer ACLs, traditional solution, fewer bits. DAC weaknesses of course exist. DAC is vulnerable from trojan horses, who exploit privileges of calling subjects. Trojan horse performs unknown action to the caller(illegitimate action). MAC aim to avoid this, which is traditional. Mandatory access control, impose restrictions on infos flow, trusting subjects, but not programs. Decouple users from subjects. Mandatory policy is a multilevel security policy. Integrity classes, assign according to trustworthiness. Biba model for integrity, opposite of La Bella Padua. Biba has its own limitation. Too restrictive (flow restriction). Integrity, a complex concept, ensuring that only authorized people are modifying data, in a proper way. Both have to be take into account, proper and authorized. Role-based policies (RBAC). Role—¿sum of authorizations. Hierarchy are imposed with roles. Easy to manage, restriction, least privilege, separation of duty, etc etc. Limit abuses, and damages. SQL example.Administrative policies. Centralized and ownership based. Static approach specify separation of duty, dynamic change overtime, is more flexible.Conditional validity for authorization, or user groups. authorization for specific groups. Conditional however can be system, content or history based. Abstractions, based on hierarchical relationships. Logically some action imply other simpler actions require to perform the original action. Enforce NEG-authorization, by exclusion, non limiting exceptions. Permissions and denials.

2.4 Security

Protection of infrastructure, of communication, of devices, of data and against malware and attacks. The royal role of data in smart societies and choices.

2.5 Emerging scenarios

Focus on the privacy of stored data, no queries or single users. Issue and solution for stored data on cloud. Encryption is one of the way. Honest-but-curious (HbC). We don't trust the server to look at the content of our data. Fine-grained access to data. Onion layers of encryption (cool). Flattened indexed for queries, decreasing the exposure. Selective infos sharing, different approaches. So that it depends by the role, role-based functional data-access. Selective encryption is a way to enforce role-access. Is also a way yo enforce accountability when something goes wrong, since is possible too know who can access what. Authorization policy is a set of permissions, only read operation are possible (implicitly). Writeable only by data owner. Representable as a 0-1 Matrix (sparse), with users(rows), resources(columns). Authorization translated into an equivalent encryption policy. Different keys, key management is difficult, thus that is no good, thus role-based should be better. Key derivation method, alternative solution, from the knowledge of one key you can get the other. Even if is not reversible, if it is know, the system is breached. Key derivation hierarchy, is a graph. (token to get from a ket to another is a function after all). Token are given functionally, cause an access should rationally guarantee the access to another one, or the role itself should have this type of dynamics. On publish token are used are usable, not associated with time vars. Secure hash function (not invertible, token has to be this way). (Complicated, like resolving an encryption function). An encryption policy is set of token,as graph links, thus obtain functional access dynamics. Guarantee equivalence thus that access is possible if necessary and authorize. (From a sparse Matrix to a graph). Token management is problematic, when there are too many permission required. Trying to simplify, reducing number of keys, optimizing key derivation. Subets of of user that have access as ACL. Vertex/key. (forward feed graph). Network is the most valuable resources we have, we have computational power, we have limit given by the band width. Network as a bottle neck. Over-encryption, outsorcuing some of the burden of policy updates. Owner encryption and Provider encryption, two keys, box inside box. Outer box is provider, owner is inside box. Revoke permission, revoke using provider key. Grant requires both, your and then provider, so to grant an eventually more precise protection, since provider can block access to specific data. Extra steps, new vertex, that's Over encrypting after all. Now when granting instead, it's difference. Adding token basically and then Over encrypting if necessary to avoid undesired access possibility. Adding is always possible, subtracting is way more difficult.

2.6 Mix&Slice

take small size of data, encrypting and then uploading, such that users can not enter any more. Since you don't know if the cloud provider is actually doing it, when deleting, it's preferable to something like this by youself. Single data owner, one only user with write privileges, what about this is not the case. Multiple users owners and writeable privileges access have to be given. Subscription-based approach on encryption is a solution (cool, check it).

2.7 Fragmentation

Encryption its heavy by nature, making more expensive to query etc etc. While Fragmentation is just a simple way to solve the problem of sensitive-association, not single values singularly taken. Sensitive is having in the same file multiple value that combine create confidential problems, and sensitive-related issues(not taking into account publication problems). Different from sensitive association and sensitive attributes. Data fragmentation can also be used with encryption. When the two servers cannot communicate, confidentiality is given by splitting (frag) into two tables in two servers (two cloud providers). Accepting constraint, both single vars and combination, quasi-identifiers basically. Encrypting is also a solution, instead of dropping vars. (key identifiers are used to join, reconstructing association if authorized to do so). Keeping in the same fragment attributes "queried" together. (Identify the optimal decomposition). Coupling fragmentation is done at the beginning before migrating, while queries are done every time, continuously. Thus these two constraints are to be satisfy and optimize (more queries execution than frag). In house gives data about this, optimizing what is mostly done and required, tailor-made frag according to business requirements basically. Two non-communicating servers is an heroic assumptions, so it's not be used except really absolute certainty, even in the future. Association is limited, only 2 fragments basically, more than 3 constraints can bring us to a terzetto impossible. Confident of fragment non linkable to one-another. Multiple non-linkable fragments that contain all of what is necessary, no join is required, and all constraints are respected. Obey Maximal visibility, encrypt only sensitive values, and fragment such that there is no combination that violates sensitive access/release. Keep minimal the number of fragments. Singleton constraints. Maximum affinity is always a good principle to encourage and use.

In the cloud undoing is impossible. Optimization criteria are of course quite important, the existent of solution. Hybrid scenarios, trust own physical server vs not trusted cloud provider servers. Avoiding encryption by using not encrypting data stored only physical on a private server of the data owner. Complete, confidentiality and non-redundancy. Privacy breach problem when giving infos back to the cloud. Minimal fragmentation is to minimize the workload as much as possible.

3 combining Indexes, Selective Encryption and Fragmentation

Building blocks, Encryption and frag. What is sensitive is the combination of various information, so making difficult to reconstruct the complete set of sensitive infos is how to build security. Combining all this possible solution is surely better, safe prospective, to use them together (portfolio theory, diversification to minimize the risk), but also emergence of new difficulties caused by the combination of those solution. Access control and indexes improve efficiency. Vertical knowledge due to values appearing in the clear in one fragment and indexed in other fragments. Knowledge can also be horizontal. (This risk is matrix-related basically). Different indexes, different consequences when combining. Combining horizontal and vertical is always a step forward disclosure. Flattened has a positive impact in inference term, indeed strong. Many queries with same index values inference will be increasingly possible to acquire, is basically long-run reverse engineering. Access confidentiality and (single access aim to which data) Pattern confidentiality (more access aim to which data). Shuffle Index, a particular type fo tree, a B tree. It is usually use as a structure to store data. Confidentiality protection through encryption as always. Co searches is use to introduce confusion to achieve access o pattern confidentiality. Access confidentiality can be also be protected through pattern confidentiality.