Pseudonymization or Data anonymization – This is the question …

Wow... Happy reading! Well if that is the question and you have managed to get past the heading, I guess that you are also looking for an answer. Before I venture down that long-trodden path, let me just make sure that we are on the same page as far as meanings are concerned. Both definitions below have been taken from Wikipedia.

"***Data anonymization*** *has been defined as a "process by which personal data is irreversibly altered in such a way that a data subject can no longer be identified directly or indirectly, either by the data controller alone or in collaboration with any other party." Data anonymization enables the transfer of information across a boundary, such as between two departments within an agency or between two agencies, while reducing the risk of unintended disclosure, and in certain environments in a manner that enables evaluation and analytics post-anonymization.*

In the context of medical data, anonymized data refers to data from which the patient cannot be identified by the recipient of the information. The name, address, and full post code must be removed, together with any other information which, in conjunction with other data held by or disclosed to the recipient, could identify the patient.. De-anonymization is the reverse process in which anonymous data is cross-referenced with other data sources to re-identify the anonymous data source. Generalization and perturbation are the two popular anonymization approaches for relational data.. The process of obscuring data with the ability to re-identify it later is also called pseudonymization and is one way companies can store data in a way that is HIPAA compliant." (https://en.wikipedia.org/wiki/Data_anonymization)

"**Pseudonymization** is a data management and de-identification procedure by which personally identifiable information fields within a data record are replaced by one or more artificial identifiers, or pseudonyms. A single pseudonym for each replaced field or collection of replaced fields makes the data record less identifiable while remaining suitable for data analysis and data processing Pseudonymization can be one way to comply with the European Union's new General Data Protection Regulation demands for secure data storage of personal information. Pseudonymized data can be restored to its original state with the addition of information which then allows individuals to be re-identified, while anonymized data can never be restored to its original state." (https://en.wikipedia.org/wiki/Pseudonymization)

So; In short pseudonymization allows data owners to hide real data, while making it possible to restore (Or to put it another way, break) and see the original data value and Data Anonymization does the same while theoretically not allowing the data to be restored.

OK. I am pleased that we have got that sorted. But now what? Well … Both methodologies were seen as vital to keeping PII (Personal Identifiable Information), PHI (Protected Health Information) and other sensitive data secret, while still being able to share or work with the data. They also come with heavy costs, in the computation required to anonymize the data as well as the pre-work required to make sure that the correct fields have been protected. And then, what happens if one of the fields that you need to protect is a field that you are required to run analytics on? Now, you have an issue.

Let me give you an example of a field that maybe required for real analytics, but should also be anonymized for PII purposes;

Date of Birth: If you are doing analytics on a specific age group or if you are looking for an age demographic. Then this data (The real data) must be available to the analytics tools.

You may have guessed by now that I am not a great lover of either approach. One breaks analytics and the other is breakable! The real question here is not which approach to use, but what other solutions are available that will protect all my data. Not just enable compliancy for compliancy sake, but one which will make sure that the data of my customers, patients, staff and other users is stored securely, in a manner that renders it useless by attackers (internal or external), & means that it is unable to be reversed back to its originating format. While leaving it available for analytics and data testing and application use. This is a lofty set of goals and requirements, and one that is obtained with.

"***Homomorphic encryption*** *is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.*

Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted. In highly regulated industries, such as health care, homomorphic encryption can be used to enable new services by removing privacy barriers inhibiting data sharing. For example, predictive analytics in health care can be hard to apply due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished."
(https://en.wikipedia.org/wiki/Homomorphic_encryption)

With Real-Time Homomorphic Encryption, data can be secured in a way that can render it un-decryptable yet enables both searching and mathematical equations to take place. Below you will see a simple example.

Let us assume that the we have a database that has every field encrypted and all the original data replaced with truly morphed data, so that if you were to break in to the database all you would see, would be useless data that looked real! However; at the same time the data has been transformed into encrypted searchable and encrypted algebraic formats. Enabling a search like this one to take place:

Find me all the operations that took place on males between the ages of 21 and 33 within the zip code 33454

The SQL could in its very simplistic form look something like the following
SELECT Operation_Data FROM Operations_Table WHERE Gender_type = 'm' AND DOB BETWEEN '01/01/1990' AND '01/01/1992' AND ZIP ='33454'

With standard data anonymization or pseudonymization the above would not be possible, as the zip code would need to be anonymized and render it unavailable for analytics and date of birth should also be excluded. However; With Real-Time Homomorphic encryption you can perform this query, while maintaining the integrity of the data and NEVER allowing the actual data values to be seen, either accidently or maliciously.

Homomorphic Does Not Work? Does it ...

The differences between Standard Homomorphic encryption and Real-Time Homomorphic encryption is all around the time it takes to work on a query. It should be as fast as the underlying database.

I have a very simple rule when it comes to security, and that is,

"*If it takes longer with security than without. Find a different method*."

The reasons for this are incredibly basic. A user needs to do their job, they assume that security is in-built, however if they are stopped from, or delayed in carrying out a task then they will find a work around. In doing so they will invariable break all an organization's security. Of course, policies and procedures come in to play here. But if a user needs to do a task then they will do that task. Who among us have not used a cloud store to enable home use of a document that the Document Management System would not allow remote access to? Purely because we had an important presentation the next day. As it stands Standard Homomorphic encryption takes a long time to work on data when doing a Full-Text Search or a Mathematical sum, and so no. It does not work! Real-Time Homomorphic encryption on the other hand is as near as possible 'Real-Time'! This is due to the management of memory and CPU cores, enabling many processes to be carried out simultaneously without having a hit on the performance of the returned data. Or put another way:
SELECT SUM(Balance) FROM Account Where Transaction_Date BETWEEN '09/15/19' AND '10/15/19'
And should return a result as fast on encrypted data as it would normally do on an unencrypted data. Oh and of course the result should also be correct!

However; this also depends on the cipher keys and where they are stored. However; That is for another time. If you want to see Real-Time Homomorphic encryption in use and play with it, please logon to our Oracle JumpStart demo environment, where you can have a system up and running in under 5 minutes and start playing with a small dataset of around 2m records.