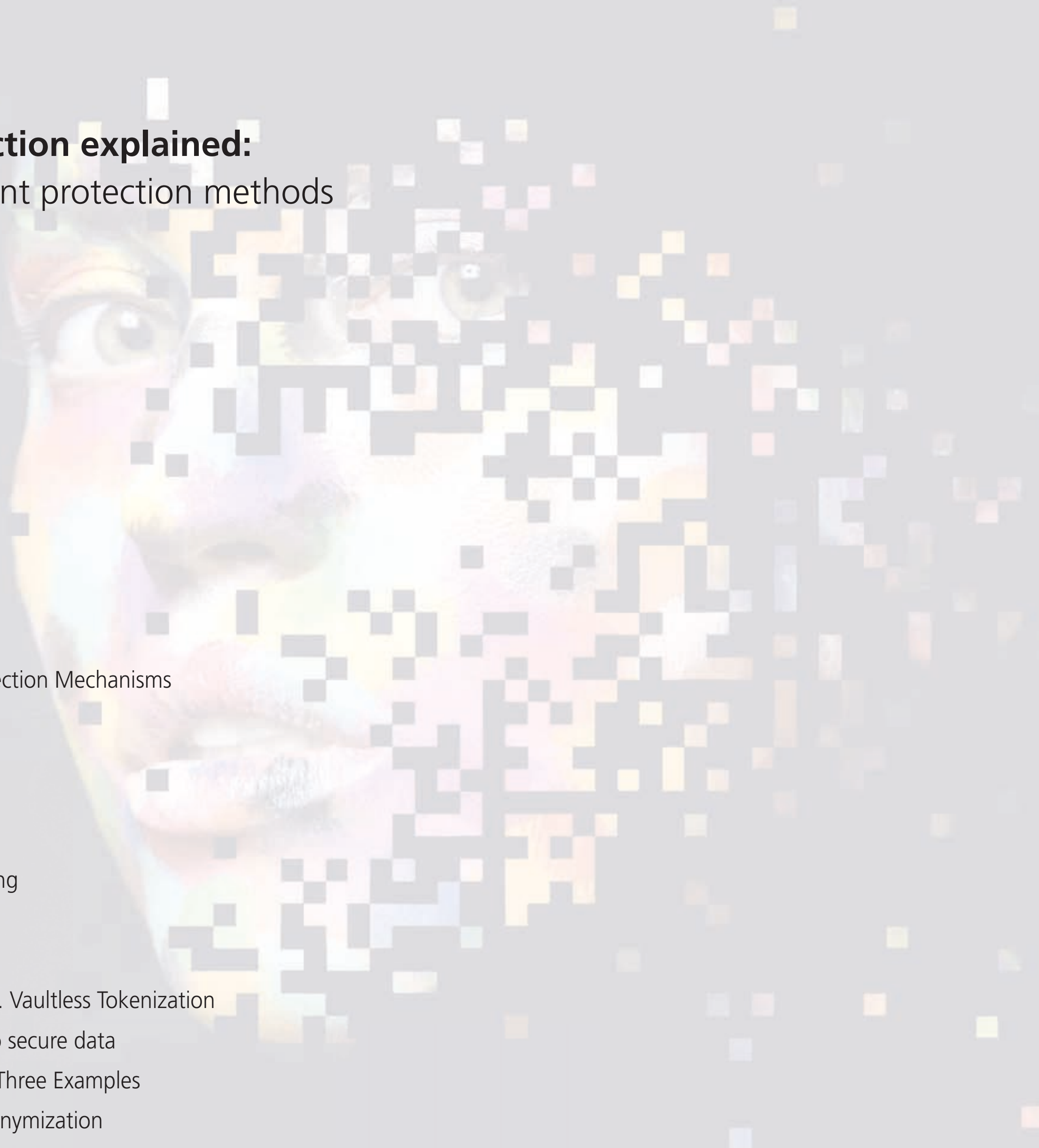
Data-centric protection explained:
Weighing the different protection methods

Data-centric protection explained:
Weighing the different protection methods



Companies are investing more and more into data protection. Compliance rules and regulations require organizations to develop sound security strategies in order to protect their valuable data assets and data privacy.

Reducing compliance burden is just one driver. Minimizing risk, or at least lowering the impact of breaches, is another one. It is a well-known fact that attackers are constantly seeking new ways to circumvent security to gain access to sensitive data. As many industry reports have shown, when it comes to a data breach, the costs can become astronomical due to the effects on stock price, customer retention, and brand reputation.

Another factor that motivates companies to protect sensitive data is monetizing regulated data to stay competitive and gain new business. Some organizations are only willing to work with companies who have the ability to share data that is already protected, rather than having to carry the responsibility of protecting the data themselves. Companies who can protect data in this aspect will be in a better position to retain existing and attract new customers and business partners.

Data-Centric Security

Protecting data is more important than ever before. Data is one of the most valuable assets for organizations as they become more and more data driven. Customers demand data privacy and partners expect that you conduct business with their data in a secure manner. Given the alarming frequency of data breaches reported worldwide, it is evident that classic perimeter defences and intrusion detection are becoming less and less effective – especially with the increasing complexity of infrastructure expedited by the complexity of cloud and hybrid infrastructure.

Data-centric security focuses on the protection of data itself and is based on two principles:

Protect data as early as possible in its life-cycle

De-protect data only when absolutely necessary

With the appropriate protection mechanisms, the security travels with the data – independent of applications, databases, and platforms while it’s at rest, in motion, or in use.
This allows organizations to take complete control of their sensitive data, lower compliance costs and significantly reduce the risk of data breaches.

How to differentiate?

Protection methods not only vary in how they change the data. There is far more that needs to be considered when choosing the right mechanism for a specific use case.

Protection methods differ in their properties: reversibility, secret symmetry, format configurability, determinism, and collision resistance. Most mechanisms can be implemented in various ways reaching from the management of secrets to the isolation model. These properties and implementation options are discussed in more detail in the following section.

Data Protection Technologies

Classic data protection technologies like encryption or hashing have a long history as a means of protecting sensitive data. Newer technologies, like tokenization, have only recently come into focus. Given the multitude of options, companies struggle to fully understand the differences, commonalities, and implications of encryption, tokenization, and other protection techniques and the broader system architecture and context they are used in.

This is especially the case as the terminology behind some protection methods is sometimes diluted by misunderstandings and the context it is used in. Often features or limitations of specific implementations are mistaken as an overall property of a protection method.
Marketing teams add another layer of obfuscation by developing messaging that positions a technology of choice in the desired way.

Basic Properties of Protection mechanisms
In general, every protection technique can be categorized based on the following five key properties:

1. Reversibility

The reversibility of a protection method determines whether or not a protected data element can be restored (reversible protection) or not (irreversible protection).
While reversible protection is a very common scheme, irreversible protection has its benefits. Using sophisticated, irreversible protection can make sure that a protected element cannot be unprotected, but still yields usable (meta-) information.

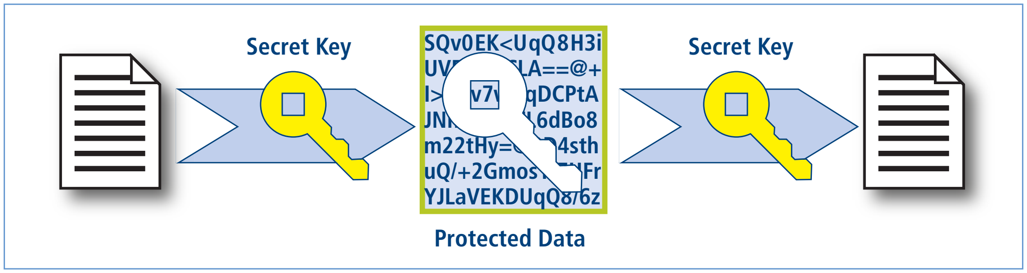
How to differentiate?

2. Secret Symmetry

Secret symmetry is a property of reversible protection schemes that determines whether the protection and the de-protection operation are using the same secret key. There are two secret symmetry schemes: symmetric protection and asymmetric protection.

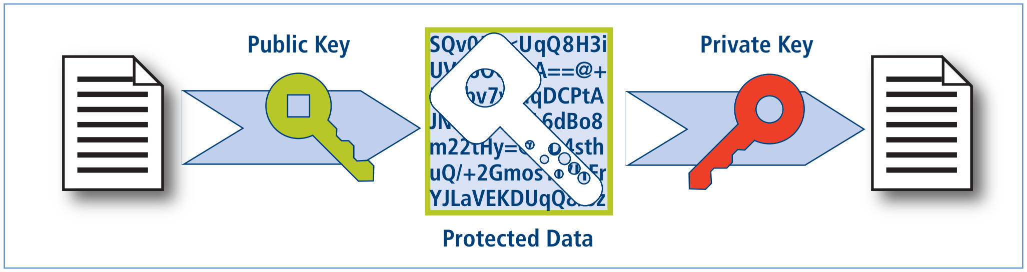
> **Symmetric Protection**

With symmetric protection, the same protection secret is used to both protect and deprotect data. Therefore all parties that have to either protect or unprotect data, must have access to the protection secret. As a result and by design, all involved parties with access to the protection secret can both protect and unprotect data.



> **Asymmetric Protection**

With asymmetric protection, there is an additional public component that is tied to the actual protection secret, which makes it possible to separate the ability to protect from the ability to deprotect. To protect data, only the public part tied to the protection secret has to be known. The deprotection of the data on the other hand requires access to the private protection secret. As a result, only those entities which need deprotection capabilities actually need access to the private protection secret. All other entities only need to know the public (i.e. non-secret) part.



How to differentiate?

3. Determinism

The determinism property of a protection method defines whether or not the protection and/or de-protection operation always results in the same output (given the same input and secret). Two-way deterministic protection techniques are those, which provide this property for both the protect and the deprotect operation.

Referential Integrity: A key resulting property of any deterministic protection technique, is referential integrity, which means that given the same protection secret, there is a one to one correlation between the protected and the unprotected element. Referential Integrity is a key prerequisite in order to correlate data sets using protected data elements.

4. Format Determination

The Format Determination of a protection method defines whether the output of a protection operation can be configured to the desired format of the protected data elements. A typical example would be to preserve the length and the alphabet used in the underlying sensitive data element.

5. Collision Resistance

Collision resistance defines how likely it is that two different inputs result in the same protected element and therefore collide. On the other hand, if using the same secret on exactly the same data element results in different protected elements over time, this is a sign that a specific protection method is not collision free/resistant. The same is true if the deprotection operation yields different results over time.

Protection Methods

This chapter takes a look at the most common protection methods, how they change data, their properties and their use cases.

Overview Properties

Protection Method	Encryption		Hashing		Data Masking	Tokenization		
Property	**Symmetric Encryption (AES, 3DES,...)**	**Asymmetric Encryption (RSA, ECC, ...)**	**Classic Hashing**	**Format Preserving Hashing**		**ODRA***	**Static Table based Tokenization**	**Encryption based Tokenization****
Reversible	**yes**	**yes**	**no**	**no**	**no**	**yes**	**yes**	**yes**
Symmetric	**yes**	**no**	**-**	**-**	**-**	typically **yes**	typically **yes**	typically **yes**
Format Deterministic	**no**	**no**	**no**	**yes**	**yes**	**yes**	**yes**	**yes**
Deterministic	**yes**	**yes**	**yes**	**yes**	**yes**	Requires additional external logic to ensure determinism	**yes** by design	**yes** by design
Collisions	**free**	**free**	**possible**	**possible**	**highly likely**	Requires additional external logic to prevent collisions	**free** by design	**free** by design

* *On Demand Random Assignment based Tokenization (ODRA)*
** *Also known as Format Preserving Encryption or FPE.*

Protection Methods

Overview Use Cases

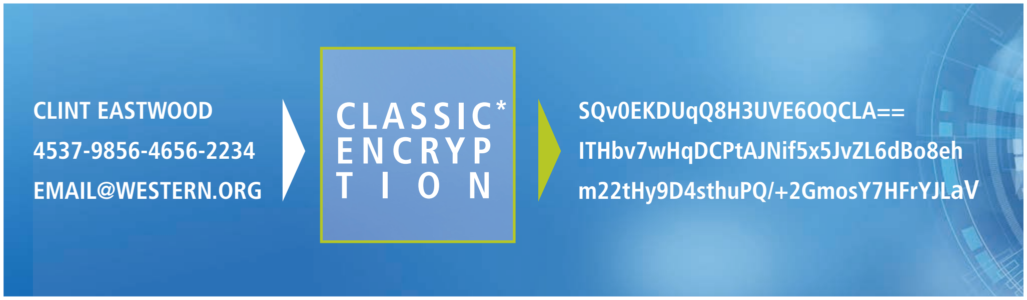
Protection Method	Encryption		Hashing		Data Masking	Tokenization		
Property	**Symmetric Encryption (AES, 3DES,...)**	**Asymmetric Encryption (RSA, ECC, ...)**	**Classic Hashing**	**Format Preserving Hashing**		**ODRA***	**Static Table based Tokenization**	**Encryption based Tokenization****
Typical Use Cases	Volume Level Encryption (VLE)	Data-in-Transit protection for complete streams	Protection and validation of passwords	Strong password protection for legacy applications with database field size restrictions	Protection of non-production data (test data, training)	High-value tokens (surrogates for payment transactions)	Data protection (PCI, PII, PHI, ...)	Data protection (PCI, PII, PHI, ...)
	Fullfile/device encryption for unstructured data	HTTPS Connections	Message Digest		Irreversible anonymiza-tion of PII	Blockchain-based tokens (eg. tokeniza-tion of assets)	Pseudonymi-zation for Data Analytics	Pseudonymi-zation for Data Analytics
	Bulk encryp-tion of large amounts of data	Digital Signatures			Limited visibility of data for employees or customers			

Protection Methods

Encryption

Classic encryption technologies have a long history as a means for protecting sensitive data. Encryption is a reversible protection mechanism that is deterministic and collision free.

When encrypting data, plaintext values are protected by using an encryption algorithm (cipher) generating a protected element (ciphertext). This ensures that only authorized parties are able to access the underlying data. Encryption can be implemented using either symmetric secrets (examples are AES or 3DES) or asymmetric secrets (RSA, EEC). With classic encryption, the format of the protected element (ciphertext) can't be configured. Furthermore it has a different length and typically contains values of a completely different alphabet.



**Algorithm: Des; Mode: CBC; Output encoded using Base64*

Changing the length and format of data elements has a huge impact when it comes to the implementation of data protection. Existing databases, message formats and applications are often programmed in a way that the maximum space each data element can take is predefined and limited to the data element's typical maximum length.

Typical use cases for encryption are data-in-transit protection for data streams, device encryption, encryption of unstructured data and binary large objects (BLOBs).

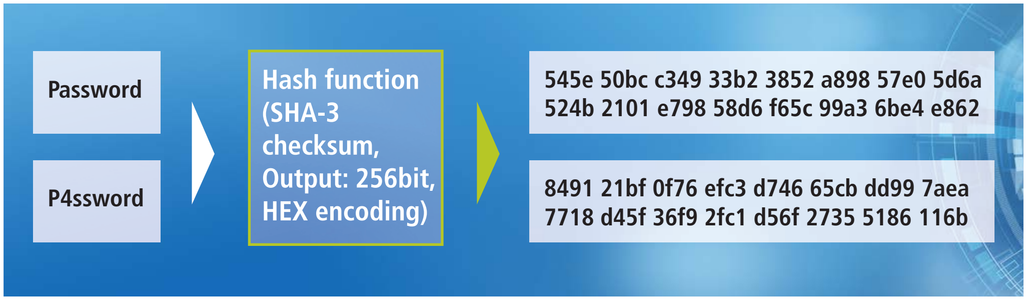
NOTE:
For more information about Format Preserving Encryption refer to section titled 'Encryption Based Tokenization'

Protection Methods

Hashing

A cryptographic hash function is an irreversible protection method. An algorithm creates a hash by mapping a data element to a bit string that is fixed in size. Small changes in the underlying data element change the hash drastically. Collisions are theoretically possible, especially when the output (lengths of the hash) is small. However, due to the fact that the number of possible hashes created by modern hash algorithms (e.g. SHA256) is huge, the probability of creating the same hash for two different data elements is infinitesimal.

While hashing is deterministic and therefore always results in the same hash for a given data element, classical hashes usually do not preserve the format, even for small domains. There are many use cases for hashing in information security, ranging from digital signatures, file-identification, identification of file corruption or message authentication to password validation.


Password
P4ssword
Hash function (SHA-3 checksum, Output: 256bit, HEX encoding)
545e 50bc c349 33b2 3852 a898 57e0 5d6a 524b 2101 e798 58d6 f65c 99a3 6be4 e862
8491 21bf 0f76 efc3 d746 65cb dd99 7aea 7718 d45f 36f9 2fc1 d56f 2735 5186 116b

Example: Hashing passwords

A common use case for hashing is the protection and validation of passwords. To validate a password, it is sufficient to know the hash of the password instead of knowing the clear text password. This is due to the deterministic property of cryptographic hashes, which by design have the same password resulting in the same hash. Thus, any given password can be validated by just hashing it and comparing it with the stored hash of the password. The storage itself also is secure as it is not feasible to calculate the original passwords back (assuming a cryptographically strong hash function was used).

Format Preserving Hashing

Classical hashes (e.g. SHA256) do not preserve the format of the protected value. Sometimes applications and databases require a specific format (e.g. length). To address this, Format Preserving Hashing algorithms can be used to provide irreversible protection with deterministic results.

Data Masking

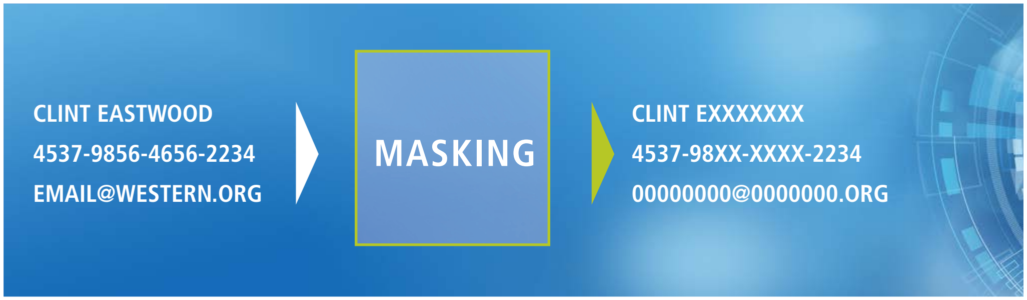
A typical example of irreversible protection is masking. Masking basically replaces a given number of characters of a sensitive value with a (set of) masking characters. Masking can be helpful to simply not show sensitive data at all or to provide sufficient amount of information to identify the data associated with the sensitive value (e.g. replace sensitive parts with X's).
It is possible to mask data in various different ways using techniques like partial x-ing or nulling out, substitution, shuffeling or number and date variance.

Due to the ireversible character of this protection method, masking usually does not involve any secrets. While the format of the masked value is highly configurable, there is a high risk of collision, as there is no one-to-one correlation between masked values and their plain text equivalents.

Typical use cases for masking are the irreversible protection of test data or data in non-production environments, the irreversible anonymization of PII or the limited visibility of data for employees or customers.

Example:
Customer support at a financial institution

If a customer wants to dispute a credit card transaction, the customer service agent needs to know which of the cards the transaction was made on. However, the customer service agent does not have to know the actual credit card number for that purpose. Instead, the last 4 digits would be sufficient to identify which card the transaction was made on.

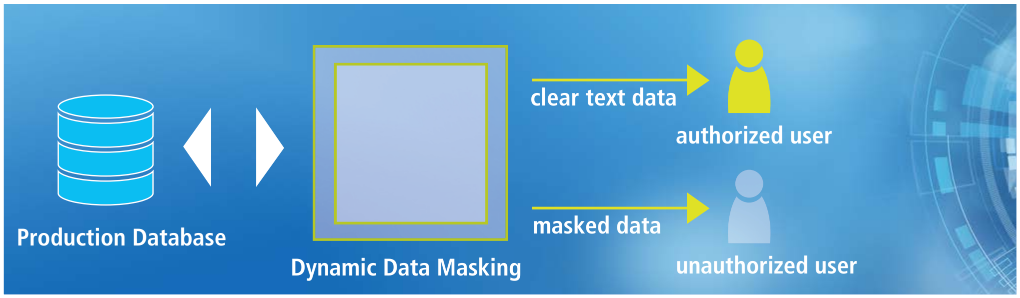


When implementing masking as a protection method, there is a difference between dynamic and static data masking.

Protection Methods

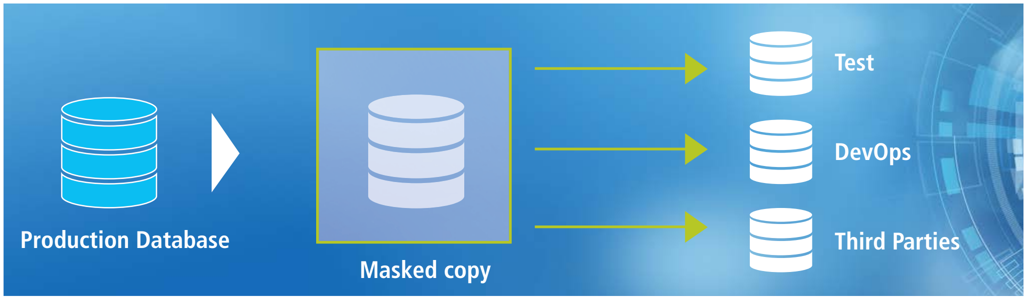
Dynamic Data Masking

Dynamic data masking protects data on its way through the application stack. In most implementations, an application intercepts the data stream and substitutes clear text values with masked values based on user rights and policies. If this capability is not provided at the application level, then this function is carried out by an agent, as is often the case in the context of PII.
While this solution provides strong benefits when it comes to executing data security policies, there is a high risk of misconfiguration and data exposure as the data still remains unprotected on core databases and on the move.



Static Data Masking

Static data masking changes the data permanently, usually on the database level. While static data masking limits the usability of a dataset permanently and might therefore not be used in production environments, it is more secure, as it reduces the risk of accidentally exposing the clear text values due to misconfiguration. This could especially be important when giving access to masked data to untrusted parties.



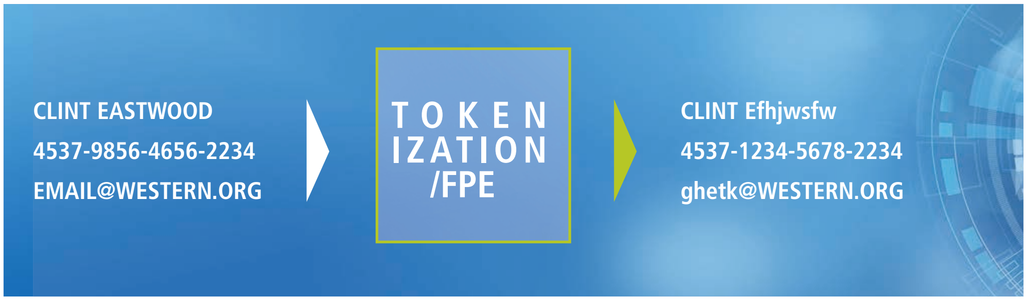
Protection Methods

Tokenization

Tokenization is a reversible protection mechanism. When applying tokenization, a sensitive data element is substituted by a so-called token. The token itself maps back to the original data element but doesn't expose any sensitive data.

The most common tokenization techniques are deterministic by design and operate collision free. In most cases, a tokenization system takes care of the tokenization and de-tokenization process and provides all necessary interfaces to allow other systems to perform protection and de-protection operations. Many tokenization systems make it possible to determine what the tokens should look like. This includes keeping the same format, staying within the same alphabet or transforming it to a different one. The main reason for the fast adoption of tokenization was the format preserving property which was a key differentiator and advantage over classic encryption. Furthermore, tokens require significantly less computational resources to process. Specific parts of data is kept partially visible for business functions such as processing and analytics while sensitive information is kept hidden. For example, when processing payments, in many cases only the first six digits of a Primary Account Number (PAN) are necessary. These digits alone cannot be used for fraudulent activities and leaving them visible saves a significant amount of computational resources. Tokenized data can therefore be processed much more efficiently, which reduces the strain on system resources. This is a key advantage in systems that rely on high performance.

Modern tokenization systems are used to protect all kinds of structured sensitive data such as payment information, healthcare records, personal identifiable information and other data elements. Due to the referential integrity between tokens and it is possible to operate on tokens instead of clear text data for many use cases.



Protection Methods

Tokenization Vault vs. Vaultless Tokenization

Tokenization can be implemented in various ways and with the help of different approaches. Some systems use a tokenization vault to map data elements to tokens. This technique, called **On Demand Random Assignment based Tokenization (ODRA**), was one of the initial techniques used to implement tokenization. For every new data element, a new mapping in a database (token vault) is created. When a token is to then be deprotected again, a lookup in the database is performed and, in case the respective mapping was found, the clear text element is returned to the requesting entity.

This results in a database that is constantly growing with every added value, making it stateful. These constant changes decrease performance and result in a database that becomes difficult to manage. For every protection operation, there is a need to look up if there is a mapping for the data element already. In addition, this lookup table has to be held and synchronized across all instances of the tokenization system.

Stateless schemes, like Static Table based Tokenization and Encryption based Tokenization (or Format Preserving Encryption), address the limitations and complexities of schemes using a token vault.

Static Table based Tokenization utilizes an algorithm that operates on a (set of) static, pre-generated tokenization table(s) to form a permutation of the original input set. These tables do not hold any token mappings, but are filled with random values. Those random values form the protection secret of the static table based tokenization scheme. The tables are static and as such no state has to be held, making Static Table based tokenization a stateless tokenization scheme. In addition, the tables are small enough (typically < 100 MB) to easily hold in memory and distribute to all instances. Static Table based Tokenization is (being a permutation) deterministic and collision-resistant by design.

Encryption based Tokenization (also known as Format Preserving Encryption or FPE) extends classic-encryption algorithms (typically AES) in a way that makes them provide the desired format determination, and essentially form a permutation of the original input set. The protection secret of Encryption based Tokenization is the key that is used in the encryption/decryption operations. Some FPE algorithms are declared to be not secure by cryptanalysis as it is theoretically possible to perform successful attacks.

NOTE:
Tokenization is a great method to implement data-centric security. It’s especially advantageous when implementing on existing environments with high throughput. In many cases, tokenization at least partially preserves the ability to process the data without deprotection due to its format preserving nature, which greatly reduces the effort required to implement it.

The actual flexibility of the general tokenization schemes listed above and covered in more detail below, vastly differ depending on the actual implementation, so it is important to understand what properties are prescribed by the general scheme vs. what properties are just a property or limitation of a specific implementation of a given tokenization technique.

Secret Isolation

Secret Isolation: the key to secure data

A very important property to take into account for overall security is the isolation of the protection system and its secrets. The level of isolation defines how hard it is to get unauthorized access.

The overall objective is to provide the highest levels of protection while still keeping the system usable. With secret isolation, there are two models:

1. the central access model
2. the shared access model

In a central access model, the protection system and the underlying protection secrets are only resident in a central system that nevertheless might consist of multiple distributed instances.
In contrast, a shared access model shares the protection method and protection secret among all entities that need access to the clear text data.

Sharing the Secret: Classic encryption is well known for sharing protection secrets and therefore complex key management. Tokenization usually implies a central access model with various layers of isolation to protect access to the system as well as the protection secret. However, it is important to note that all of the protection methods can be implemented in either way. Even in a central access system architecture, sharing a protection secret with non-isolated entities increases the risk of protection secrets being breached.

Rotation of the Protection Secret: One topic relevant to protection in general is the rotation of the protection secret. Regular protection secret rotation is intended to limit the impact of a potential breach by limiting the amount of data protected with the same protection secret.
This is most important when using a shared access model where unauthorized entities might get access to secrets. With classic-encryption, key rotation is a best security practice, and some algorithms like AES-GCM even require it because they lose their protection properties after a certain amount of data has been encrypted with them. In the case of tokenization, rotation of protection secrets doesn't have to be as stringent because tokenization typically implies an even greater degree of secret isolation. In addition the rotation of secrets often leads to missing referential integrity and implementation challenges as data sets are protected using different secrets over time.

NOTE:
The more isolated the system, the more secure. Of course, when taken to extremes, a 100% secure system is one that nobody ever can get access to. Such a system, while 100% secure, nevertheless defies its purpose as it is not usable.

Translated to real life: Imagine there is a need to secure the access to a building. In a central access model, basically one person, the guard, would hold the key (the protection secret) to the building, and would let authorized people in. The guard can also make sure they don't bring in unauthorized third parties. In a shared access model, however, all authorized people would get a key (the protection secret) and could get into the building on their own at any time.

Clearly with the central access model, be it in real life or in software, access control and auditing can be easily enforced. On the other hand, the shared access model is complex and tough to secure and difficult to audit.

However, a central access model comes with the risk that the enforced central access becomes a single point of failure. Imagine the guard is off sick. While the building is still secure, nobody can get in at all. That means the scheme is secure but not usable. To address this, a team of guards can be used, which still would be more secure than handing the keys (i.e. the protection secret) to the actual building users, as the team of guards can be especially trained ("hardened") for secure operation. The same is true with a software system, where a security hardened central access system deployed in a clustered fashion, yields the optimal combination of security, availability, scalability and reliability.

Regulations & Standards: Three Examples

Since 2013, approximately 10 billion data records have been lost or stolen, of which only about 4% were encrypted or tokenized, rendering them useless. The rest may very well be for sale on the dark web. To help companies deal with these breaches, numerous standards have evolved over the last few years which describe how data should be protected. Legislators and industry leaders are constantly updating their standards and regulations as new threats and new counter measures emerge.

Sometimes regulations require specific protection methods for sensitive data like primary account numbers (PANs), healthcare records, or other personal identifiable information (PII)*. Some consider data to be out of scope only when a specific configuration is applied. This could result in limitations when looking for the right protection mechanism.

** PII includes any data elements that can be traced to a specific person, including GPS data, genetic and Biometric data, browser cookies, mobile identification identifiers (UDID and IMEI), IP addresses, MAC addresses and application user IDs, among many others.*

Finding the right solution:
Implementing data-centric security helps organizations worldwide to achieve cross-regulatory compliance while reducing costs and effort when implementing new applications or changing architecture.

Liability and Obligations in Case of a Data Breach:

In the event of a breach, organizations will not necessarily be penalised, but they will have to demonstrate that their security apparatus was up to par and that they responded accordingly upon discovering the breach. For example, GDPR states that organizations are obligated to report any breaches of sensitive data to the appropriate parties in a timely manner. Failure to do so is what can result in considerable penalties. If the sensitive data involved in the breach was protected with the appropriate measures, such as tokenization or encryption, then it is not necessary to report it.

Regulations & Standards: Three Examples

The **Payment Card Industry Data Security Standard (PCI DSS)** is an information security standard for organizations that process, store, or transmit credit cards. For example, Requirement 3.4 stipulates that PANs must be unreadable anywhere they are stored. It specifies that data at rest can be protected with tokenization, truncation, one-way hashes of the entire PAN or encryption with proper key management. Requirement 4 calls for similar measures to protect data being transmitted over public networks.

These requirements are nearly identical to Article 32 of the EU's **General Data Protection Regulation (GDPR)**, which calls for "pseudonymisation and encryption of personal data… whether in storage, transmitted or otherwise processed". Given the definition of pseudonymisation as described in Article 4(5), personal data must be stored and processed in such a way that it cannot be traced back to a specific data subject without the use of tightly secured additional information.

The United States **Health Insurance Portability and Accountability Act of 1996 (HIPAA)** established standards to protect individuals' medical and personal health information. It applies to health plans, health care clearinghouses and health care providers that conduct transactions electronically. HIPPA requires companies that deal with personal health information to fully protect those records from unauthorized access while at rest and in motion.

PCi Data Security Standard V 3.2

Requirement 3.4

Render PAN data unreadable anywhere it is stored. Technology solutions may include strong oneway hash functions of the entire PAN, truncation, index tokens with securely stored pads, or strong cryptography.

General Data Protection Regulation

Article 32

Data Security measures should, at a minimum, allow: Pseudonymizing (tokenization) or encrypting personal data.

HIPAA COMPLIANCE

Public Law 104-191

An act to amend the Internal Revenue Code of 1986 to improve portability and continuity of health insurance coverage in the group and individual markets, to combat waste, fraud, and abuse ...

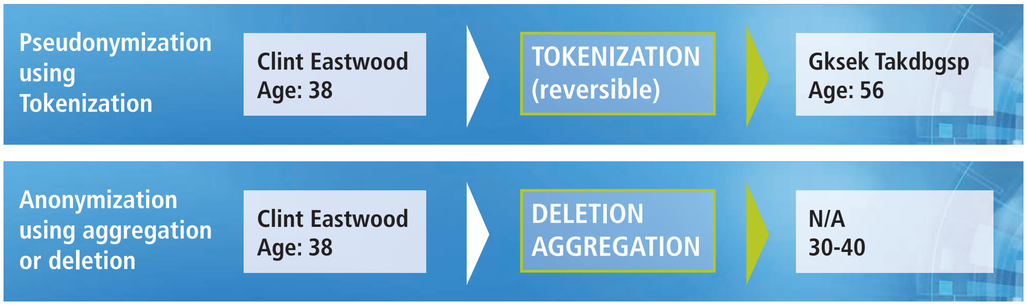
Other regulations like the California Consumer Privacy Act (CCPA), the Brazilian General Data Protection Law (LGPD), the India Personal Data Protection Bill, the New Zealand Privacy Bill, the Chile Privacy Bill Initiative, among many others, aim to regulate how organizations handle and protect sensitive data.

Anonymization vs. Pseudonymization

The two terms data anonymization and data pseudonymization have been broadly discussed since their appearance in GDPR and they can easily be confused when describing these very different processes.

The difference between pseudonymization and anonymization is all about the ability to de-identify personal data. When pseudonymized, data is processed in a way that it cannot be attributed to a specific data subject or an individual without the use of additional information. Data is only definitely pseudonymized, when the secret (additional information) is kept separately and is "subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person." (GDPR Article 4 Section 5 'pseudonymisation' https://www.privacy-regulation.eu/en/article-4-definitions-GDPR.htm).

When using anonymization, the data has to be changed in a way that a data subject can no longer be identified. Anonymization is irreversible.


Pseudonymization using Tokenization
Clint Eastwood Age: 38
TOKENIZATION (reversible)
Gksek Takdbgsp Age: 56
Anonymization using aggregation or deletion
Clint Eastwood Age: 38
DELETION AGGREGATION
N/A 30-40

While from a logical perspective this sounds easy, technically there are various ways to implement both techniques and, especially when talking about anonymization, implementation can be complicated as de-identification is not as easy to achieve as it seems.

The use of either pseudonymization or anonymization also has regulatory implications.
As pseudonymization is reversible, with GDPR for example, pseudonymized data is still considered personal data and therefore requires protection and consent for usage. However, if strong protection mechanisms are applied, it is not required to disclose a breach when only pseudonymized data has been stolen. Anonymized data is not considered personal information anymore and allows organizations to freely gain valuable insights from it.

5 Steps to implement Data-Centric Security

Step 1

Locate Sensitive Data
In order to effectively secure sensitive data, companies must identify all places where data is stored, processed or used. This is a necessary first step in complying with many regulations such as carrying out regular risk assessments, logging access and data disposal.

Step 2

Data Minimisation and Reduction of Scope
It is a common best practice to reduce the amount of data being processed. This has the advantage of minimising risk and reducing the time, effort and costs associated with securing data.

Step 3

Data Protection Risk and Impact Assessments
The threats to personal data and cardholder data are changing constantly. In order to keep up, organisations must conduct regular reviews to gauge how well data is protected. In addition, whenever an organisation undergoes major changes that might affect data security policy and processes, such as mergers and acquisitions, relocation or the adoption of new data processing systems, risk assessments must be carried out.

Step 4

Define Policies & Protection Methods
Security policies should define which data is going to be protected and which protection methods to use. Data classification tools can help to identify data elements and decide the right protection method. Limiting access to sensitive data and define access policies is another key component. Every account with access to sensitive data is a possible attack vector and therefore limiting access is analogous to limiting vulnerability. Implementing data-centric security using a central access model helps to execute access policies in real time while increasing visibility on data usage on user level. Based on business and regulatory needs, all protection methods have their strengths and weaknesses. In many cases, various protection techniques are combined together as building blocks within a broader system to use the respective right tool for the right purpose.
While reversibility is one of the main factors to consider, there might be other properties that are important when protecting data-at-rest, in motion or in use.

Step 5

Audit Trail
In addition to the accessibility limitations referenced above, logging access to sensitive data is another indispensable part of any data security strategy. Access logs are useful for proactively detecting potentially malicious activity and, if a breach does occur, they are essential to investigations to determine the source of the breach. Again, using a central access model enables organizations to log every single access to sensitive data and helps to create a detailed and meaningful access trail.

Conclusion

No matter how many cyber-attacks you manage to prevent, you can never assume you are stopping them all. Data security technology and regulations are constantly evolving to confront the dynamic threats to sensitive data. As with all forms of security, a multi-layered approach is the best method to prevent breaches. Data-centric security is a proven approach for protecting the center of a multi-layered approach: the data.

Protecting sensitive data at its earliest point of entry into your systems and reducing the need to expose the data allows your business to continue to grow while complying with regulations and reducing risks.

In summary, to evaluate the appropriate protection techniques to use, it is imperative to look beyond the high level buzzword messages and really understand the properties of the various methods and the isolation model. While this document can provide a good starting point for that, it is also recommended to discuss with experts in the field who have the experience to analyze your specific use cases.

comforte Data Protection Platform

comforte's data protection platform allows organizations to take complete control of their sensitive data. It has been built from the ground up to best address data security in a world that is driven by digital business innovations, empowered customers and continuous technology disruptions.

Based on your business and regulatory needs, it offers various protection methods including classic encryption, tokenization, format preserving encryption, format preserving hashing and masking. The basis of the platform is a flexible and sophisticated integration framework, which allows multiple layers of data protection for new and existing applications. In many cases data protection can be achieved without having to change the respective application.

Offering high performance, scalability and fault-tolerance, it enables protection of sensitive data with little to no impact on existing applications. It can be seamlessly integrated with other enterprise security solutions and provides a comprehensive and mature set of capabilities such as IT automation (designed for Infrastructure as Code (IaC)), access control and policy management, analysis and auditing of access, and data protection operations.

**comforte's data protection suite is in production at many leading organizations across multiple industry sectors, such as payments, telco, retail, healthcare and manufacturing.
It helps organizations achieve end-to-end data protection, lower compliance scope and costs, and significantly reduce the impact and liability of data breaches.**

To find out more about the core concepts and the architecture behind comforte data protection platform and why it is the best possible choice for your Enterprise-wide data-protection needs – read this solution brief: https://www.comforte.com/resources-detail/news/enterprise-tokenization-core-concepts-architecture/

comforte helps companies all over the world to protect hundreds of millions of payment transactions, healthcare records, insurance records, and more, reliably running in business-critical environments.

With more than 20 years of experience in data protection on truly mission-critical systems, comforte is the perfect partner for organizations who want to protect their most valuable asset: data.

We are here to enable your success by providing expertise, an innovative technology suite, and local support. To learn more, talk to your comforte representative today and visit www.comforte.com