developerWorks®

Developing GDPR compliant applications, Part 3: Minimizing application privacy risk

Dave Whitelegg

May 25, 2018

Part 3 of this series provides practical application development techniques that can alleviate an application's privacy risk. These solutions include utilizing database encryption, data pseudonymization, and assuring a robust level of application security.

Part 3 of this series provides practical application development techniques that can alleviate an application's privacy risk. Part 1 summarizes the GDPR and Part 2 provides developer guidance for integrating privacy risk evaluation and mitigation within the software development lifecycle.

Introduction

There are a number of technical solutions that can be utilized by applications to significantly enhance the protection of personal data. While the GDPR does not require the usage of personal data anonymization, pseudonymization or encryption, such technical solutions are referred to within the GDPR as methods to mitigate a data subject's privacy risk. These techniques can also significantly reduce the GDPR compliance scope and risk to application provisioning organizations, where they are in a responsibility role of a controller or a processor (hosting). Therefore application developers should understand and consider applying each of the personal data protection techniques referred to by the GDPR.

The GDPR does not provide a descriptive set of technical security requirements, but best practice application security is regarded as a data privacy fundamental obligation. Weak application security can result in a personal data breach which could cause distress and harm to high numbers of individuals. Under GDPR Article (5) f, an application can be considered as unlawful should personal data be either maliciously or accidentally compromised, due to the application’s security not being at an appropriate standard.

Article (5) *Personal data should be (f) processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures ('integrity and confidentiality').*

Mass personal data breaches caused by weak application security, presents a significant risk for the organization developing and providing the application (controller). As such incidents are likely to invoke substantial financial sanctions by data privacy supervisory authorities, and cause significant reputational damages due to the GDPR’s data breach disclosure obligation.

This guidance provides developers with an explanation of the application data privacy mitigation techniques referred to within the GDPR. The guidance also provides an overview of the appropriate application development security practices, as expected by supervisory authorities overseeing GDPR compliance.

Anonymization

Anonymization is a personal data sanitation process that changes personal data so it is no longer possible to identify any individuals (data subjects). Data anonymization is a one-way process. When a dataset has been anonymized, it should be impossible to reverse the process and transform the dataset back into personal data.

Personal data that has been adequately anonymized is no longer classified as personal data, which not only safeguards application user's privacy, but takes the data out of scope of all the GDPR legal privacy obligations. This is confirmed within Recital 26, which states the principles of data protection do not apply if personal data is rendered anonymous in such a manner that the data subject is no longer identifiable.

Recital 26 *The principles of data protection should therefore not apply to anonymous information, namely information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable.*

Application processes, third party tools, and direct SQL database commands can provide effective personal data anonymization. For personal data anonymization to be considered successful, the anonymization process must null or remove all direct and indirect personal data identifiers, which include:

- The data subject's name
- Application account and user IDs
- Email addresses, including those without data subject names
- Nicknames, including forum and online handles
- Identifiers that can be looked up to identify an individual elsewhere, such as account numbers and reference numbers; biometric data, including genetic data
- Location and tracking data, including street address, geotag ordinates, IP addresses and application cookies
- Text fields that potentially could hold personal data (for example, user comments)

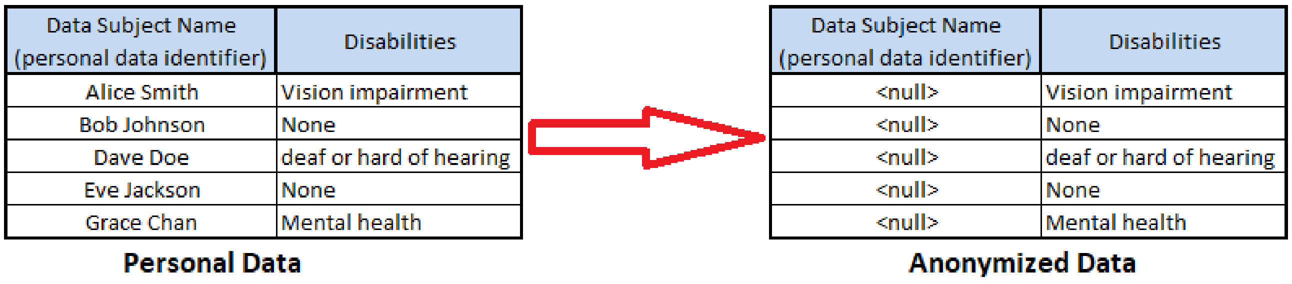
Data anonymization methods

There are different advantages and privacy risks with the various types of data anonymization methods. The most common personal data anonymization methods are described below.

Nulling, deletion, and redaction

Anonymization is achieved by deleting or blanking (nulling) all direct and indirect personal identifier fields.

Figure 1. Example of nulling anonymization



Data Subject Name (personal data identifier)	Disabilities
Alice Smith	Vision impairment
Bob Johnson	None
Dave Doe	deaf or hard of hearing
Eve Jackson	None
Grace Chan	Mental health

Personal Data

Data Subject Name (personal data identifier)	Disabilities
<null>	Vision impairment
<null>	None
<null>	deaf or hard of hearing
<null>	None
<null>	Mental health

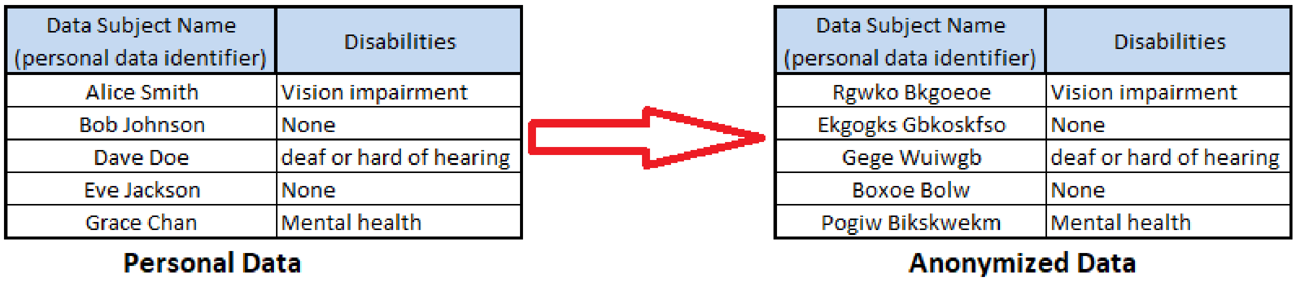
Anonymized Data

- Pros: There is little risk of de-anonymization (the process being reversed engineered back into personal data.
- Cons: The loss of the dataset structure can be issued for developers seeking to create test datasets from production data, as null or blanked data fields significantly hinders application testing capabilities, and can cause application errors.

Substitution

Anonymization is achieved by overwriting personal data identifier fields with fake personal data. Substitution anonymization involves either swapping in customised made-up data from a table, or by using an algorithm to generate random data which matches the expected value for the field type.

Figure 2. Example of substitution anonymization algorithm



Data Subject Name (personal data identifier)	Disabilities
Alice Smith	Vision impairment
Bob Johnson	None
Dave Doe	deaf or hard of hearing
Eve Jackson	None
Grace Chan	Mental health

Personal Data

Data Subject Name (personal data identifier)	Disabilities
Rgwko Bkgoeoe	Vision impairment
Ekgogks Gbkoskfso	None
Gege Wuiwgb	deaf or hard of hearing
Boxoe Bolw	None
Pogiw Bikskwekm	Mental health

Anonymized Data

- Pros: The anonymized dataset maintains its structure, therefore is a suitable anonymization technique for creating production quality test data for use within the testing phase of the Software Development Lifecycle (SDLC) and for replicating application issues as part of support.
- Cons: There is potential for direct or indirect personal data identifier fields to be missed and to remain undiscovered within the dataset. As actual personal data within the anonymized dataset can be difficult to observe and verify.

Data masking

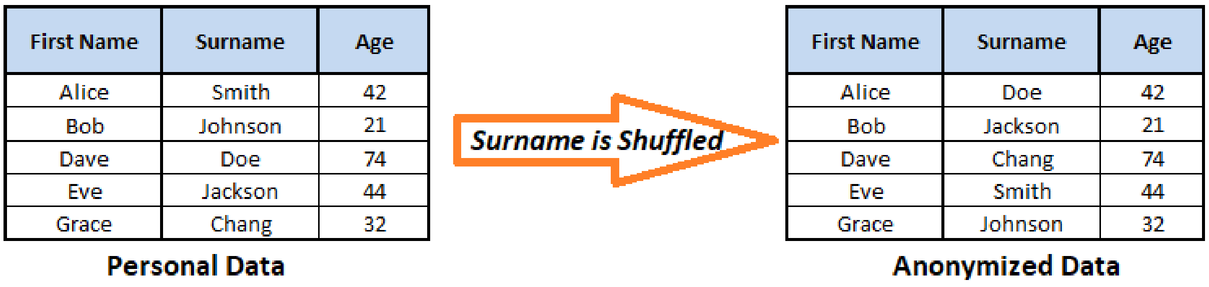
Data masking is a hybrid of the substitution and nulling anonymization techniques, and anonymizes personal data by substituting field characters with a ‘mask’ character. Data masking is widely used within the payment card industry, to allow payment cards to be identified without any risk of fraud, by using masking to partially reveal a payment card's primary account number e.g. 1234 56 XX XXXX 1234. You may be familiar with this particular use of masking on payment receipts, statements, and within online banking applications.

- Pros: Masking can reduce the privacy risk where personal data is required to be shared, displayed or printed.
- Cons: While personal data masking is suitable for account and reference number identifier fields, it can be a weak form of data anonymization with other types of fields, as masked personal data may still be either identifiable or reversible e.g. John Smith masked to J*** Sm**** could still be enough identify a data subject.

Scrambling \ shuffling

Data scrambling or shuffling is a substitution method where the personal dataset is solely used to swap 'columns' of data. In the simple example shown in Figure 3, the surnames have been shuffled to create an anonymized dataset.

Figure 3. Data shuffling example



First Name	Surname	Age
Alice	Smith	42
Bob	Johnson	21
Dave	Doe	74
Eve	Jackson	44
Grace	Chang	32

Personal Data

First Name	Surname	Age
Alice	Doe	42
Bob	Jackson	21
Dave	Chang	74
Eve	Smith	44
Grace	Johnson	32

Anonymized Data

Surname is Shuffled

- Pros: The anonymized dataset maintains its structure therefore remains suitable for use as test data.
- Cons: Data shuffling has a considerable de-anonymization risk, given the anonymized dataset could be pieced back together to identify individuals. The shuffling algorithm is also at risk of being deciphered. In the example shown in Figure 3, the anonymized data could be reversed engineered by cross referencing a record's first name and age fields with a third party dataset, or by deciphering the shuffle algorithm, which is substituting in the surname field which held two records ahead each time.

Aggregation \ generalization

Data aggregation or generalization is a data anonymization technique where personal data is analyzed and then rendered into a statistical or summary form, removing personal data identifiers as part of the process.

- Pros: Aggregation can reduce privacy risk for certain types of datasets, such as research data and outputs of big data analytics. For example, if a web application has a requirement to track the general location of its users, an application process could generalize each user's

recorded IP address to a country of origin. This process anonymizes the personal data identifier, namely the user’s IP address, while still achieving the application’s requirement of summarizing the general location of users.
- Cons: The aggregation and generalization of personal data, is only of value where there is a requirement to gather statistical or summary information about groups of individuals.

Reidentification

For personal data to be considered anonymized, there must be no possibility of reidentification, whereby individuals can be still be identified by matching the anonymized data against other external data, such as publicly available information on social media and websites, or from other data sets.

Advances in data mining techniques and data analytics means there is a greater likelihood of the re-identification of anonymized personal data. For example, as part of a data mining competition in 2006, Netflix publicly released 100 million records of movie ratings by 500,000 of its customers. The dataset was anonymized by substituting Netflix usernames with random numbers. Sixteen days after the release, two researchers at the University of Texas reported they were able to identify data subjects in the dataset, not by reversing the anonymization process, but by cross referencing movie rankings and timestamps with public held information on the Internet Movie Database website, IMDb.com. Using this reidentification technique, the researchers were able to identify significant numbers of Netflix users. The researchers also found it was possible to reveal individual Netflix users viewing history, which is regarded as a serious privacy intrusion, as an individual's movie viewing habits could reveal their sexual orientation.

Testing reidentification

Developers should seek to verify all personal data anonymization processes are sufficient by arranging for anonymized datasets to be tested by a suitably qualified penetration tester. It is a security best practice for applications to be penetration tested within the testing phase of the SDLC. Developers should ensure reidentification testing of all anonymized datasets is included within the penetration test scope. Including both application process and developer created anonymized datasets; that is, the creation of test data.

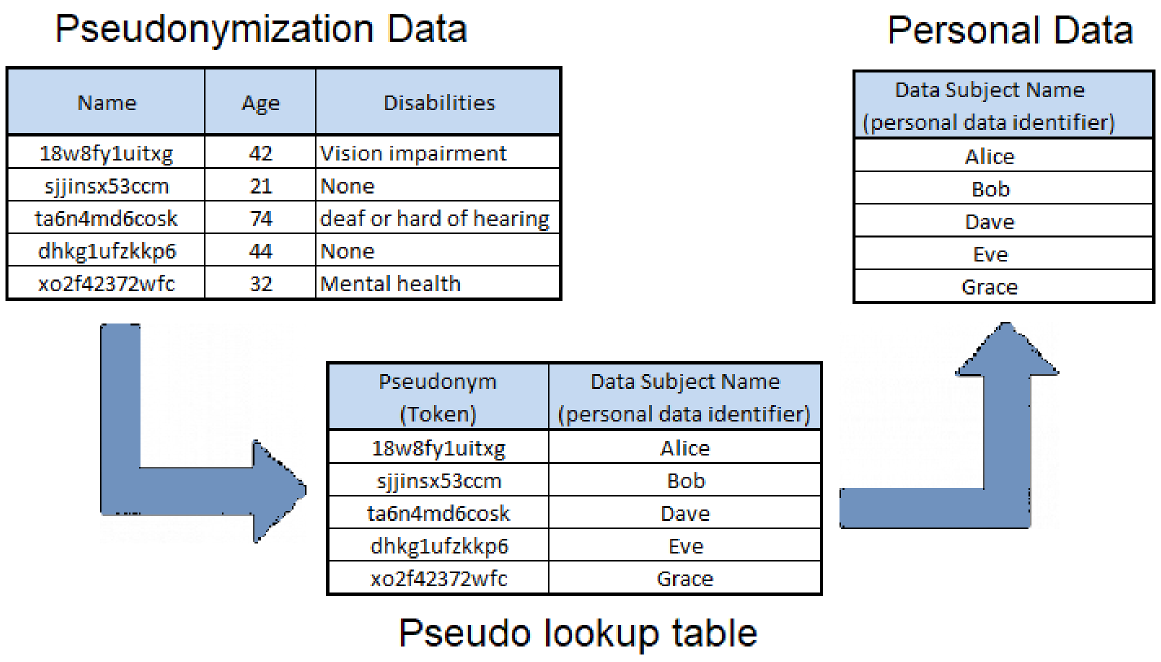
Reidentification testing of anonymized data should include:

- Attempting to identify individuals with one or more private facts
- Attempting to reverse engineer the anonymization process
- Using publicly available and lawfully obtained data to identify individuals within the anonymized data set

Pseudonymization

Pseudonymization is referred to in GDPR Article 4, and is a process that replaces personal data identifiers with realistic fictional data, known as a pseudonym or a token. Unlike anonymization, the pseudonymization of personal data is intended to allow the reidentification of personal data when required. A pseudonym value replaces personal data fields in the application dataset. This data substitution is tracked within a separate lookup table, which then can be used reidentify personal data.

Figure 4. Pseudonymization



Pseudonymization Data

Name	Age	Disabilities
18w8fy1uitxg	42	Vision impairment
sjjinsx53ccm	21	None
ta6n4md6cosk	74	deaf or hard of hearing
dhkg1ufzkkp6	44	None
xo2f42372wfc	32	Mental health

Personal Data

Data Subject Name (personal data identifier)
Alice
Bob
Dave
Eve
Grace

Pseudo lookup table

Pseudonym (Token)	Data Subject Name (personal data identifier)
18w8fy1uitxg	Alice
sjjinsx53ccm	Bob
ta6n4md6cosk	Dave
dhkg1ufzkkp6	Eve
xo2f42372wfc	Grace

Pseudonymization reduces application privacy risk by moving the personal data identifiers to a separate database server, which is typically isolated from the application's network environment. If the application dataset holding pseudonymization data is compromised, an attacker or third party does not have the ability to lookup the pseudo value and identify personal data subjects.

Article 4 (5)*'pseudonymisation' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such 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.*

Pseudonymization is also known as tokenization, and can be configured to provide token data that matches the field type and expected data value, allowing even legacy databases to maintain structure. Tokenization solutions are used by Apple Pay, Samsung Pay and Android Pay to prevent payment card fraud, by storing a token value on each smart phone instead of a payment card number. When a smart phone card payment is requested, the token held on the smart phone is sent and matched to the stored payment card details held within a secured data centre, where the payment is processed. Tokenization solutions have proven popular within the Payment Card Industry (PCI), as they reduce the risk of payment card fraud and data breaches, and substantially lower organizational costs by significantly reducing the number of PCI Data Security Standard requirements an organization has to comply with.

In conclusion, the use of tokenization by applications should be considered as a proven technology to reduce GDPR compliance overheads and cost while mitigating data subject privacy risk, with pseudonymization specifically cited as a risk mitigation option within Article 32 (1).

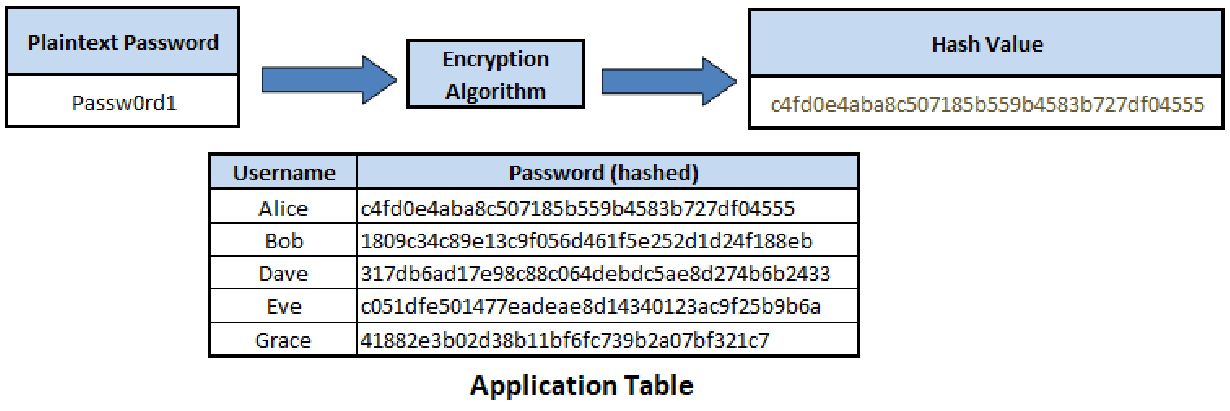
Article 32 (1)*...the controller and the processor shall implement appropriate technical and organisational measures to ensure a level of security appropriate to the risk, including inter alia as appropriate (a) the pseudonymisation and encryption of personal data;*

Hashing

Hashing is a form of pseudonymization that uses a mathematical algorithm to transform personal data fields into fixed length obscure alphanumeric strings, known as hash values or message digests. The hashing of plaintext fields into hash values is intended to be a one way data transformation process, as a hashing algorithm is not reversible; that is, able to transform a hash value back into plaintext. Hashing is typically used by applications to verify plaintext inputs by hashing a plaintext input value using a defined hashing algorithm. the resulting hash value can be checked against a previously calculated and stored hash value. If the hash values are the same, it is extremely likely the plaintext input and previously hashed plaintext are the same. The privacy risk mitigation is that the plaintext values are not held within the dataset.

Applications should use hashing to protect passwords held in databases, storing password hash values instead of plaintext passwords. When a user inputs a plaintext password during the authentication process, the application converts the plaintext password using the defined hashing algorithm to a hash value, and then checks if it matches the password hash held in the database. Should the application database be compromised, although the password hash values are visible and accessible, they cannot be used for account authentication, and it is near impossible to reverse the hash values back into plaintext passwords.

Figure 5. Password hashing

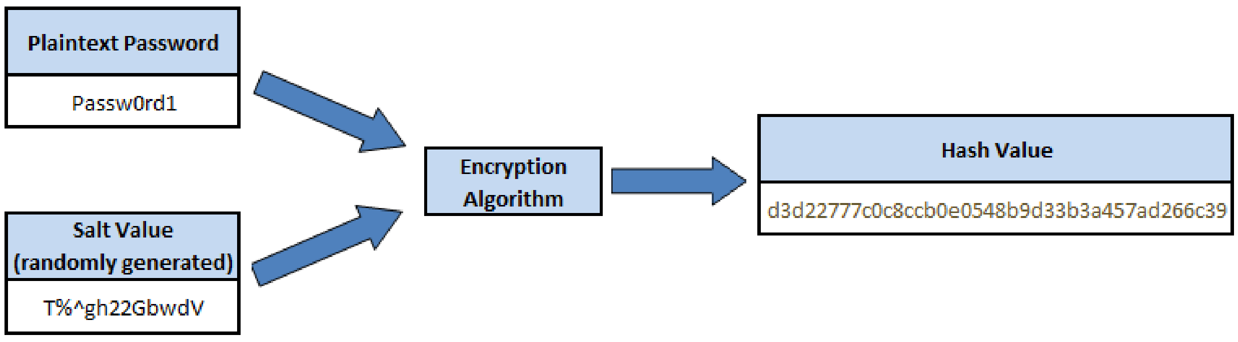


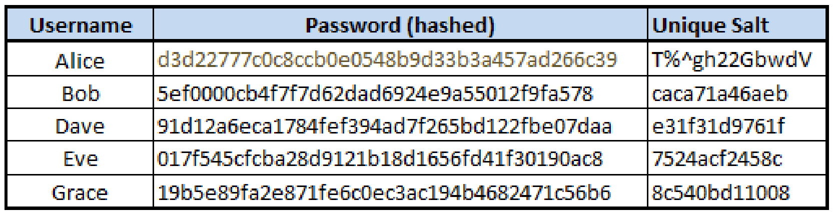
Username	Password (hashed)
Alice	c4fd0e4aba8c507185b559b4583b727df04555
Bob	1809c34c89e13c9f056d461f5e252d1d24f188eb
Dave	317db6ad17e98c88c064debdc5ae8d274b6b2433
Eve	c051dfe501477eadeae8d14340123ac9f25b9b6a
Grace	41882e3b02d38b11bf6fc739b2a07bf321c7

Application Table

While hashing is intended to be irreversible, it is possible to attempt to crack hash values by using a rainbow table, which is a large pre-computed list of all the possible hash values to plaintext values. However, by 'salting' the hashing formula by adding an unique record parameter, and using strong hashing algorithms such as SHA-512, it greatly decreases the likelihood of reverse engineering hash values back into their plaintext value. As given the use of a salt, an attacker would have to pre-compute a unique rainbow table, while the use of a strong hashing algorithm like SHA-512, which uses 512 bits, means it would take an immense amount of computer power and time to create the rainbow table, making the attack method not viable.

Figure 6. Salted password hashing


Plaintext Password
Passw0rd1
Salt Value (randomly generated)
T%^gh22GbwdV
Encryption Algorithm
Hash Value
d3d22777c0c8ccb0e0548b9d33b3a457ad266c39



Username	Password (hashed)	Unique Salt
Alice	d3d22777c0c8ccb0e0548b9d33b3a457ad266c39	T%^gh22GbwdV
Bob	5ef0000cb4f7f7d62dad6924e9a55012f9fa578	caca71a46aeb
Dave	91d12a6eca1784fef394ad7f265bd122fbe07daa	e31f31d9761f
Eve	017f545cfcba28d9121b18d1656fd41f30190ac8	7524acf2458c
Grace	19b5e89fa2e871fe6c0ec3ac194b4682471c56b6	8c540bd11008

Application Table

The application's enforced user password strength is another key element in protecting hashed passwords within compromised datasets. The risk posed by dictionary and brute force attacks, which can use compromised hash values to verify success, can be significantly reduced by enforcing complex user passwords of 8 characters or more.

A complex password strength is considered to include at least one of the following:

- Uppercase letter
- Lowercase letter
- Numerical digit
- Special character (#, $, *, %)

While hashing avoids the management overhead of encryption and can play a role in mitigating privacy risk in protecting passwords, most personal data fields required by applications do not typically suit the hashing pseudonymization approach.

Personal data encryption

Data encryption is the method that can greatly reduce privacy risk, and should be considered where anonymization and pseudonymization are not viable options in protecting personal data processed by an application. However, encrypting personal data has a key management overhead and potential security weaknesses that developers need to appreciate before designing and implementing personal data encryption.

The GDPR does not explicitly require personal data to be encrypted, but refers to using encryption in Article 6, 32 and 34, and in Recital 83. According to Article 34, if the application data is encrypted, and is then lost or stolen without the means (the private key) to decrypt it, the data loss would not constitute high risk to users (data subjects), and as such does not need to be reported. This means encrypted personal data without the means to decrypt it, can be no longer considered as personal data.

Article 6 (4)*Where the processing for a purpose other than that for which the personal data have been collected is not based on the data subject's consent or on a Union or Member State law which constitutes a necessary and proportionate measure in a democratic society to safeguard the objectives referred to in Article 23(1), the controller shall, in order to ascertain whether processing for another purpose is compatible with the purpose for which the personal data are initially collected, take into account, inter alia: (e) the existence of appropriate safeguards, which* ***may include encryption*** *or pseudonymisation.*

Article 32 (1)*... the controller and the processor shall implement appropriate technical and organisational measures to ensure a level of security appropriate to the risk, including inter* ***alia as appropriate*** *(a) the pseudonymisation and* ***encryption of personal data****;*

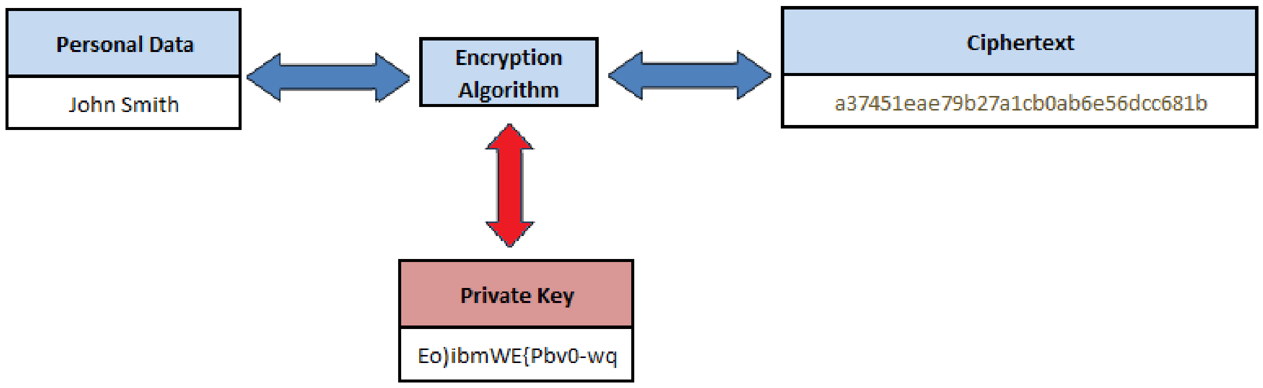
Recital (83)*(1) When the personal data breach is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall communicate the personal data breach to the data subject without undue delay. (3) ...the communication to the data subject referred to in paragraph 1 shall not be required if any of the following conditions are met: (a) the controller has implemented appropriate technical and organisational protection measures, and those measures were applied to the personal data affected by the personal data breach, in particular those that render the personal data unintelligible to any person who is not authorised to access it,* ***such as encryption****;*

Article 34*In order to maintain security and to prevent processing in infringement of this Regulation, the controller or processor should evaluate the risks inherent in the processing and implement measures to mitigate those risks,* ***such as encryption****.*

Encryption overview

Data encryption uses a mathematical cryptographic algorithm with a secret key code to convert personal data from 'plaintext' into 'ciphertext' which is unreadable without the knowledge of the secret key to reverse the encryption process, as depicted in Figure 7. Modern cryptographic algorithms are considered to be secure against attack, but the secret or private key is the Achilles Heel with encryption, in that if the key is compromised by an attacker, they would have the ability to decrypt the data.

Figure 7. Symmetric encryption



There are various software and hardware encryption solutions that can be utilized by applications to encrypt personal data. Encryption can be fully managed by the application, performed directly at the database server, or utilize dedicated encryption management devices known as Hardware Security Modules (HSM). In addition to database encryption, there is filesystem encryption, for protecting personal data within application server temporary files, within audit logs, and within data backups.

Encryption methods

There are two main types of encryption methods: symmetric key encryption and asymmetric key encryption. Symmetric key encryption uses a single private key, for example, AES-256. Asymmetric key encryption utilizes a combination of publicly shared and private keys to allow two parties to encrypt data over a non-trusted connection, for example, using digital certificates, the RSA algorithm. Application personal data encryption commonly uses symmetric key encryption, which requires a single private key to be safeguarded and securely managed.

Private key management

There are a number of processes in managing private encryption keys that need to be accounted for within the design of an application intending to utilize data encryption, including where the application will be reliant on a third party to provide or manage the encryption.

- Key generation: Newly created private keys should be randomly generated. Not to be confused with creation of a password or passphrase, an encryption key lengths are commonly 128 bits or greater, and should be generated using a random number generator or pseudorandom number generator. This ensures the private keys are completely randomly formed, reducing the risk of keys being guessed or reversed engineered.
- Key storage: Private keys must always be stored and held securely, including when in use by the application to encrypt and decrypt personal data.
    - A private key must never be application 'hardcoded', namely written within the application’s code, as this greatly increases the possibility of the encryption being compromise. This includes hashing or encoding the encryption key to disguise the private key value within the code.
    - Given the private key needs to be accessible by the application to perform encryption and decryption, and storing this key, even temporarily, presents a significant risk in

subverting the encryption, the solution is to encrypt the private key as well. This can be achieved by using Key Encryption Key (KEK), a process that is typically supported by a component outside of the application, such as by a HSM. It is important that the KEK strength is at least as strong as the personal data encryption strength, otherwise personal data encryption strength is considered to be weakened to the KEK strength.
    - Private keys can also be held and called from a secure password vault, protected by a separate encryption process.
    - Private keys should be stored in as fewest locations as possible.
- Key distribution: Private keys must be securely transmitted to and from the application. Accessibility to the private key must be strictly controlled to a "need to know" basis, and key access locked down accordingly.
- Key change\rotation (cryptographic period): Changing the private key on a regular basis is a symmetric encryption best practice, as it reduces the value of the private key to attackers. The length of time in requiring a change to the private key is called a cryptographic period, and should be set based on risk. At least two key changes per annual are considered a best practice in protecting sensitive data. Applications that manage their own encryption must provide the functionality for administrators to regularly change private keys.
- Key deletion: Private keys that are no longer used or needed must be deleted to ensure they cannot be used to decrypt older versions of encrypted personal data that might exist; that is, as held on outdated backups or past unknown lost or compromised datasets.

GDPR application security requirements

The GDPR does not provide any specific and descriptive IT security or application security controls. Instead within Article 32 it states to 'apply an appropriate level of Information Security'. The appropriate level of security for an application is determined by a combination of adhering to industry best security practices, and by adopting a risk assessed approach. The latter is covered by the Data Protection Security Impact Assessment, which is detailed in the second part of this GDPR guidance series.

Article 32 (1)*Taking into account the state of the art, the costs of implementation and the nature, scope, context and purposes of processing as well as the risk of varying likelihood and severity for the rights and freedoms of natural persons,* ***the controller and the processor shall implement appropriate technical*** *and organisational measures* ***to ensure a level of security appropriate to the risk****, including inter alia as appropriate:*
(a) the pseudonymisation and encryption of personal data
(b) the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services
(c) the ability to restore the availability and access to personal data in a timely manner in the event of a physical or technical incident
(d) a process for regularly testing, assessing and evaluating the effectiveness of technical and organisational measures for ensuring the security of the processing

Application security best practice

To comply with Article 32 (1c), assure the confidentiality, integrity and availability of personal data managed by applications, developers should follow industry recognized application security best practices, such as developing the application in adherence to the ten top principles of the Open Web Application Security Project (OWASP).

OWASP Top Ten 2017:

1. Injection
2. Broken authentication and session management
3. Sensitive data exposure
4. XML external entity (XXE)
5. Broken access control
6. Security misconfiguration
7. Cross-site scripting (XSS)
8. Insecure deserialization
9. Using components with known vulnerabilities
10. Insufficient logging and monitoring

For further detailed guidance on application security and OWASP Top Ten, review Scan your app to find and fix OWASP Top 10 - 2017 vulnerabilities.

Assuring application security GDPR compliance with IBM® Security AppScan

To comply with Article 33 (1d), assure testing, assessing, and evaluating the effectiveness of technical measures securing the processing of personal data by the application, developers should perform thorough application security testing and vulnerability scanning within the SDLC testing phase. IBM Security AppScan is a web application vulnerability scanning tool that can be used to verify and report an application's security compliance with both the OWASP Top Ten and the GDPR. For further guidance review the tutorials on AppScan and download a free trial of IBM Security AppScan.

IoT Application security GDPR compliance

Applications developed to manage Internet of Things (IoT) devices often process personal data, and therefore fall into scope of GDPR compliance. For further security guidance on IoT application development, review the article, Combating IoT Cyber Threats.

Data breach disclosure

GDPR Articles 33 and 34 emphasises the need for robust application security, given any failure with the security of the application that causes the compromise of personal data must be reported by the controller to a supervisory authority within 72 hours and to data subjects without undue delay.

Article 33 (1)*In the case of a personal data breach, the controller shall without undue delay and, where feasible, not later than 72 hours after having become aware of it, notify the personal data breach to the supervisory authority.*

Article 34 (1)*When the personal data breach is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall communicate the personal data breach to the data subject without undue delay.*

Where an application developing organization is not in a role of a controller, the developing organization is still at risk of significant public and client reputational damages, especially where weak application security has caused large-scale or severe personal data breaches.

Conclusion

There are a number of different technical solutions that can significantly mitigate an application’s privacy risk and ease an organization’s GDPR compliance. Each privacy risk mitigating solution has security weaknesses and trade-offs that need to be fully understood and considered during the design of the application. For instance, data anonymization and pseudonymization has the risk of being reverse engineered, while data encryption has the overhead of requiring robust key management processes. Once determined in design, application privacy mitigation measures must be scrutinized to ensure the application is developed to provide a standard of data protection and a level of privacy risk, which is considered to be appropriate by GDPR supervisory authorities.

The GDPR does not stipulate a descriptive set of IT and application security requirements, instead the regulation requires 'an appropriate level of information security' to be adopted. Given the substantial risk of weak application security leading to high impact compromises of personal information, including mass data breaches, developing and testing applications to proven application security practices should be considered as a benchmark towards achieving a GDPR 'compliant' application. Developing applications that adhere to the principles of the OWASP Top Ten, and testing applications for security vulnerabilities using IBM Security AppScan, are essential SDLC practices to reduce GDPR non-compliance organizational risk, and more importantly, protect the privacy of the application's users.

Related topics

- Summary of GDPR articles
- OWASP Top 10
- IBM Rational AppScan Standard product page