aepd
agencia
española
protección
datos

EDPS
European Data Protection Supervisor

INTRODUCTION TO THE HASH FUNCTION AS A PERSONAL DATA PSEUDONYMISATION TECHNIQUE

v. October 2019

EXECUTIVE SUMMARY

This essay is intended for data controllers who wish to use hash techniques in their data processing activities as a safeguard for personal data pseudonymisation. The fundamentals and properties of hash techniques are presented throughout the text. Application of such techniques may sometimes entail a high risk of identifying the message generating the hash. This document analyses the sources of risk of re-identification in application of hash techniques, and establishes the need to carry out an objective analysis of this risk in order to determine whether this pseudonymisation or even anonymisation technique is appropriate. This analysis involves both the process followed and any other elements that form the hash systems, paying special attention to message entropy and to information linked or linkable to the value represented by the hash.

CONTENTS

I. INTRODUCTION AND PURPOSE

Currently, the value of data is indisputable. Data have become a key factor for scientific research, public administration and an ever-growing digital economy. Development of promising technologies such as Big Data or Machine Learning greatly depends on being fed a large quantity of data.

This increasing demand for personal data has entailed a renewed interest in anonymisation techniques and processes. Hash functions have been used for a long time in order to provide an additional protection when processing personal data. However, there is doubt regarding to what extent hash functions constitute an efficient pseudonymisation technique, as well as whether, under certain circumstances, such as the original message having been deleted, the hash value may be even considered as anonymised[1].

This decision is of paramount importance to determine, among other things, effective compliance of the rights recognised by the GDPR in certain types of processing, such as research, traffic data analysis or geolocation, blockchain and others. Legal, technical and process management considerations are factored in when making the appropriate decision and, therefore, those involved in making this decision may need to have a basic knowledge of hash techniques and their potential risks.

This essay is addressed to data controllers who wish to use implementations based on the use of hash functions in order to pseudonymise or anonymise personal data. It briefly presents the basic aspects of hash functions, their properties and the possibility to re-identify the message generated by the hash, while also establishing certain guidelines to analyse the suitability of hash function-based processing.

II. DIGEST OR HASH FUNCTION

A digest or hash function is a process which transforms any random dataset in a fixed length character series, regardless of the size of input data.

The output is called hash value or code, digest, image or hash. Often, the term “hash” is used both in reference to the hash function as to the hash value, which is the output of playing this function on a particular message. The data that are to be run through the hash function are called the message or preimage. The set formed by all possible messages or preimages is the message domain or message space.

For example, if the hash function SHA256[2] is used to determine the hash value for “Hello” the output shall be the following digest:

SHA256(Hello) =
104123755207289804956272089233072171500574028163694787093464498998073160165519[3]

[1] Blockchain and the General Data Protection Regulation. http://www.europarl.europa.eu/thinktank/es/document.html?reference=EPRS_STU%282019%29634445

[2] You can find a hash functions similator at https://hash.online-convert.com/

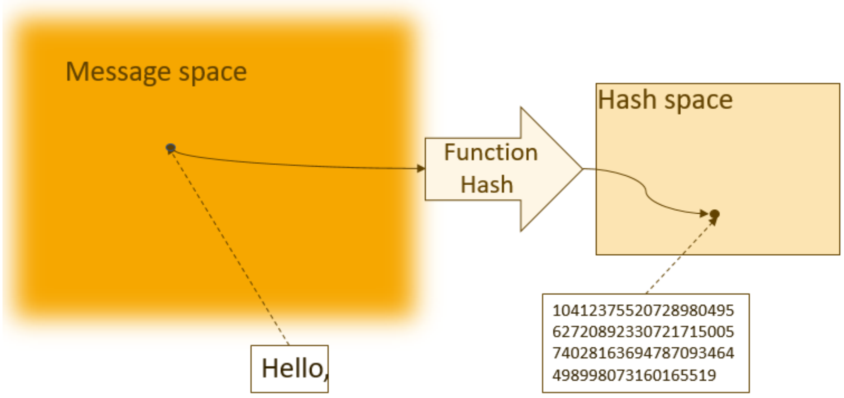
[3] Hexadecimal notation is normally used since there is a direct correspondence between digits and underlying bits, which would be lost if decimal notation was used. The information above in hexadecimal notation is: E633F4FC79BADEA1DC5DB970CF397C8248BAC47CC3ACF9915BA60B5D76B0E88F

In this example, "Hello" is translated into a set of bits, from which, after a series of operations[4], a 256-bit string is obtained (here represented by its value in decimal notation).

However, when a more complex message, for example, a pdf file containing the full text of the *Quixote* (471 pages), is run through a hash function, the output of the hash function will be different, but the output will be a digest with the same size in bits as the previous one:

SHA256 (<<PDF file containing unabridged edition of the Quixote>>) =
60306958082729627487200375730074782666835392077101919947310301380935
652572169[5]

The first message, "Hello" used 32 bits[6]; however, the second message, the unabridged edition of the *Quixote,* consisted of a document whose size exceeded 8 million bits. Therefore, the hash function may be represented in the following way:



To the left, the message space is represented, that is, all possible datasets which may be created and from which a hash may be generated. In this figure, this set is represented with blurred limits, since this space may be infinite, given that it is always possible to create a larger message.

To the right of the image, the hash space is represented as a box of smaller size and defined limits. The size of the hash space depends on the number of bits used in the hash outlet. Taking SHA-256 as an example, the outputs of this hash have a size of 256 bits, but depending on the corresponding algorithm may be of any size; the most common are 32 to 512 bits. In order to have an idea of how many different hash values may be obtained with a hash size of 256 bits, the total number shall exceed the result obtained by multiplying one million by one million thirteen times[7].

Hexadecimal notation extended representation of numbers beyond the common 0-9 notation to include 10 (A) 11 (B) 12 (C) 13 (D) 14 (E) y 15 (F). Consequently, numbers are represented by digits 0 to F. Its advantage lies in the fact that a single digit may represent 4 bits. Such 4 bits may present 16 different combinations.

[4] A very clear description of this process may be found at https://medium.com/biffures/part-5-hashing-with-sha-256-4c2afc191c40

[5] 60306958082729627487200375730074782666835392077101919947310301380935652572169

[6] In ASCII code

[7] In order to calculate the approximate value of a dataset it may be considered that only 10 bits are needed to code 1024 statuses; therefore, 20 bits yield over one million statuses (1024x1024). As for 256, when divided by 20, the result obtained is 13.

DESIRABLE PROPERTIES IN A HASH FUNCTION

The optimal properties of a hash function are:

- It may be played on digital contents of any size and format: at the end of the day, for a computer, all types of digital content (text, images, videos, etc.) are numbers.
- Any given input may produce a fixed size numerical output.
- This output is deterministic, that is; the same input message or dataset always yields the same output.
- Reconstructing the original input from the hash function output must be extremely difficult, if not outright impossible.
- A minimum variation in the original message (one bit) must yield a completely different hash (diffusion).
- Taking an input message, finding another message with the same digest must be extremely difficult (weak collision).
- Finding any two messages that yield the same summary must be also extremely difficult (strong collision).
- The hash algorithm must cover the entire hash space uniformly, which means that any output of a hash function has, in principle, the same probability of occurrence as any other. Therefore, all values in the hash space may be an output of the hash function.

DESCRIPTION OF A HASH FUNCTION

In general, hash functions work as follows:

- The input message is divided into blocks.
- Then the hash for the first block, a value with a fixed size, is calculated for the first block.
- Then, the hash for the second block is obtained and added to the previous output.
- This process is repeated until all blocks are calculated.

A very simple example of a hash function is briefly described below. Our first task is to design a hash function that, from a text, generates a three digit decimal hash (000 to 999). The message used to calculate the hash shall be the first lines of the Quixote (*In some village in La Mancha, whose name I do not care to recall, there dwelt not so long ago a gentleman of the type wont to keep an unused lance, etc.*). In Spanish:

> *En un lugar de la Mancha de cuyo nombre no quiero acordarme no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, etc.*

In the first stage, the above text would be divided into blocks of (for the purpose of this example) twenty characters. In this manner, in the following table each row represents one block:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
E	n		u	n		l	u	g	a	r		d	e		l	a		M	A
n	c	h	a		d	e		c	u	y	o		n	o	m	b	r	e	
n	o		q	u	i	e	r	o		a	c	o	r	d	a	r	m	e	
n	o		h	a		m	u	c	h	o		t	i	e	m	p	o		q
u	e		v	i	v	í	a		u	n		h	i	d	a	l	g	o	
d	e		l	o	s		d	e		l	a	n	z	a		e	n		a
s	t	i	l	l	e	r	…												

The next step would be to assign a numerical value to each character, for example: A-1; B-2; C-3; etc., finishing with zero for a space. Consequently, the following coding would be obtained for the first block:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	14	0	22	14	0	12	22	7	1	19	0	4	5	0	12	1	0	13	1

The hash value can be obtained in multiple ways, for example, by multiplying the value assigned to a character with its position within the block[8] and then adding together all results. For this particular block, the final output would be 1331. As the output of this hash function must have only three decimal digits, the output may be directly truncated removing any digits above the third. Therefore, the hash output for the first block would be 331.

Next, the following row or block would be:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
14	3	8	1	0	4	5	0	3	22	26	16	0	14	16	13	2	19	5	0

If the same hash function were played over the same block, the output would be 1947, which, once truncated above the third digit, would be 947.

Subsequently, the two blocks would be linked by means of adding the results of both blocks, 331+947 = 1278, a hash value for the first two blocks that is again truncated to 278. This process would be repeated with all rows or blocks until obtaining the final output.

The system presented here does not show good properties as a hash function: the hash value is too short (there are only 1000 possible outputs), it is possible to alter the text and maintain the hash[9], implementation is not efficient, etc. For this reason, other types of more appropriate hash functions have been developed, such as the SHA[10], the MD5[11] family or others[12].

III. HASH AS AN UNIQUE IDENTIFIER

The number of possible outputs of a hash function is very high, but is not infinite. Since the message space may be infinite, there are infinite messages that may yield the same hash value. The message set which yield the same hash value as the outcome is called the preimage set.

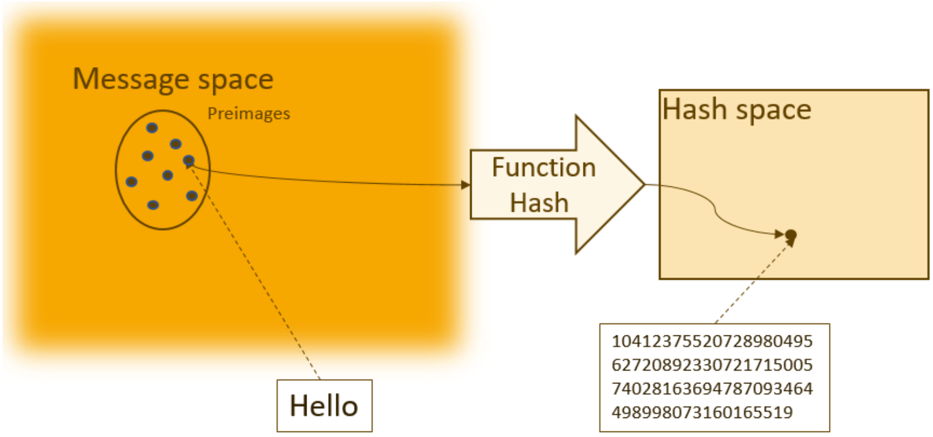
[8] This prevents obtaining the same hash when the same letters are moved around within a text, as if replacing "En un lugar de la Ma…" by "De un lugar en la Ma…".

[9] The letter with a value of 5 in position 6 gives the same result as the letter with value 6 in position 5. Using weighing such as prime factors would prevent using these equivalences.

[10] FIPS PUB 180-4 Secure Hash Standard (SHS) https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=910977

[11] RFC1321 The MD5 Message-Digest Algorithm https://www.ietf.org/rfc/rfc1321.txt

[12] https://en.wikipedia.org/wiki/List_of_hash_functions



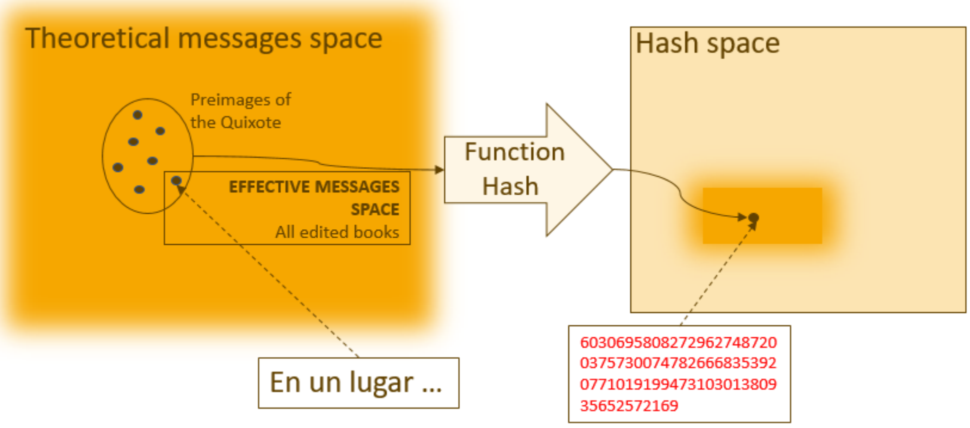
The existence of sets of preimages, which may cast a doubt over the usefulness of the hash function as a unique identifier, is considered exclusively in a theoretical environment and not for a specific processing operation.

ANALYSIS BASED ON A SPECIFIC PROCESSING OPERATION.

Let us consider that a processing activity intends to associate a hash value to each book that has been ever published worldwide, based on their digitalised contents. A few years ago, Google reported that the number of books published worldwide was 130 million[13].

This figure is significantly lower than the number of possible hash results for a function such as SHA-256. Even multiplying the total number of books by 7,000, the result of which would exceed one million million, we would be very far from saturating the hash value space.

In a graphic representation, the set of all edited books is a subset of all possible messages, and a sufficiently small subset, so that preimages of, for example, the *Quixote*, are left out.



[13] http://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html

The shadowed area over the hash spaces represents the set of hash values that would be generated by obtaining the hash for all edited books. This set would represent a tiny part of the set of possible hash values.

Although the number of edited books is high, if the square corresponding to this number were to be drawn in the figure above at the same scale as the square corresponding to the hash space, the first would be so tiny that it would be practically invisible.

The difference between the set of edited books and any other possible messages that may constitute a preimage of the Quixote lays in the concept of "order". Books are not formed by just any combination of letters, but by words in a specific language, which are structured according to the rules of specific grammar, following a narrative structure and yielding a meaningful message. Therefore, if the same hash corresponded to a specific book and to a meaningless message, the former would be discarded as not belonging to our effective message space, i.e. not belonging to our processing operation.

The stricter this "order" is (for example, in the case that only books in Spanish and not in any other language were admitted), the smaller the set of books (processing message space) will be and the less likely a collision will be.

However, there is no guarantee that two hash values corresponding to two different books do not match, although the possibility is negligible in a well-built algorithm. This likelihood may be determined through analysis by a generalisation of the birthday paradox[14]. For this reason, if within the hash space a square is set that contains all hashes corresponding to the real message space, the limits of this square will be blurred; that is, if there are a million million edited books and the hashes for all of them are obtained, it is highly probable -though not guaranteed - that a million million different hashes will be generated. However, for practical purposes, there is a bi-univocal, i.e. 1 to 1 correspondence between books and hashes.

IV. THE REIDENTIFICATION PROBLEM

Hash functions aim to be irreversible (please see section on *Desirable properties in a hash function*) and therefore the result of applying a hash function to a direct identifier should prevent the re-identification of this direct identifier. In spite of the above, the same "order" implicit in any processing activity, and which guarantees the hash effectivity as a single identifier, also increases the likelihood of identifying the original message from the hash.

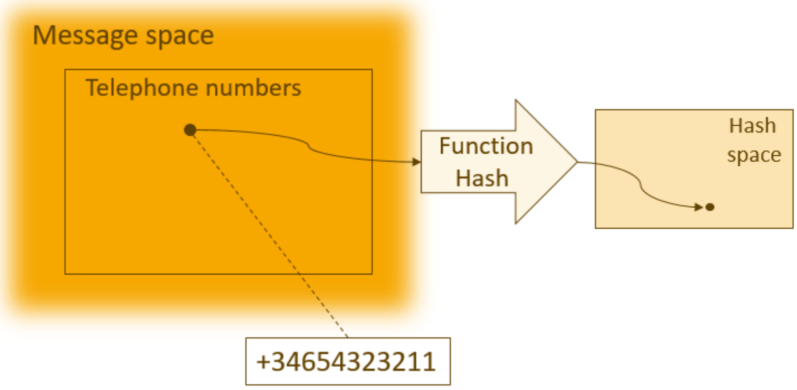
PLAYING A HASH OVER TELEPHONE NUMBERS

As an example, let us consider that as part of a processing operation, the hash of a mobile telephone number of a telecommunications company is assessed. The processing designer uses a hash function that generates a hash value with a size of 64 bits[15].

A telephone number is formed of 9 digits, plus two digits corresponding to the regional code and preceded by the "+" symbol, totalling 12 symbols. If each symbol is stored in a byte, the total number of bits is: 12 x 8 = 96 bits.

[14] https://en.wikipedia.org/wiki/Birthday_problem#Generalizations

[15] For example, Cityhash https://github.com/google/cityhash



In the above figure, it is easy to see how the telephone number space is larger than the hash space. If all possible hashes for all possible 96 bits combinations were to be calculated, the entire hash space would inevitably be covered, and more than four thousand collisions would occur. In principle, the hash function would have to compress the 96 bits of the number into the 64 bits of the hash.

On one hand, if the 96 bits of the number included information, converting this number to a 64-bit hash would necessarily imply losing information and would make the hash irreversible. On the other hand, it is possible to consider whether this processing design complies with the requirement of having a univocal identifier.

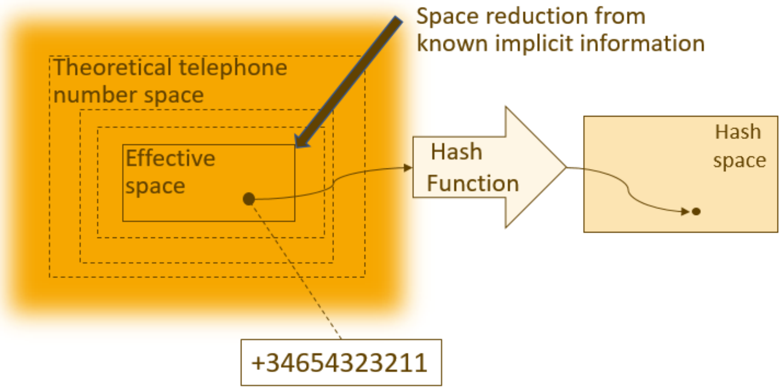
PROCESSING ANALYSIS

A more detailed analysis would provide an answer to these questions.

- On one hand, eleven of the digits are numerical, which yield 100,000 million combinations. If we consider that a keyboard only includes 20 possible symbols, the coding of an additional symbol in order to register the initial "+" would increase the number of combinations to 2 billion[16]. This may seem a fairly large figure, but, translated into bits, the amount of data has been reduced from 96 bits to approximately 41 bits. This is far below the 96 initial bits and also below 64 bits, i.e. the hash size. Therefore, it would be useful as a single identifier.
- Since the processing framework supposes that all numbers correspond to Spanish subscribers, it is known they all have the prefix +34. Therefore, these data are fixed. The other 9 digits yield one thousand million combinations (approximately 30 bits).
- If the telephone numbers to be processed are Spanish mobile phone numbers, they will begin with 6 or 7. Therefore, since the first number is fixed, there are only 200 million combinations (approximately 28 bits).
- However, there are not 200 million telephone subscribers in Spain. The current number of mobile lines does not even reach 60 million.
- The operator with the highest number of mobiles does not reach 20 million (20 bits of information). Therefore, if the mobile operator to which the telephone number hash corresponds is known, and access to the telephone list is granted, the number of combinations decreases greatly, and actual

[16] A billion here means a million millions.

information contained by the original 96 bits of data are only 20 bits of information.


Space reduction from known implicit information
Theoretical telephone number space
Effective space
Hash Function
Hash space
+34654323211

By way of an example, the definition of processing implicitly includes an order that limits the set of possible messages in the framework of this treatment (its message space). The number of possible valid messages (2^{20}) is much lower than the number of possible hashes (2^{64}). The possibility of collision is therefore very low[17] and the hash would practically serve as a single identifier.

RE-IDENTIFICATION ANALYSIS

With regard to the possibility that, from a given hash, it is possible to find out which number it corresponds to, keeping in mind that a desktop computer is capable of calculating over 1 million hashes per second it is possible to create a directory for all possible hashes for the telephone numbers of a given operator in less than 20 seconds, i.e. with practically no delay at all[18]. That is, the information referenced by the hash may be recovered.

In this case, the amount of information was small, but even for much larger message spaces, holding more information, there are techniques known as Rainbow Tables[19] that allow for the reversal of a hash.

ORDER, DISORDER AND INFORMATION

When the data used in a processing operation have an implicit order, the set of possible messages (message space) is greatly reduced, which makes message reversal (i.e. re-identification) easier.

The stricter the implicit order in a dataset, the more fixed a message is, and the less real information it contains. Knowledge is obtained from the data structure itself (e.g. Spanish mobile telephone numbers start either with +346 or +347) or from the processing environment (e.g. the telephone numbers managed by the operator are known). For these reasons, it is necessary to distinguish between the data in a message (96 bits in the above example) and the information included in this data (20 bits in the above example).

[17] Very low but not zero. If analysed as a memory-less system, the likelihood of two or more hashes matching, with 20 million occurrences as in this case, would be calculated as follows: $P=1- [2^{64}! / ((2^{64*20.000.000})*(2^{64-20.000.000})!)]$. Due to the complexity of calculating large numbers, it may be guaranteed that such likelihood is well below 0.002%.

[18] https://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance/

[19] Rainbow Tables attack: Data attack method which uses a pre-computed table of hash chains (Fixed length message digests) in order to identify the original data source. CCN-STIC 401

The degree of order or disorder in a dataset is called entropy[20]. The higher the entropy, the more information a data set contains. On the contrary, a lower disorder level (entropy) involves the existence of fewer alternatives, and therefore the same data will contain less information.

The smaller the message space and the lower the entropy are, the lower the risk of collision in hash processing is, but re-identification will be more likely. On the contrary, the higher the entropy, the higher the possibility of a collision, but the risk of re-identification will be lower.

Measuring the amount of information, which is very different from the number of bits used to record a message, is one of the most important analyses that needs to be carried out every time that a message needs to be protected, either by means of hash functions or by using any other techniques, such as encryption, and requires an accurate analysis[21].

V. LINKING INFORMATION TO A HASH

In the previous example, the more information that was available about the potential message space (its structure, the geographical location of users, their telephone operator), the higher the implicit "order" of the message was and the less information was included in the hash itself, which made reidentification more likely.

IDENTIFIERS LINKED TO A HASH

A file with personal data may include "identifiers" which are, by their own means, univocally associated to a data holder (e.g. a person's full name, ID document or passport number).

When there are identifiers linked to a hash, for example, when an ID number and a hash corresponding to a telephone number linked to that ID are stored, it is obvious that the information belongs to a certain data holder. With regard to the confidentiality of information represented in the hash, the fact of having a linked identifier adds an additional vulnerability to the existing weakness of the relevant hash, since, from that ID number, information may be obtained which reduces the effective message space for that particular hash.

That is, the more personal information that is linked to the hash, the higher the risk of identifying the contents of this hash.

PSEUDOIDENTIFIERS LINKED TO A HASH

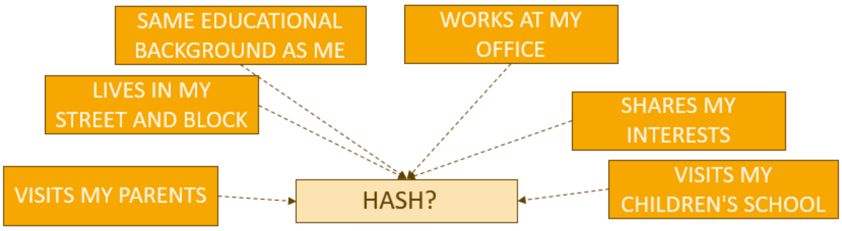
Files with personal data may also include other data which, when conveniently bundled and crossed with other information sources, may result in the successful identification of an individual. Such data are called "pseudoidentifiers", "quasi-identifiers" or indirect identifiers[22]. The relationship between these and the hash value can be established in two ways.

[20] Entropy is a principle of thermodynamics which establishes that a system's disorder only increases over time. In order to see an example of its relationship with information, let us imagine a freshly painted, perfectly white wall, which could be described in a few words: "three metres tall, five metres long, pure white". As time goes by, some areas become greyish, cracks and stains appear, etc. That is, its entropy increases. At this stage, the wall cannot be described in a few words; a lengthy description is needed to record all its flaws. Therefore, there is more information.

[21] An analysis of the information linked to processing may be carried out by analysing entropy, identifying the set of all possible statuses x(i) and its associate probability P(xi) by calculating: Information = $- \Sigma_{(1..n)} P(x_i) \log_2 P(x_i)$

[22] AEPD: K-anonymity as a privacy measure.

The first one is that the hash may be linked to pseudoidentifiers as a secondary effect that is not the purpose of the processing. The second case occurs when the purpose of the processing operation is to link pseudoidentifiers to one another by means of a hash value.


SAME EDUCATIONAL BACKGROUND AS ME
WORKS AT MY OFFICE
LIVES IN MY STREET AND BLOCK
SHARES MY INTERESTS
VISITS MY PARENTS
HASH?
VISITS MY CHILDREN'S SCHOOL

In one way or another, the information provided by these pseudoidentifiers decreases the efficiency of hash functions by providing hints on the information contained in the hash value, which may allow for the identification of the data holder. The risk of re-identification will depend not only on the type of pseudoidentifiers but also on the correct application of randomisation or generalisation techniques[23] on indirect identifiers. In order to determine the extent to which this set of information is anonymous, it would be necessary to make an analysis of k-anonymity, for example[24].

LINKS TO OTHER TYPES OF INFORMATION

There might be an additional factor that arises when the theoretical design of a processing operation manifests itself implemented in an IT system and a series of working procedures. The actual operation may create circumstances which allow for the linking of additional information to a hash, and which were not provided in the original processing plan, such as:

- The relative position of the hash in a table or data chain allows for the establishment of relationships between information stored previously or at a later date.
- The hash record date in the corresponding table allows for the linking of this hash with information included in other tables or chains completed on the same date.
- In the same way, log files associated with transmission, use of computing services, access to storing systems, etc., are information sources that allow for data crossing.

All such possibilities must be factored in for the purpose of determining the hash re-identification risk level.

VI. STRATEGIES TO HINDER REIDENTIFICATION

KEY REUSE ENCRYPTION

A strategy to hinder re-identification of the hash value is to use an encryption algorithm[25] with a key that is confidentially stored by the data controller or with the other person taking part in the processing, so that the message is properly encrypted before

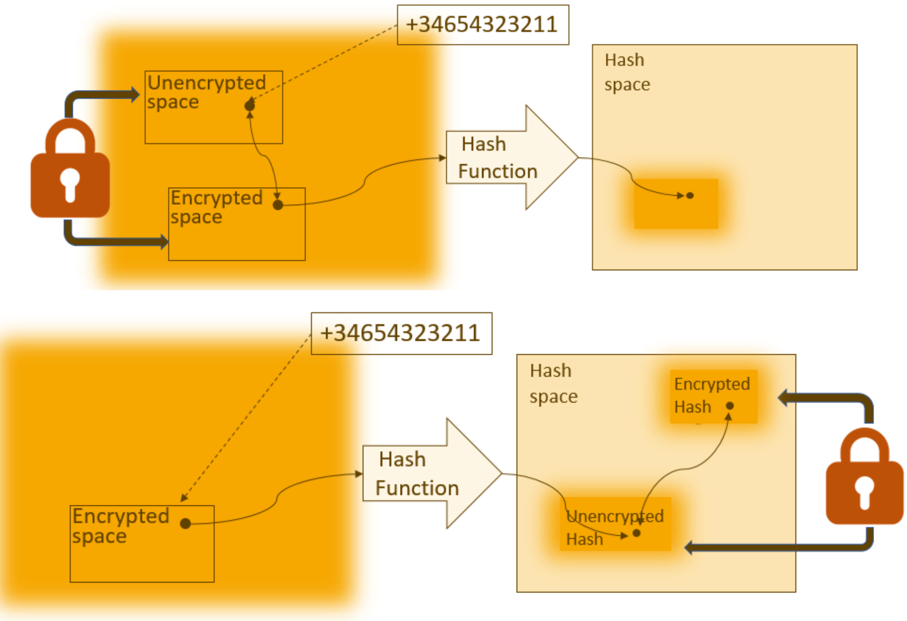
[23] WP29: Opinion 05/2014 on anonymisation techniques.

[24] AEPD: K-anonymity as a privacy measure.

[25] Whether the encryption system is symmetrical or asymmetrical is indifferent.

the hash is completed. Alternatively, the hash is encrypted once it is computed. The encryption process yields a new message (encrypted text) from the original (plain text). There is an efficient process to obtain one from the other by using keys.

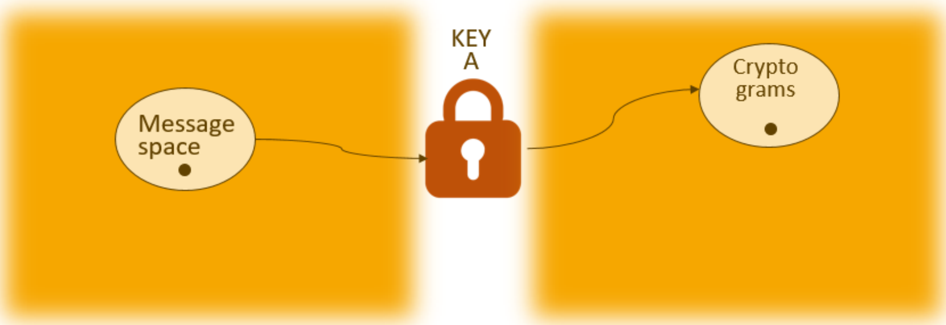
Both strategies are shown in the figure below:



As shown in the figure above, encryption achieves a kind of correspondence between different message spaces, in the first instance, or, when the hash itself is encrypted, between two hash spaces.

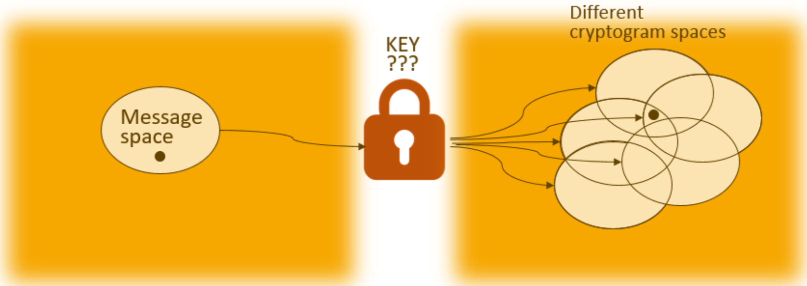
The encryption process, unlike the hash computing process, is an inherently reversible process, which, by its own nature, keeps the encrypted information. That is, information is right there, it does not increase or decrease with the encryption process and the encrypted and non-encrypted spaces are linked by the key.

In this case, we ca show the connection between message spaces and encrypted spaces (cryptogram) for a key in the following manner:

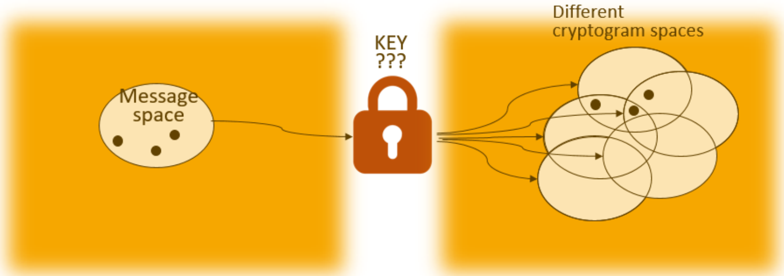


In the figure, the point between the two spaces represents the same message, unencrypted at the left and encrypted to the right.

When the key is not known, the only reference for an observer would be that there is an encrypted message, and that this message may correspond to almost as many keys as may be thought up.



However, if there are several encrypted texts from different messages, at some point all they may only correspond to a single space of cryptograms, which may be generated by a single key:



This principle is known under the name of Unicity Distance[26] and it establishes that, over a given threshold of encrypted text, both the plain text and the key are determined. This entails that, even though the key may be deleted by the data controller, the key is implicit in the encrypted text, so that neither the key nor the information it protects may disappear, and, therefore, they may both be recovered[27].

Re-identification will therefore be protected by a set level of certainty that will depend on the weaknesses inherent to any encryption system:

- Compromise of key confidentiality. Working in distributed environments may increase this risk[28].
- Generating a strong key.
- Vulnerability to attacks such as those of known plain text or chosen plain text[29].
- The volume of encrypted information: the more information, the easier it will be to carry out a cryptanalysis.
- The principle of unicity distance.
- The development of computing power and new cryptanalysis algorithms.
- The existence of unknown weaknesses in the encryption system.

[26] Established by Shannon in the report "Communication Theory of Secrecy Systems", explained in the following terms: depending on the entropy of the encryption systems, the length of the message based on which, given a specific cryptogram and encryption algorithm, both the key and the plain message are entirely determined.

[27] This distance may be very short, if Unicity Distance = Log_2(nº claves) / Redundancy = Log_2(nº keys) / ($\log_2$(symbols)-entropy) for a text in Spanish encrypted by means of AES256 unicity distance equals $\log_2(2^{256})/(\log_2(27)-2)$ that is, approximately 95 characters. That is, a single 100-character encrypted text exceeds such distance.

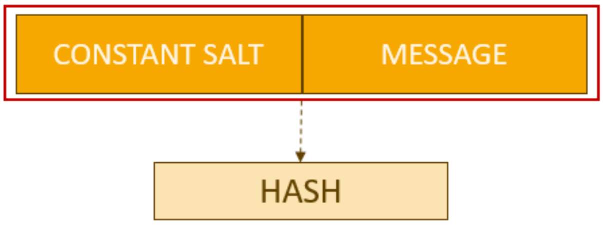
[28] In any case, this safety level may be increased by means of asymmetrical cryptosystems and key list management strategies.

[29] The first refers to those cases in which the encrypted information is public, and the second example refers to those cases in which a third party may cause a chosen message to be encrypted.

ADDING A MESSAGE HEADING OR REUSABLE SALT MODELS

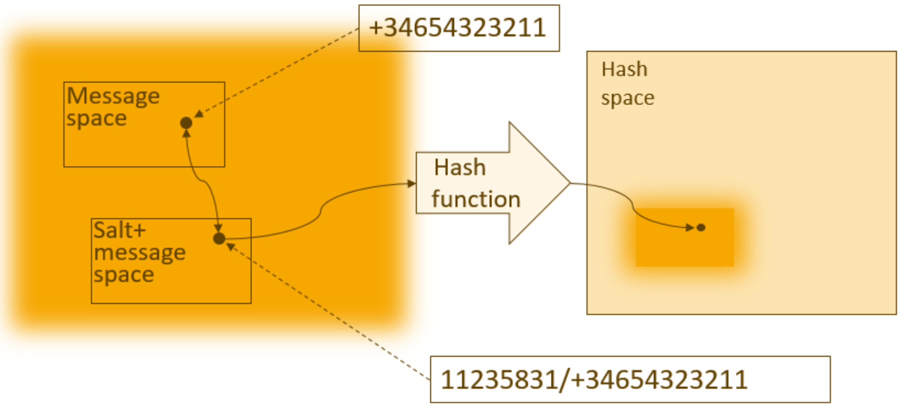
Another strategy to difficult re-identification is adding a constant value or "salt" to all messages before assessing the hash. Any random value[30] added to the original message is known as "salt". Its randomness must be independent from the same message or any other information.

The message format changes, since the "salt" field must be added to the original message. The newly extended message will have the following format:



Let us consider the case of processing hashes corresponding to telephone numbers. With this new strategy, the message space for the processing does not only comprise telephone numbers but by the pair "salt + telephone numbers". For example, before calculating the hash of telephone number "+34654323211" a random previously calculated value needs to be added to the message. 112358314+3465323211. Once the extended message has been created, the hash is computed. In order to calculate the hash for any message, only the salt stored in the system should be added to the original message. If, due to a collision, the hash of a telephone number matches a stored hash, this match will not validate the number, as isolated numbers are not a part of the extended message spaces.

As in the case of using keys, the salt value must be kept secret and the amount of information in the message space is a constant: it neither increases nor decreases. The process may be shown in the following manner:



If the salt value is removed by the data controller, the information does not disappear, but access to it is protected by a certain level of security, since an approximation of the

[30] Not all pseudorandom number generators are appropriate for this purpose; they must have some specific characteristics. These are called CPRNG, CSPRNG or cryptographically secure pseudo-random number generators. Documents on how to check the appropriateness of a CSPRNG in http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

principle of unicity distance (as it is the case with encryption) is already complied with. Although the hash algorithm is not inherently reversible, the situation, the status of the message spaces with regard to the salt+messages space is similar to the dynamics with the encryption system. That is, the salt is also implicit in the generated hash space[31].

Besides, re-identification shall be protected by a set level of certainty which will depend on the system weaknesses, including:

- The commitment to maintaining the confidentiality of the only salt used, the control of which is more difficult in a distributed environment.
- Salt properties will be dependent on their length and the random nature of the salt generation system. A shorter salt will be vulnerable to brute force attacks and Rainbow Table attacks. A salt value generation system that presents weaknesses may infer that the value of this salt is determined within a limited set of values.
- The development of computing power and of new hash disruption algorithms, which call for ever longer salts[32].
- The presence of unknown weaknesses in the hash algorithm or in the processing system.

SINGLE-USE SALT MODELS

Using single-use salts allows for the development of methods that reduce the risk of hash re-identification. In the case that the relevant salt is deleted, the original message and -only if certain guarantees are complied with - the identifier on which a single-use salt model is applied may be considered anonymised[33].

Single-use salts generate a separate random element for each message. This random element must be completely independent of any message and any other salt generated for any other message.

A very informative system for this model would be the following:

Firstly, the format of the original message is expanded to an “extended message” formed of three fields:

- The message itself.
- An additional salt, which we will call nonce[34], the sole purpose of which is to increase the message entropy to a value approximate to the number of output bits of the hash function. Of course, this nonce value must be generated randomly, be independent both from the message and from any other nonce value, i.e. it must also be single-use.
- A single-use salt value, independent from the two previous values and from any other salt value. It is therefore also single-use.

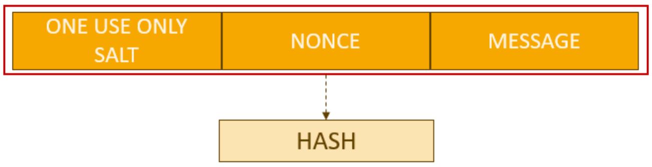
The format of the extended message for processing would be as follows:

[31] A very simplified example, which does not take into account block linking: an A hash algorithm works with k sized blocks and uses an S salt, with a size $n=k$. The attacker knows, or succeeds in computing, the hash H of a message M of a size k. $A(S \| M) = H$. Due to the division in blocks, I know that $A(S)+A(M) = H$. Therefore, $A(S) = H-A(M)$. If the S hash is known, I can develop an attack by means of a Rainbow Table or other strategies in order to obtain an equivalent salt.

[32] Considering Bitcoin mining activities in particular, which consist in obtaining hash values with certain characteristics, the SHA256 algorithm speed optimisation is undergoing an important development.

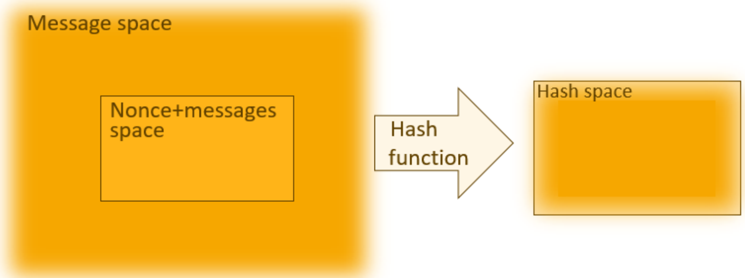
[33] See the section regarding the analysis of hash functions as a pseudonymisation in this same document.

[34] It approximates the concept of salt. Nonce is an arbitrary number, usually randomly generated, which may only be used once in the encryption scheme.

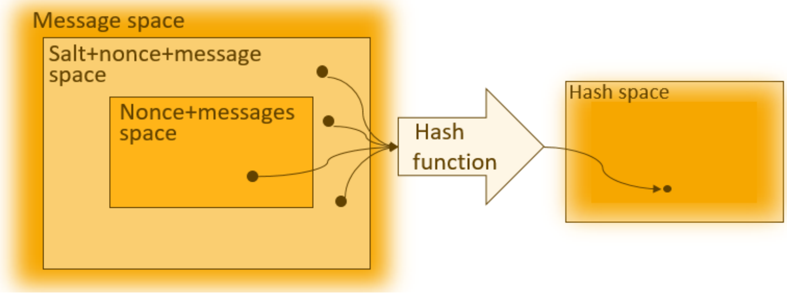


A valid message in the processing message space should have the above structure, which may be called the “extended message”. The extended message is stored by the person responsible and the salt and nonce fields are kept confidential for each of those messages. The person responsible shall have the following tasks:

- On the one hand, adding a nonce to each possible message in order to extend the entropy in the message space. A message space of the same order of the hash space would be obtained, which would ensure a highly uniform coverage of the latter, i.e. it is highly likely that any possible hash value may correspond to a valid message.



- On the other hand, since the salt is single-use and unshared, the space of possibilities (salt+nonce+messages) is increased in such a way that each hash vale has a high probability of corresponding to as many triads (salt+nonce+message) as existing hash values. That is, the possibility of collision is guaranteed.



The characteristics of this model must be as follows.

- In order to compromise re-identification, someone would need to be able to access the salt and nonce values associated with each message.
- The possibility of collision in the extended message space is guaranteed, since for the same message there is a high probability that all the messages in the original space may have the same hashes if the appropriate salt and nonce are selected.

- The hash value does not identify a single extended message within the space of possible extended messages, although the current status of technology makes it difficult to build an extended message (salt+nonce+fake message) with the same hash than that was generated from the extended message (salt+nonce+actual message). It therefore complies with its functions as a single identifier. For this reason, the salt+nonce length must be long enough.
- On the other hand, the entity that holds the triad (salt+nonce+message) may validate the hash.

In the case that this extended message is deleted by the data controller, i.e. that both the message and the salt and nonce associated with that particular message are deleted, it is not possible to validate the hash straightaway. Using single-use random information hinders re-identification, provided that there are no links between the salt of the different hashes. If the salt is long enough and the original salt has been removed, any salt validating the hash may be constructed for any message (provided that no additional information was linked to it), which would provide an association between the hash and the original message.

DIFFERENTIAL MODELS

The application of differential privacy models, and with the same preventions and conclusions, is the application of differential privacy models.

In this case, the strategy consists in adding a "noise value" to the message, which, unlike the salt which is added as a heading of the message, is incorporated to the message itself. This noise value may be applied by means of several methods and on graphic, sound or other scalable information, using differential privacy techniques in order to design and implement such techniques, such as those used in digital watermarks[35].

The noise level should comply with certain conditions to be acceptable. In the first place, it is necessary to analyse its random nature and its lack of links to the message contents. On the other hand, the lack of correlation between the noise introduced between different messages, besides ensuring a level of entropy largely over the number of bits of the hash output, and, finally, it must be verified that the status of technology does not allow processing models that allow for reidentification[36].

One of the advantages of the previous single-use salt model is the removal of a vulnerability associated with processing in blocks of the implementation of most hash algorithms.

VII. HASH ANALYSIS AS PSEUDONYMISATION OR ANONIMYSATION SYSTEM FOR PERSONAL DATA

The above paragraphs describe a theoretical approach to the operational methods of hash techniques. However, in order to assess their appropriateness to protect personal data, all elements that make up an actual processing operation have to be factored in,

[35] A digital watermark is an identification code which is directly inserted in the contents of a multimedia file (image, audio, video), so that it is imperceptible to people, but easily detectable using a given algorithm and a key in a computer. http://digital.csic.es/bitstream/10261/8864/1/Marcas_de_agua_en_el_mundo_real.pdf

[36] For example, models have been developed to prevent the noise introduced by image scanning from affecting the hash that compares the original document with a copy. Ahmad, Fawad and Lee-Ming Cheng. "Paper Document Authentication Using Print-Scan Resistant Image Hashing and Public-Key Cryptography." SpaCCS (2019).

some of which have already been described, while others depend on the implementation, such as:

- The computation of the hash itself, which in turn includes other elements:
  - Selection of a specific algorithm[37] (e.g. SHA-512, BLAKE-256 or SWIFFT).
  - The specific implementation of this algorithm in a code or circuit (e.g. OpenSSL, Bouncy Castle or Libgcrypt).
  - The type of system on which the processing is implemented, local, remote, transactional, in a cloud service, etc.
- The message spaces for the processing, considering aspects such as:
  - Its entropy.
  - Message pre-processing before playing the hash, such as adding padding[38] or random elements.
  - Related to the above, redundancy and repetitive elements in the message structure.
- Linking the hash to other information of the processing environment, particularly:
  - Information directly linked to the records in which the hash is included, both identifiers and pseudoidentifiers.
  - Information directly linked, as the correlation established between prior and later records, correspondence between record updates in the different tables, log files of all services included in the processing, as well as any other element arising from the processing implementation procedures.
- Passwords and other random elements, which may be salt or otherwise.
  - Generation, storage, mechanisms, distribution and removal mechanisms.
  - The size and entropy of these and intervention in message pre-processing, as stated above.
- Continuous management and audit of the above element, including physical safety and human factors, which are affected by technological evolution and changes in processing.

The sum of aforesaid elements includes a series of weaknesses and introduces different risk elements. The likelihood of those risks materialising increases over time due to the cumulative effect of information, detected vulnerabilities, technological development, etc.

These weaknesses affect cryptographic systems in the same way. A parallel may be drawn with the fact that the appropriateness of a cryptographic system for specific processing is determined based on the “useful life” of the message being encrypted. The useful life of a message is defined by the moment in which the relevant message loses its value[39] or relevance. An encryption system shall be appropriate for processing when it is reasonably expected to protect the message during the whole of its useful life.

In general, the useful life expectancy of personal data is as long as that of the data holder, especially when special data categories are considered. An encryption system whose requirements include reasonable protection expectations over seventy years would be an exceptionally robust system. Therefore, the requirements imposed on a

[37] https://en.wikipedia.org/wiki/List_of_hash_functions

[38] Padding techniques consist of adding data to the end of a message so it achieves an appropriate size to form a block for the hash algorithm input.

[39] Information such as the listing of certain stock or a breaking news story loses its value after a few hours, a trade secret may be valuable for a few years, while diplomatic secret information must be kept confidential for decades.

hash system in order to consider it as having appropriate pseudonymisation, or even anonymisation, techniques, are many.

Finally, it must be stated that, in order to anonymise a file, the corresponding data ‘should be such as not to allow the data subject to be identified via “all” “likely” and “reasonable” means’ by the data controller or by any third party.[40] Therefore, **anonymisation procedures must ensure that not even the data controller is capable of re-identifying the data holders in an anonymised file**. If the hash function has been implemented considering the mentioned factors, when the data controller deletes random elements introduced and added noise, data are anonymised. The natural consequence is that the data controller loses their capacity to validate the hash.

VIII. CONCLUSIONS

Opinion 5/2014 on anonymisation techniques establishes hash functions as a pseudonymisation technique.

Using hash techniques to pseudonymise or anonymise personal data must be justified by a re-identification risk analysis associated with the specific hash technique used in the processing. Such risk analysis should analyse both the hash process and any other elements that make up the hash system, paying special attention to any information that is or may be linked to value represented by the hash. This analysis must result in an objective assessment[41] of the probability of re-identification in the long term.

Independently of the risk analysis, the basic elements to consider using hash functions for protecting information are, among others:

- A high level of information entropy when establishing the hash.
- The use of single-use salt/random values.
- When appropriate, the size of a salt may exceed the size of the hash block, provided that the former is not a multiple of the latter.
- The use of appropriate random information generators for the implementation of cryptographic techniques.
- Safe access to the hash playing process.
- Zero links with identifiers, pseudoidentifiers and other information, especially in the same record and across records, tables or parallels chains.
- The regular performance of audits on the of the hash system management procedures.

In order to consider the hash technique an anonymisation technique, this risk analysis must also assess:

- The organisational measures that guarantee the removal of any information that allows for reidentification.
- The reasonable guarantee of the system robustness beyond the expected useful life of personal data.

Finally, the implementation of guarantees regarding application of the principles established by the GPDR requires a strict previous qualitative analysis in order to objectively establish its appropriateness.

[40] WP29: Opinion 05/2014 on anonymisation techniques. Section 2.1.

[41] GDPR– Cons. 76: (---) This risk must be weighed considering an objective assessment.

IX. REFERENCES

- Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC
- WP29: Opinion 05/2014 on anonymisation techniques.
- AEPD: AEPD Guidelines and Guarantees for Personal Data Anonymisation Procedures
- AEPD: K-anonymity as a privacy measure
- ENISA: Recommendations on shaping technology according to GDPR provisions. An overview on data pseudonymisation
- FIPS PUB 180-4 Secure Hash Standard (SHS) Federal Information Processing Standards Publication National Institute of Standards and Technology Gaithersburg, MD 20899-8900
  RFC1321 The MD5 Message-Digest Algorithm
- Blockchain and the General Data Protection Regulation. Can distributed ledgers be squared with European data protection law? European Parliament. European Parliamentary Research Service. July 2019
- C.E. Shannon Communication Theory of Secrecy Systems
- B. Schneier: Applied Cryptography, 2º edition. Wiley

X. ANNEXES

GDPR Extracts

Recitals 26, 28, 29, 75, 78, 85 and 156 consider anonymisation, the most relevant for the question considered in this document are:

> *26. The principles of data protection should apply to any information concerning an identified or identifiable natural person. Personal data which have undergone pseudonymisation, which could be attributed to a natural person by the use of additional information should be considered to be information on an identifiable natural person. To determine whether a natural person is identifiable, account should be taken of all the means reasonably likely to be used, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly. To ascertain whether means are reasonably likely to be used to identify the natural person, account should be taken of all objective factors, such as the costs of and the amount of time required for identification, taking into consideration the available technology at the time of the processing and technological developments. 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. This Regulation does not therefore concern the processing of such anonymous information, including for statistical or research purposes.*
>
> *28. The application of pseudonymisation to personal data can reduce the risks to the data subjects concerned and help controllers and processors to meet their*

data-protection obligations. The explicit introduction of 'pseudonymisation' in this Regulation is not intended to preclude any other measures of data protection.

29. In order to create incentives to apply pseudonymisation when processing personal data, measures of pseudonymisation should, whilst allowing general analysis, be possible within the same controller when that controller has taken technical and organisational measures necessary to ensure, for the processing concerned, that this Regulation is implemented, and that additional information for attributing the personal data to a specific data subject is kept separately. The controller processing the personal data should indicate the authorised persons within the same controller.

75. The risk to the rights and freedoms of natural persons, of varying likelihood and severity, may result from personal data processing which could lead to physical, material or non-material damage, in particular: where the processing may give rise to discrimination, identity theft or fraud, financial loss, damage to the reputation, loss of confidentiality of personal data protected by professional secrecy, unauthorised reversal of pseudonymisation, or any other significant economic or social disadvantage; where data subjects might be deprived of their rights and freedoms or prevented from exercising control over their personal data; where personal data are processed which reveal racial or ethnic origin, political opinions, religion or philosophical beliefs, trade union membership, and the processing of genetic data, data concerning health or data concerning sex life or criminal convictions and offences or related security measures; where personal aspects are evaluated, in particular analysing or predicting aspects concerning performance at work, economic situation, health, personal preferences or interests, reliability or behaviour, location or movements, in order to create or use personal profiles; where personal data of vulnerable natural persons, in particular of children, are processed; or where processing involves a large amount of personal data and affects a large number of data subjects.

78. "The protection of the rights and freedoms of natural persons with regard to the processing of personal data require that appropriate technical and organisational measures be taken to ensure that the requirements of this Regulation are met. In order to be able to demonstrate compliance with this Regulation, the controller should adopt internal policies and implement measures which meet in particular the principles of data protection by design and data protection by default. Such measures could consist, inter alia, of minimising the processing of personal data, pseudonymising personal data as soon as possible, transparency with regard to the functions and processing of personal data, enabling the data subject to monitor the data processing, enabling the controller to create and improve security features. When developing, designing, selecting and using applications, services and products that are based on the processing of personal data or process personal data to fulfil their task, producers of the products, services and applications should be encouraged to take into account the right to data protection when developing and designing such products, services and applications and, with due regard to the state of the art, to make sure that controllers and processors are able to fulfil their data protection obligations. The principles of data protection by design and by default should also be taken into consideration in the context of public tenders."

Article 4.5 of the GDPR establishes that:

'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

Besides, the following references are included in the rest of the articles:

- Article 6 Lawfulness of processing, section 4 on collecting data for a purpose other than that for which the personal data have been collected, paragraph e)

the existence of appropriate safeguards, which may include encryption or pseudonymisation.

- Article 25 Data protection by design and by default

1. *Taking into account the state of the art, the cost of implementation and the nature, scope, context and purposes of processing as well as the risks of varying likelihood and severity for rights and freedoms of natural persons posed by the processing, the controller shall, both at the time of the determination of the means for processing and at the time of the processing itself, implement appropriate technical and organisational measures, such as pseudonymisation, which are designed to implement data-protection principles, such as data minimisation, in an effective manner and to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects.*

- Article 32 Security of processing

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;

- Article 40 Codes of conduct

2. *Associations and other bodies representing categories of controllers or processors may prepare codes of conduct, or amend or extend such codes, for the purpose of specifying the application of this Regulation, such as with regard to:*

   d) the pseudonymisation of personal data

- Safeguards and derogations relating to processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes

1. *Processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes, shall be subject to appropriate safeguards, in accordance with this Regulation, for the rights and freedoms of the data subject. Those safeguards shall ensure that technical and organisational measures are in place in particular in order to ensure respect*

> *for the principle of data minimisation. Those measures may include pseudonymisation provided that those purposes can be fulfilled in that manner. Where those purposes can be fulfilled by further processing which does not permit or no longer permits the identification of data subjects, those purposes shall be fulfilled in that manner.*

EXTRACTS OF OPINION 5/2014 ON ANONYMISATION TECHNIQUES

Opinion 5/2014 on anonymisation techniques establishes hash functions as a pseudonymisation technique

> ***Hash function:** this corresponds to a function which returns a fixed size output from an input of any size (the input may be a single attribute or a set of attributes) and cannot be reversed; this means that the reversal risk seen with encryption no longer exists. However, if the range of input values the hash function are known they can be replayed through the hash function in order to derive the correct value for a particular record.*
>
> *For instance, if a dataset was pseudonymised by hashing the national identification number, then this can be derived simply by hashing all possible input values and comparing the result with those values in the dataset. Hash functions are usually designed to be relatively fast to compute, and are subject to brute force attacks. (Such attacks consist in trying all the plausible inputs in order to build correspondence tables.). Pre-computed tables can also be created to allow for the bulk reversal of a large set of hash values.*
>
> *The use of a "salted-hash" function (where a random value, known as the "salt", is added to the attribute being hashed) can reduce the likelihood of deriving the input value*
>
> *but nevertheless, calculating the original attribute value hidden behind the result of a salted hash function may still be feasible with reasonable means. Especially if the type of attribute is known (name, social number, date of birth, etc.). To add computational requirement, one could rely on a key derivation hash function, where the computed value is hashed several times with a short salt.*
>
> ***Keyed-hash function with stored key:** This corresponds to a particular hash function which uses a secret key as an additional input (this differs from a salted hash function as the salt is commonly not secret). A data controller can replay the function on the attribute using the secret key, but it is much more difficult for an attacker to replay the function without knowing the key as the number of possibilities to be tested is sufficiently large as to be impractical.*
>
> ***Deterministic encryption or keyed-hash function with deletion of the key:** this technique may be equated to selecting a random number as a pseudonym for each attribute in the database and then deleting the correspondence table. This solution allows diminishing the risk of linkability between the personal data in the dataset and those relating to the same individual in another dataset where a different pseudonym is used. Considering a state-of-the-art algorithm, it will be computationally hard for an attacker to decrypt or replay the function, as it would imply testing every possible key, given that the key is not available.*
>
> *(…)*
>
> *A major misconception when dealing with anonymisation is equating it to encryption or key-coding. This misconception is grounded in two assumptions – namely, a) that once encryption is applied to some attributes of a record in a*

database (e.g. name, address, birth date), or these attributes are substituted by a seemingly randomized string as a result of a key-coding operation like keyed hash-function, then that record is "anonymised"

(…)

Focusing only on the robustness of the encryption mechanism as a measure of the degree of "anonymisation" of a dataset is misleading as many other technical and organizational factors affect the overall security of an encryption mechanism or hash function.

Extracts from the AEPD Guidelines and Guarantees for Personal Data Anonymisation Procedures

3.8 SELECTING THE RELEVANT ANONYMISATION TECHNIQUES: KEYS

1. HASH ALGORITHMS: the usefulness of encryption algorithms for anonymising little data is indisputable. Hash algorithms are especially useful. A hash algorithm is a mechanism which, when applied to a specific data, creates a single or quasi-single key which may be used to represent a single piece of information. For example, we have a piece of information that we wish to hide or anonymise, and for that purpose, we use a hash algorithm, such as SHA1 or MD5. From applying of this algorithm to a certain data, we may obtain a key or digital fingerprint, which may be used to replace the actual piece of information. The hash algorithm generates a digital fingerprint from which it is impossible to reconstruct the original information. On the other hand, any change in the original data shall generate a different digital fingerprint, which, expressed in computational terms, means that a change in a single bit in the original information stored in a computer would yield an entirely different key or an entirely different digital fingerprint.

The hash algorithm allows to generate the same digital fingerprint from the same data or little data. However, such original data may never be retrieved from a given digital fingerprint. Confidentiality is guaranteed by the fact that this is a single-sense mathematical operation. The keys resulting from the application of a hash algorithm are usually known as "digital key" since it is understood that they univocally represent a specific piece of data or of little data.

However, a hash algorithm is not enough by itself to render anonymisation irreversible, since small text chains, such as the little data corresponding to a person's postcode, a telephone number, etc. may be easy to reidentify by means of a computer program which generates consecutive numbers and its corresponding digital fingerprints. If we wish to guarantee anonymisation of little data it is necessary to use a cryptographic mechanism which ensures that the generated digital fingerprint is kept secret. A good option is the HMAC based on RFC2014 HMAC, which may be used in combination with different hash algorithms such as, for example, MD5, and which applies on the digital fingerprint or key resulting from the hash algorithm or key in function of a secret key.

The use of HMAC in combination with non-trivial secret keys and a diligent policy of key destruction may be used to guarantee the irreversibility of the anonymisation process. When the keys used with HMAC are kept, they may be used to generate pseudonymised data, which require further reidentification, by the data holder. Hash mechanisms with secret key may be useful to mask data. However, there must be a procedure that allows for safe key removal and that *offers the possibility to certify that the procedure to guarantee the irreversible nature of the procedure has been completed.*

EXTRACTS FROM THE ENISA DOCUMENT

Hashing without key

Hashing is a technique that can be used to derive pseudonyms, but, as will be shown later in this Section, has some serious drawbacks with regard to the design goals set in Section 3.1. Still, it is a starting point for understanding other stronger techniques in the field and this is why we present it first. Moreover, hashing can be a useful tool to support data accuracy. A cryptographic hash function h is a function with specific properties (as described next) which transforms any input message m of arbitrary length to a fixed-size output h(m) (e.g. of size 256 bits, that is 32 characters), being called hash value or message digest. The message digest satisfies the following properties [Menezes, 1996]:

i) given h(m), it is computationally infeasible to compute the unknown m, and this holds for any output h(m) - i.e. the function h is mathematically irreversible (pre-image resistance),

ii) for any given m, it is computationally infeasible to find another m'≠ m such that h(m')=h(m) (2nd pre-image resistance),

iii) it is computationally infeasible to find any two distinct inputs m, m' (free choice) such that h(m')=h(m) (collision resistance). Clearly, if a function is collision-resistant, then it is 2nd pre-image resistant too.

In other words, a cryptographic hash algorithm is one that generates a unique digest (which is also usually called fingerprint) of a fixed size for any single block of data of arbitrary size (e.g. an initial identifier of any kind). Note that for any given hash function, the same unique digest is always produced for the same input (same block of data). It is important to point out that state-of-the art hash functions should be chosen; therefore, commonly used hash functions such as MD5 and SHA-1 [Menezes,1996] with known vulnerabilities – with respect to the probability of finding collisions - should be avoided (see [Wang, 2005], [Dougherty,2008], [Stevens, 2017a], [Stevens,2017b]). Instead, cryptographically resistant hash functions should be preferable, e.g. SHA-2 and SHA-3 are currently considered as state-of-the-art [FIPS, 2012], [FIPS,2015].

The above properties of hash functions allow them to be used in several applications, including data integrity and entity authentication. For instance, once an app market has a hash server storing hash values of app source codes, any user can verify whether the source code has been modified or not via a simple validation of its hash value - since any modification of the code would lead to a different hash value (see, e.g., [Jeun, 2011]). Similarly, recalling the discussion in Section 3.1 on data accuracy, a pseudonym that is generated via hashing user's identifiers may be a convenient way for a data controller to verify a user's identity. However, when it comes to pseudonymisation, despite the aforementioned properties of a cryptographic hash function, simple hashing of data subjects' identifiers to provide pseudonyms has major drawbacks. More precisely, with regard to the aforementioned D1 and D2 design goals, we have the following:

• The D2 property does not hold, since any third party that applies the same hash function to the same identifier gets the same pseudonym.

- In relation to the above observation, the D1 property also does not necessarily hold, since it is trivial for any third party to check, for a given identifier, whether a pseudonym corresponds to this identifier (i.e. though hashing the identifier).

Therefore, a reversal of pseudonymisation is possible whenever such an approach is adopted, as having a list of the (possible) initial identifiers is adequate for any third party to associate these identifiers with the corresponding pseudonyms, with no any other association being in place. In fact, following the GDPR definition of pseudonymisation, one could argue that hashing is a weak pseudonymisation technique as it can be reversed without the use of additional information. Relevant examples are provided in [Demir, 2018] (and in references therein), where the researchers refer to the Gravatar service and describe how users' email addresses can be derived through their hash value, which is shown in the URL that corresponds to the gravatar of the user, without any additional information. Hence, hash functions are generally not recommended for pseudonymisation of personal data, although they can still contribute to enhancing security in specific contexts with negligible privacy risks and when the initial identifiers cannot be guessed or easily inferred by a third party. For the vast majority of the cases, such pseudonymisation technique does not seem to be sufficient as a data protection mechanism [Demir, 2018]. However, a simple hashing procedure may still have its own importance in terms of data accuracy, as stated previously.

Hashing with key or salt

A robust approach to generate pseudonyms is based on the use of keyed hash functions – i.e. hash functions whose output depends not only on the input but on a secret key too; in cryptography, such primitives are being called message authentication codes (see, e.g., [Menezes, 1996]). The main difference from the conventional hash functions is that, for the same input (a data subject's identifier), several different pseudonyms can be produced, according to the choice of the specific key – and, thus, the D2 property is ensured. Moreover, the D1 property also holds, as long as any third party, i.e. other than the controller or the processor, (e.g. an adversary) does not have knowledge of the key and, thus, is not in the position to verify whether a pseudonym corresponds to a specific known identifier. Apparently, if the data controller needs to assign the same pseudonym to the same individual, then the same secret key should be used. To ensure the aforementioned properties, a secure keyed-hash function, with properly chosen parameters, is needed. A known such standard is the HMAC [FIPS, 2008], whose strength is contingent on the strength of the underlying simple hash function (and, thus, incorporating SHA-2 or SHA-3 in HMAC is currently a right option). Moreover, the secret key needs to be unpredictable and of sufficient length, e.g. 256 bits, which could be considered as adequate even for the post-quantum era . If the secret key is disclosed to a third party, then the keyed hash function actually becomes a conventional hash function in terms of evaluating its pseudonymisation strength. Hence, recalling the definition of pseudonymisation in the GDPR, the data controller should keep the secret key securely stored separately from other data, as it constitutes the additional information, i.e. it provides the means for associating the individuals – i.e. the original identifiers – with the derived pseudonyms.

Keyed hash functions are especially applicable as pseudonymisation techniques in cases that a data controller needs - in specific data processing context - to track the individuals without, however, storing their initial identifiers (see also [Digital Summit, 2017]). Indeed, if the data controller applies - always with the same secret *key - a keyed hash function on a data subject's identifier to produce a pseudonym, without though storing the initial user's identifier, then we have the following outcomes:*

- *The same pseudonym will always be computed for each data subject (i.e. allowing tracking of the data subject).*
- *Associating a pseudonym to the initial identifier is practically not feasible (provided that the controller does not have knowledge of the initial identifiers).*

Therefore, if only tracking of data subjects is required, the controller needs to have access to the key but does not need to have access to the initial identifiers, after pseudonymisation has been performed. This is an important consideration that adheres to the principle of data minimization and should be considered by the controller as a data protection by design aspect. Moreover, a keyed hash function has also the following property: if the secret key is securely destroyed and the hash function is cryptographically strong, it is computationally hard, even for the data controller, to reverse the pseudonym to the initial identifier, even if the controller has knowledge of the initial identifiers. Therefore, the usage of a keyed hash function may allow for subsequent anonymisation of data, if necessary, since deleting the secret key actually deletes any association between the pseudonyms and the initial identifiers. More generally, using a keyed hash function to generate a pseudonym and subsequently deleting the secret key is somehow equivalent to generate random pseudonyms, without any connection with the initial identifiers. Another approach that is often presented as an alternative to the keyed hash function is the usage of an unkeyed (i.e. conventional) hash function with a so-called "salt" – that is the input to the hash function is being augmented via adding auxiliary random-looking data that are being called "salt". Again, if such a technique is appropriately applied, for the same identifier, several different pseudonyms can be produced, according to the choice of the salt – and, thus, the D2 property is ensured, whilst the D1 property also holds with regard to third parties provided that they do not have knowledge of the salt. Of course, this conclusion is valid only as long as the salt is appropriately secured and separated from the hash. Note that, as in the case of keyed hash, the same salt should be used by the controller in cases that there is need to assign always the same pseudonym to the same individual. Moreover, salted hash functions can be utilized in cases where the controller does need to store the initial identifiers, while still being able to track the data subjects. Last, if the salt is securely destroyed by the controller, it is not trivial to restore the association between pseudonyms and identifiers. However, it should be stressed that in several typical cases employing salts for protecting hashes has some serious drawbacks:

On one hand, the salt does not share the same unpredictability properties as secret keys (e.g. a salt may consist of 8 characters, i.e. 64 bits, as in the cases of protecting users' passwords in some Linux systems). More generally, from a cryptographic point of view, a keyed hash function is considered as more powerful approach than a salted hash function. There exist though several cryptographically strong techniques for generating salted hashes, which in turn could be considered as appropriate candidates for generating pseudonyms – a notable example being the bcrypt [Provos, 1999].