To: The European Data Protection Board (EDPB)

14/3 2025

**Comments and suggestions for improvements of Guidelines 01/2025 on Pseudonymization**

As a citizen of the European Union, I have a strong interest in ensuring that the rules and guidelines governing personal data protection are both effective and practical. As a privacy professional, I have spent my entire career working with cryptography, information security, and privacy. For the past 6 years, my focus as founder and CEO of the Danish company PII Guard has been on privacy-enhancing technologies, such as pseudonymization, and technical methods for enhancing privacy protection.

Beyond privacy itself, I have a deep interest in helping to strike the right balance between protecting personal data and enabling the use of data to create business value and drive innovation. The European Union is under pressure, in part due to the regulatory burden placed on businesses. While strong privacy protection is essential, we must also ensure that regulations do not unnecessarily hinder competitiveness and economic growth. Striking the right compromise between privacy and commercial interests is crucial to reducing this burden. Fortunately, I believe this balance is achievable – if we take a smart and pragmatic approach.

In my work, I have engaged with numerous companies, organizations, and professionals responsible for handling personal data. A recurring challenge I have observed is a widespread difficulty in understanding and appreciating pseudonymization – both in terms of its privacy-enhancing properties and its legal standing under the GDPR. This lack of understanding often leads organizations to avoid implementing pseudonymization altogether. Instead, they rely on legal departments or DPOs to construct justifications for why their existing practices are sufficient – effectively "writing their way out" of adopting technical safeguards like pseudonymization.

Against this backdrop, I greatly appreciate the EDPB’s initiative in developing comprehensive guidelines on pseudonymization. These guidelines have the potential to bridge the gap between legal compliance and technical implementation, encouraging organizations to adopt practical and effective privacy-enhancing measures. The following comments aim to support this effort by identifying areas where the guidelines could be further refined and strengthened to provide clear, actionable, and balanced guidance for organizations navigating the complexities of pseudonymization.

## Pseudonymization with type preservation

*(section 3 in general)*

Pseudonymization can be performed in a way where the pseudonym is obviously a pseudonym (e.g. replacing the phone number "+45 70203020" with "phone#004532") or in a way that preserves its type characteristics (e.g. replacing the phone number "+45 70203020" with "+49 76002243065"). The latter allows pseudonyms to be used in lieu of the original data, and thus the pseudonyms to be stored and processed in exactly the same way as the original data, thereby significantly reducing the friction of working on pseudonymized data.

Type preservation is of particular importance when generating test data, as the behavior and structure of the original data have to be preserved in all details in in the output data in order for the data to be valid for test purposes.

I suggest that the guidelines include information about the possibility of generating pseudonymized data with preserved format to facilitate a wider adoption of pseudonymization.

It should also be noted that type preservation cannot in most cases be achieved using hash/MAC functions, as the output domain is too small, which increases the risk of unintended collisions. Keep in mind that only one collision is enough to ruin a dataset. Type preservation in reality requires bijective (i.e., strict 1:1) methods, such as symmetric cryptography.

## Acknowledge related types of privacy-enhancing technology for achieving the effects of pseudonymization

*(section 3.1.2)*

I encourage classification that the term pseudonymization is to be understood in a broad sense. The guidelines already mention classical table-based pseudonymization and schemes built on hash/MAC functions. Advanced cryptographic approaches, such as FPE (Format-Preserving Encryption) and TPE (Type-Preserving Encryption) should also be mentioned. These algorithms have the same functional outcome, i.e. a 1:1 substitution from original values to pseudonyms, but typically rely on cryptographic methods instead of lookup tables.

TPE and FPE have several advantages over table-based pseudonymization because they compute the replacement value instead of looking it up in a table:

- **Improved security:** Eliminates the lookup table, which itself contains personal information, and instead relies on cryptographic keys, which organizations have extensive experience handling securely.
- **Type-preservation:** Partial for FPE, full for TPE
- **Stronger adaptability in distributed setups:** It is easier to exchange and maintain keys between systems, whereas the distribution and maintenance of pseudonymization databases can be troublesome and potentially pose a security risk.
- **Improved performance:** Database lookups are generally slow, whereas algorithmic approaches require only a limited number of clock cycles and no network or storage I/O to perform a transformation.

I would like to emphasize that algorithmic methods, like pseudonymization in general, require careful design and consideration. Our research has shown that many methods with type preservation, in particular FPE when used on values of varying length and/or data types, will leak significant amounts of information about the original data. If relevant, we would gladly share details of our findings and documentation. For your convenience, I have attached a paper describing our findings.

A pseudonymization scheme based on lookup tables may also have these vulnerabilities, depending on how the pseudonymized values are generated. TPE has been designed to overcome format leak problems and thus offers a safer pseudonymization approach, regardless of the data type. I recommend including a word of caution in this regard in the recommendations.

## Hash and MAC functions

*(section 3.1.2, paragraph 89)*

It is well known that designs based on hash functions must be implemented with great care. In particular, if the input domain is relatively small – as is the case for for example phone numbers and social security numbers – this allows for a brute-force search to determine which input value led to a given pseudonymized value. Or, put differently: Hash functions are impossible to reverse directly, but they can be brute-forced by computing all possible values, making decryption practically feasible. An appropriate design involving a salt and/or a secret key can often mitigate this issue.

Furthermore, if the output of the hash or MAC function is truncated (to fit a limited-size data field or to be encoded in such a way that it resembles a known data type, e.g., a phone number), the resulting transformation function will likely have a high probability of collision, making it unsuitable for pseudonymization.

I therefore recommend nuancing the statement: "preference should generally be given to one-way functions [..]" and highlighting their challenges. Perhaps even recommending against the use of hash functions due to their non-intuitive security properties and risk of collisions.

**Cryptographic keys**

*(section 3.1.2)*

Most organizations have established secure practices for handling cryptographic keys, with appropriate technical, organizational, and legal controls in place. By contrast, pseudonymization tables are less common, meaning best practices are less developed.

I also argue that it is typically easier for an attacker to gain access to the contents of a database table than to a cryptographic key. Thus, pseudonymization tables may introduce an additional risk.

I therefore suggest recommending key-based approaches over table-based approaches.

**Emphasize the importance of protecting data copies**

*(new example of application of pseudonymization)*

Most organizations maintain several copies of their data beyond their primary production data. These include data in test environments, data warehouses, AI training datasets, and datasets shared with partners or vendors. The more copies that exist, the greater the risk of data leaks, especially since non-production environments typically have weaker security controls than production systems. Many recent high-profile data breaches have originated from non-production datasets storing non-protected copies of production data.

In most cases, non-production environments do not require exact production data. Their purpose is typically not to process personal data but to analyze trends, test functionality, or train models on generalized patterns. For example:

- **A test environment** requires data with representative characteristics but does not need real identities.
- **A data warehouse** is designed for population-wide analysis and statistical insights rather than for drilling down into individual records.
- **AI training datasets** often focus on behavioral patterns rather than specific individuals.

Given these use cases, non-production environments can and should be loaded with pseudonymized data instead of production data. In fact, to comply with GDPR requirements on data minimization, purpose limitation, and security by design and default, pseudonymization should be mandatory for non-production datasets.

To illustrate this, an example in the annex would be beneficial. It could demonstrate how pseudonymized data remains functional for secondary purposes by using type-preserving pseudonymization while ensuring that datasets remain segmented and unlinkable. For instance, using Type-Preserving Encryption (TPE) with different keys for different datasets ensures that even if an attacker gains access to multiple environments, data from different sources cannot be correlated or re-identified.

**Other technical recommendations**

I have seen several times that security systems are designed at a certain point in time, but as the surrounding infrastructure evolves, the security systems are not always updated accordingly. I suggest emphasizing that the design of the system and the underlying data mapping must be kept up-to-date. Ideally, automated data mapping should be performed periodically using an appropriate tool, with detected differences being reviewed and updated by the data owner.

As mentioned in the introduction, I often see that legal departments and DPOs attempt to "write their way out" of adopting technical safeguards like pseudonymization. I believe this is rarely done out of bad faith, but often out of desperation. Many find it hard to navigate the intersection between technology and legal requirements. Many are unfamiliar with the available privacy-enhancing technologies and fear that implementing them will ruin the business value of their data (which it typically won't). Many also struggle with the perceived complexity and costs (which are often exaggerated).

I sincerely hope these guidelines will clearly convey the importance, legal requirement, and benefits of using pseudonymization and associated privacy-enhancing technologies, so that we can achieve more "real" protection instead of mere "paper" protection of our personal data. And if we can at the same time enable more innovation and business in the EU, that would be fantastic.

I encourage the EDPB to consider these clarifications and to focus on striking the right balance between privacy, security, and business needs. Too much regulation stifles innovation, while too little weakens data protection. By refining these guidelines – particularly by emphasizing the benefits of using privacy-enhancing technology in terms of compliance, security, and better access to data – the EDPB can support both effective privacy safeguards and a fair, competitive data ecosystem.

If you have any questions, need additional information, or would otherwise benefit from further dialogue, please do not hesitate to contact me.

Best regards

Martin Staal Boesgaard
M.Sc.
Citizen of Denmark
CEO and Founder of PII Guard

# Format-preserving information leakage – A severe threat to data anonymization

Martin Staal Boesgaard

PII Guard, Korskildelund 6, 2670 Greve, Denmark https://piiguard.com

**Abstract.** Anonymization- and de-identification technologies are often used to manipulate data such that the data can be used for more purposes or be shared with other parties without risk of exposing personal information stored in the data. Anonymization and de-identification often strive at delivering anonymized data with the same properties as the original data, including formatting, so that the anonymized data behaves like original (non-anonymized / not de-identified) data such that it can be processed in place of original data, using the same methods and tools. It is well known that anonymization and de-identification is a difficult art, and that protected data sets often can be re-identified for example by matching with other data sets.
We share our discovery of a new class of methods that can be used to re-identify persons in anonymized data sets: Many types of data, such as person names, addresses, e-mail addresses, and names of medical diagnoses and treatments, have a varying length and a variety of format properties, which are preserved by many anonymization methods. This kind of information turns out to be very helpful in re-identification attacks. By using real-world data, we have measured that $\gg 22\%$ of the persons in the data set can be uniquely identified when their names and addresses are stored under protection by a format-preserving anonymization method. And the remaining persons will fall into groups of very few candidates. This information leakage causes such anonymization to be essentially of not effect.

**Keywords:** Anonymization · de-identification · de-anonymization · re-identification · format-preserving encryption · pseudonymization · data masking · data format · privacy protection · k-anonymity · data leak prevention

## 1 Introduction

We have identified a severe information leak present in many anonymization / de-identification schemes, which (alone or combined with other methods) can be used to re-identify persons in anonymized / de-identified data sets.

With over 2.6 billion personal records breached in 2021–2022 [2] and an exponential growth in generated data, strong and reliable methods for anonymization and de-identifications are essential for safeguarding personally identifiable information against falling into the hands of cyber intruders.

An example to illustrate the severity of the leak presented in this paper: Imagine that an attacker has gained an anonymized copy of a data set relating to 10 million persons. This data set contains name, address, and e-mail address of each person in masked, pseudonymized, or format-preserving encrypted form. Only by utilizing the weakness presented in this paper, the attacker will be able to uniquely identify most of the persons in the data set only using the knowledge of the real person's name, address, and e-mail address using a trivially simple method. To be able to identify most persons in such a large data set uniquely with such little effort is disastrous.

The leak materializes itself in situations where anonymization is performed by replacing one piece of text with varying length and/or format with another text of same length and/or format. This is typically the case for anonymization methods such as pseudonymization, format-preserving encryption, and redaction/masking. Sensitive types of data include names, addresses, and e-mail addresses. But also less obvious types, such as dates and IP addresses, may be sensitive. The presence and severity of the leak depends on which anonymization methods are used and how they are implemented and configured.

In this paper, we use the terms "anonymization" and "anonymization methods" in a wide sense where it includes methods such as pseudonymization, format-preserving encryption, and data masking. The discussed anonymization methods may be building blocks together with other to achieve "full anonymization". The anonymization methods described in this paper can be used with different configurations. The choice of configuration can significantly affect the severity of the problem presented in this paper. We assume a configuration like the one described in the next section, which to our experience represents the vast majority of the user-cases where protection of personally identifiable information is performed through format-preserving encryption and pseudonymization as well as a significant part of the use-cases where redaction is used.

## 2 Format-preserving anonymization methods

Several methods for anonymizing data operate on text level where they generate an anonymized output text having a format with similarities to the input text. The rationale for such designs is that anonymized data maintain integrity and facilitate data integration since it can be stored and processed in place of original data without changes to file formats or processing methods. The anonymized data *appears* to be correct even though it has been anonymized.

Figure 1 above illustrates how an anonymization method with format preservation anonymize an e-mail address. It generates an output which has the same type of characters at the same positions as in the input. In particular, '@' and '.' are transferred unchanged from the input to the output since they have an important role in ensuring that the output is a valid e-mail address.

The format-preserving anonymization methods are in general characterized by:

- Taking a text string as input and generating a text string as output.

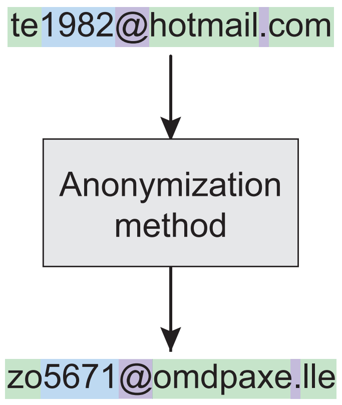

**Fig. 1.** Preserving format in anonymization. Note: Even though the text changes from the input object to the output object, the characters at a given position preserves its type: letter remains letter, number remains number, etc.

– The input text string and the output text string have the same length.
– The type of character at a given position of the input text string determines the type of character at that position of the output text string. A character on a given position in the input is replaced by a character at the same position in the output, similarly, digits are replaced by digits.
– Special characters (for example '@', '.', and '-') are preserved unchanged in the output since they typically have an important role in preserving an object's format.

Note that most methods have some degree of freedom in their configuration to be able to generate output of certain types. For example, it may in some cases be relevant also to distinguish between capital and non-capital characters.

### 2.1 Pseudonymization

Pseudonymization is a method for replacing text strings containing Personally Identifiable Information (PII) or other sensitive information with alternative text string, where a map defines the relations between the input strings and the output strings. The use of a map for translation ensures that if the same input is met several times, then it is always translated into the same output. In this way, relations between records can be preserved. The map may be defined beforehand and/or new input and output values may be added when an input string not already in the map is met in the input data. It should be ensured that output values in the map are unique. Pseudonymization is in practical cases often configured according to the characteristics mentioned in the previous section but may in some cases be configured otherwise.

Pseudonymization is in many practical applications configured to be format-preserving. But may also be configured to not be.

### 2.2 Format-preserving encryption

Format-preserving encryption (FPE) is a method for encrypting text string so that the encrypted version of the string has the same format as the original version.[1, p. 35] This contrasts with most other encryption methods, such as AES, which cannot encrypt an arbitrary input text string into a text output string of the same or similar format.

FPE can typically operate with or without a tweak. Without a tweak, it can preserve referential integrity in an encrypted data set, similar to pseudonymization. With a tweak, each string is encrypted in a unique way, whereby referential integrity is not preserved (but may more secure). It should be mentioned that the problems described in this paper also apply to FPE with tweak.

In its typical configuration, FPE exhibits all the previously defined characteristics of format-preserving anonymization (no matter if tweak is used or not).

### 2.3 Redaction

Redaction of text (sometimes called masking) is the process of selectively removing or obscuring PII or sensitive information from a text. For example,

> Mr. Smith (social security number 1234567890) has been treated for diabetes.

can be redacted into

> Mr. Xxxxx (social security number 0000000000) has been treated for xxxxxxxx.

Redaction can be configured in many ways. It is common to see that redaction exhibits all the previously defined characteristics of format-preserving anonymization.

## 3 The format-preserving information leak

Preserving the format of a text string when anonymizing it has obvious practical advantages, which is probably the reason why it is so commonly used. But it also leaks information about the input to the output as illustrated in figures 1 and 2.

It should be noted that the leak presented in this paper is not due to an error in the design or implementation of anonymization algorithm as such. In particular, for format-preserving encryption, this is not a cryptographic attack. Figure 2 illustrates that the format information bypasses the anonymization algorithm. So, no matter how strong the encryption or randomization may be, it will leak format information.

Realizing that letters are mapped to letters, numbers to numbers, and special characters are preserved, a string of anonymized text reveals the length and what kind of characters are at which positions in the original text.

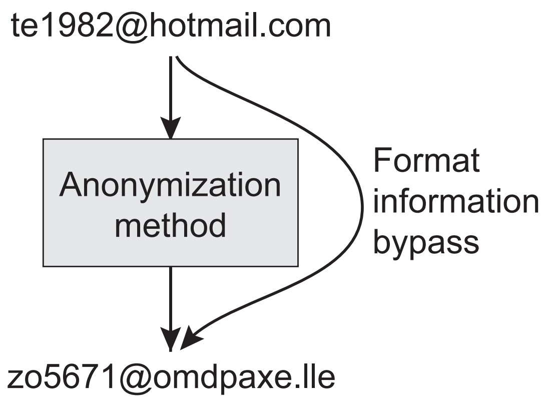

**Fig. 2.** Text goes through anonymization whereas format goes around. Since the format is the same before and after anonymization, an amount of information regarding the original data is leaked to the anonymized output data.

It must be assumed that an attacker knows in advance or can realize from observing a protected data set which data entries are processed with a format-preserving method. Security should not be based on the assumption that an attacker does not know or understand how a security system has been constructed.

The problem applies to all anonymization methods based on text strings with varying length and/or varying positions (or presence) of format-defining characters. Obvious examples of vulnerable data types are e-mail addresses, names, and postal addresses.

### 3.1 Reidentification via combination with other data sets

In classical reidentification, an anonymized data set is matched with publicly available data sets (or data otherwise available to an attacker), to deduce which data records in the anonymized data belongs to a given person.[5]

Previously, explicit information could be compared. For example, if the anonymized data set contains the postal code and the birth data of a person, these could be compared. If an anonymized data set also contains anonymized data where format is preserved, the format information can be used to rule out candidates when there is no match in the same way as for example knowledge of a birth date can be used to rule out candidates with no match.

### 3.2 Effect on $k$-anonymity

$k$-anonymity[4] is a measure used to assess the strength of anonymization efforts by putting a number on how precisely a person can be identified in an anonymized data set. More precisely, $k$-anonymity means that a person in a data set cannot be distinguished from at least $k-1$ other persons in the data set.

In the case where an anonymized data set contains data with preserved format, an attacker can use this information to re-identify a person or a group of persons or use it together with other method to achieve more precise re-identification that if using the other methods alone. Thus, it is essential to consider preserved format information when computing the $k$-anonymity of an anonymized data set.

An example: If we imagine a data set with information on the entire population of Denmark (about 6 million people), this data set contains for each data record the person's date of birth in readable form and the e-mail address encrypted via format-preserving encryption. Since the average number of births in Denmark was 157 per day in 2023, a simple estimate of $k$-anonymity of the data set only considering the birth date would be $k = 157$. In reality, $k$ would be lower due to uneven distribution of persons with a given birth date, but we ignore that for now. If you also consider the knowledge presented in this paper, we gain 10.2 bits of information by knowing the format of the e-mail address. That amount of information is enough to reduce the number of candidates in a group to $2^{-10.2} \approx \frac{1}{1000}$ of its previous number, whereby we can conclude that the vast majority of persons in the data set would be uniquely identifiable based on date of birth combined with format of e-mail address. Thus, $k = 1$, which means that the anonymization in this configuration is absolutely useless.

### 3.3 Modelling the nature of the leak

To better understand and analyze the format-preserving leakage problem, we need a model to describe the phenomenon. This model should incorporate the observation of certain properties being preserved through the anonymization process.

Our proposed model is to walk through the input string character-by-character and translate the characters using table 1 into a resulting format properties string.

**Table 1.** Mapping from actual character into format properties character. By processing each character of a string, the string's format properties string can be defined.

| **Character group** | **Characters in group** | **Character in model** |
|---|---|---|
| Letter | 'a'–'z', 'A'–'Z' | a |
| Number | '0'–'9' | d |
| Format-defining character (and other bypassed characters) | ' ', '.', '@', '-', etc. (depends on data type) | unchanged / preserved |
| Other characters | Other characters | x |

For reference, the text string "info@piiguard.com" translates into "aaaa@aaaaaaaa.aaa".

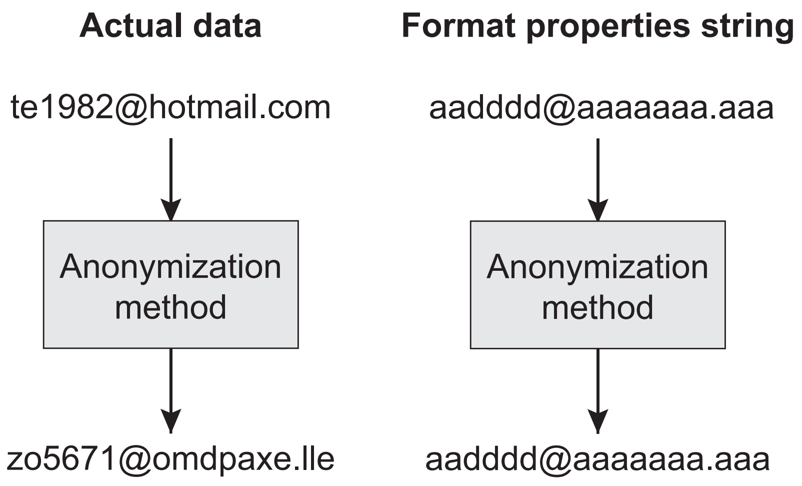

**Fig. 3.** Whereas actual data is processed and changed by the anonymization method, the format properties are not changed and thus bypasses the anonymization method.

Figure 3 illustrates how original text strings and anonymized test strings can be run through the format properties model and will yield the same results. This means that the anonymization methods are transparent to the format properties string, i.e. if you know one (input format or output format) then you also know the other. To achieve this transparency is the entire point of the design of the format properties string.

As previously mentioned, the anonymization methods in question can be configured in different ways. The same algorithm may even be configured differently for different types of data (e.g., to handle e-mail addresses with one set of rules and person names with another). In case a different configuration is used or even where several different configurations are used in the same data set, this can typically be identified by an attacker by analyzing the anonymized data and investigating the properties of the anonymized data. For example, if an anonymized person's name is "Omdraq Xtdme Axlgdagrd", then the model seems to also preserve distinguishing between capital and non-capital letters, whereas if the anonymized name is "omdraq xtdme axlgdagrd" (all non-capital) or even "OmDRaq xTdME AxlGDaGRD" (random capital and non-capital), then the model does not seem to preserve distinguishing between capital and non-capital letters. The model for defining a format properties string can be adapted to correspond to a specific configuration. For example, if capital and non-capital letters are distinguished, these can be represented by 'A' and 'a' respectively instead of just 'a' collectively.

## 4 Amount of information leaked in format

With the model described above, we achieve that anonymization methods preserving the format becomes transparent to the model. I.e., an input object having a given format will after anonymization have the same format. This is illustrated in figure 3.

In order to assess the real severity of the leak, we have managed to gain access to real customer data containing PII. Here, we can measure different properties as described in the following.

### 4.1 Acquiring and processing data

Two Danish financial organization have supported this work by running our analysis scripts on a subset of their customer database and returning statistical data. We are highly appreciative of their assistance in making this research possible.

The combined sample set represents 154 449 persons. Statistics were collected for a number of different data types, of which we will focus on e-mail addresses, person names, postal addresses, and city names in this paper. For each type of data, the data was first filtered to remove empty values and obvious errors (it is common that data is “dirty” and has misplaced or illegal data in it).

The analysis script executed on the data generated a format properties string for each text string processed. For each data type, a map mapping from “format property string” to “count of occurrences” was maintained. After processing, each map was written to a text file and the text file was shared with us for further processing.

The amount of information revealed by the format properties is calculated for each data type. The calculation is based on Claude Shannon’s concept of information entropy[3] via the equation $H(X) = -\sum_{x \in \chi} p(x) \log_2(p(x))$.

There may be doublets in the data, i.e. the same person may be represented more than once, since two independent data sets are merged, and a person may be represented in both data sets. However, this may also be the case for data from a single data set, depending on its application, if explicit care has not been taken to prevent this. Ideally, larger data sets are preferred in order to get more accurate computations (in particular of the entropy).

### 4.2 Results

**E-mail:** Out of a total of 59 243 e-mail addresses, 7 723 different format property strings were encountered and 4 100 of these were only met once (6.9% e-mail addresses were unique). The entropy of the distribution is calculated to 10.2 bits.

**Name:** The data sets had 144 916 names (first name and last name, sometimes one or more middle names or initials) divided between 31 443 different format property strings. 23 249 format property strings (16.0% of all names) were unique and the information content was 11.6 bits. The most frequent format properties string was “aaaa aaaaaaaaa” with 2 801 occurances = 1.9% of all records.

**Address:** The data set contained 144 803 addresses (street name, street number, and sometimes floor and door on floor info). These were distributed between 20 519 different format property strings of which 12 557 (8.7%) were only encountered once. The most frequent format properties string: “aaaaaaaaaa dd“ was seen 4 171 times = 2.9% of all records.

**City:** The data set contained 83 469 city names as part of the postal code. As expected, the number of format property strings were quite low as the number of cities in postal codes is limited. 167 different format property strings of which 27 (0.0%) were only met once.

The results are summarized in table 2.

**Table 2.** Statistical results from format property analysis of e-mail addresses, names, addresses, and city names. For each of the types, the total number of values and the number of different and unique values are counted and the entropy of their distributions are calculated.

| **Type** | **Total** | **Different** | **Unique** | **Unique %** | **Entropy** |
|---|---:|---:|---:|---:|---:|
| E-mail | 59 243 | 7 723 | 4 100 | 6.9% | 10.2 bits |
| Name | 144 916 | 31 443 | 23 249 | 16.0% | 11.6 bits |
| Address | 144 803 | 20 519 | 12 557 | 8.7% | 10.8 bits |
| City | 83 469 | 167 | 27 | 0.0% | 5.2 bits |
| Aggregated | 154 449 | - | $\gg$ 33.916 | $\gg$ 22% | 11.6–17.2 bits |

The four types of data may not represent independent variables. For example, a person may use their name or part of it as their e-mail address. And likewise, a postal address may have different probability distributions in different cities. To ensure that our numbers are conservative, we consider name and e-mail address as dependent and thus only use the highest value of the two. And the same for city and address. The two groups are assumed mutually independent.

**Aggregated unique percentage:** A lower bound for aggregated unique percentage is computed as $\frac{23\,249+12\,557}{154\,449} - \frac{23\,249 \cdot 12\,557}{154\,449^2} = 22.0\%$ or 33 916 record. This computation estimates the percentage of records where at least one value is globally unique. It assumes that only the percentage from name and address can be used and that these are independent variables. This computation does not take into account that a combination of otherwise non-unique values together may be unique. Thus, this computation is considered as very low lower bound.

**Joint entropy:** Joint entropy is greater or equal to the maximum individual entropy and less or equal to the sum of all entropies. Since the data set being analyzed has a limited number of records, the total entropy can obviously not exceed the information needed to uniquely describe a record in the set, which is $\log_2(154\,449) = 17.24$ bits. Considering a measured entropy of 11.6 bits for the name and a maximum entropy of 17.2 bits for describing any record in the data set, one only needs 17.2 bits − 11.6 bits = 5.6 bits for describing all records. As discussed previously, address and city can be assumed as independent variables and address has a measured entropy of 10.8 bits. Thus, it is plausible that the total entropy of name and address will be quite close to the ceiling of 17.2 bits.

***k*-anonymity:** As mentioned before, analyzing the data sets with format properties strings degenerate the data set into 1-anonymity since at least 22% of all persons can be uniquely identified. The largest set of persons which cannot be distinguished from each other (considering only the name and address columns) is expected to be about 76 (multiplying the largest sets from them both and dividing by total size).

### 4.3 Additional leak problems

The previous section assessed obvious PII records with variable format. But it may also apply to less obvious types. For example, if the date "3/10-2023" (in day/month-year format) is stored as a text string and this text string is anonymized using a format-preserving algorithm, then the output could become "0/64-3971". This anonymized version actually reveals that the day of month is 1-digit (e.g., 1–9) and the month number is 2-digit (e.g., 10–12). Hence, there are only $9 \cdot 3 = 27$ days of a year this anonymized date can represent (or $27/365.24 = 7.4\%$ of all potential dates).

In a situation where an attacker has gained access to a data set with anonymized data, the attacker can use the format properties model to reverse anonymization in several ways, for example described in the following sections.

This may also apply to other types. For example, a format-preserving anonymized postal code or phone number reveals information about which country they come from (number of digits, using space or parentheses in phone number, using number or letters in postal codes, etc.). A format-preserving anonymized IP address reveals information about the original IP address in the same way as a date in the previous example.

## 5 How to avoid the format information leak

In order to prevent the leak presented in this paper, the link between formats must be broken between input and output. However, we must keep in mind that there is a reason why format-preserving methods have been used: The protected data must have the same behavior as the original data, in particular, it should have the same data type as the original data.

The solution to the problem is to process data according to its data type instead of according to its format. If we observe an e-mail address as example, a valid e-mail address can have a variety of lengths and combinations of letters and numbers (at most positions) – as long as it contains a '@' and later at least one '.'. So, when anonymizing an e-mail address, the output should not preserve the length and positions of letters, numbers, and control characters, but only make sure that the overall rules for the type are respected.

This approach may be more demanding, partly since the type of data must be identified or known in advance and partly since the rules for each type must be implemented in the software performing the processing. But it is paramount for obtaining efficient anonymization / de-identification.

For a given data type, different formattings may be applied. For example, a Danish social security number if officially formatted “010180-1234” (i.e., a dash after 6$^{\text{th}}$ digit). But it is not uncommon to see social security numbers without dashes or even with other (odd) formatting, such as white spaces at various positions. For many applications, it would be best to normalize the formatting, i.e., always use dash after 6$^{\text{th}}$ digit and nothing else. But for some applications, for example generation of test data or data for AI training, it may be important to preserve format variations in order to ensure that the tested software or the trained model behaves correctly when they encounter odd formatting on real-world data. In that case, it is recommended to first run statistics on input data to measure format distribution and then apply these formats random to the output, but with the appropriate probabilities.

## 6 Conclusion

We have shared our discovery of a new methods for re-identifying persons in anonymized data sets. This method relies on the observation that certain anonymization methods, such as pseudonymization, format-preserving encryption, and redaction, in many configurations generate an output having the same length and format properties as the input.

The presented method does not relate to an error in the design or implementation of anonymization methods as such. In particular, for format-preserving encryption, it is not a cryptographic attack. Figure 2 illustrates that the format information bypasses the anonymization method. So, no matter how strong the encryption or randomization may be in altering or randomizing digits and letters, it does not affect the format information leakage as long as the format is preserved.

To assess the severity of the issue, we have analyzed 154 449 records of real personally identifiable information. The analysis showed that at least 22.0% or 33 916 records (probably significantly more) can be uniquely identified only based on the length and formatting of the person’s name and postal address. And the remaining records can be organized into small groups of candidates. The vast majority of the groups will contain only two or a few records. The largest group will have approximately 76 records. Consequently, format-preserving anonymization of this data set would be trivially reversible.

The issue can be mitigated by using anonymization methods designed to preserve the type of the data instead of preserving the format of the data. This has the downside that data type must be correctly identified before being processed and that methods for different data types have to be implemented. But it also has a significant upside – beyond mitigating the very severe leakage problem described in this paper – that data becomes more correct.

**Acknowledgments.** The author is grateful to Michael van Hauen, Tobias From, and Karina Staal Boesgaard for fruitful discussions and feedback. Thanks to the two financial organizations that supported this work by allowing us to analyze their data.

**Disclosure of Interests.** The author is Founder and CEO of the Danish company PII Guard, which develops and sells software and services specialized in data anonymization/de-identification. The author is inventor of Type-Preserving Encryption, which is the next generation of Format-Preserving Encryption and has been designed to mitigate the problems described in this paper.

## References

1. Pseudonymisation techniques and best practices, ENISA publication, nov. 2019 https://www.enisa.europa.eu/publications/pseudonymisation-techniques-and-best-practices, last accessed 2024/04/30
2. Verizon 2023 Data Breach Investigation Report, https://www.verizon.com/business/en-au/resources/reports/dbir/, last accessed 2024/04/30
3. A Mathematical Theory of Communication, C. Shannon, The Bell System Technical Journal (1948)
4. k-anonymity: A model for protecting privacy, L. Sweeney, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (2002)
5. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization, P. Ohm, UCLA Law Review, Vol. 57, p. 1701, 2010 (2009)