# Obfuscation Padding Schemes that Minimize Rényi Min-Entropy for Privacy

Sebastian Simon, Cezara Petrui, Carlos Pinzón, Catuscia Palamidessi

# Obfuscation padding schemes that minimize Rényi min-entropy for privacy

Sebastian Simon[⋆] Cezara Petrui[⋆] Carlos Pinzón[†]
Catuscia Palamidessi

Inria Saclay, France
Laboratoire d'Informatique de l'École Polytechnique (LIX)

[⋆] Authors contributed equally.
[†] Corresponding author: carlos.pinzon@lix.polytechnique.fr

Keywords: obfuscation, privacy , padding , Rényi min-entropy

**Abstract**

Consider a set of users, each of which is choosing and downloading one file out of a central pool of public files, and an attacker that observes the download size for each user to identify the choice of each user. This paper studies the problem of padding the files to obfuscate the exact file sizes and minimize the expected accuracy of the attacker, without exceeding some given padding constraints. We derive the algorithm that finds the optimal padding scheme, prove its correctness, and compare it with an existing solution that uses a similar but different attack model. We also discuss how the two solutions are related in terms of private information leakage.

## 1 Introduction

Consider a set of users, each of which is choosing and downloading one file out of a central pool of public files, and an attacker that observes the download size for each user and is willing to identify the choice of each user. The files are public, but the choices are private. The objective is to pad the files with some small overhead to obfuscate the information gained by the attacker and reduce his chances of discovering the choices of the users. This paper studies the problem of minimizing the expected accuracy of the attacker by padding the files without exceeding some given padding constraints.

On one extreme, if the files are not padded at all, the attacker might easily map the observed download sizes with the original files; e.g., if there is just one file of size 10.32MB and the attacker observes that the network traffic of some

user corresponds to a file of size 10.32MB, he will immediately know what file was chosen. This can be prevented by padding several files to common sizes to obfuscate the information gained by the attacker. On the other extreme, if all files are padded to a common size, this common size should be large enough to cover the largest file in the set, and, as a consequence, many small files will be padded excessively, increasing the bandwidth use. The ideal solution lies between these two extreme cases. For this reason, this paper considers the problem of maximizing privacy while respecting some flexible padding constraints, like, for example, that no file can increase its size more than 10%.

The attacker we consider makes just one attempt to re-identify the file, and to maximize his chances, he will of course guess a file that has the maximum posterior probability given the observed (obfuscated) size. This model of attack is known in literature as *one-try attack* [14], and it has been characterized in information-theoretic terms using *Rényi min-entropy*. More specifically, entropy in general represents the (lack of) information content of a discrete probability distribution, and Rényi min-entropy is a form of entropy that emphasizes the highest probability value. The prior and posterior entropies represent the probabilistic knowledge of the attacker before and after he observes the obfuscated size, respectively. In particular, Rényi posterior min-entropy is related to *hypothesis testing* and, as a measure, it closely corresponds to the *Bayes error*. The difference between the prior and posterior entropies represents how much the knowledge of the attacker (and hence his probability of success) increases thanks to the observation, and it is, therefore, a measure of the efficiency of the padding scheme. In literature this difference is known as Rényi min-entropy *leakage*.

The padding problem considered in this paper might also apply to equivalent scenarios in which an attacker exploits time side-channel information. For illustration, consider an intelligence service that is surveiling people entering and exiting a building. They can use the time each user took inside to infer the type of service he received, e.g., whether he was at the bank, shopping, or at the cinema in the mall. In this case, the users can waste some time inside the building on purpose to confuse the observer. Equivalently, a server can delay its responses in a planned manner to prevent an attacker from inferring the chosen type of request. More generally, an algorithm can sleep on purpose to prevent leaking information about the input, as exploited by timing attacks [13, 15].

## 1.1 Contributions

- We propose two algorithms that derive the optimal padding schemes, one for the deterministic case, and one for the randomized case (PRP and POP, defined in Section 2).
- We prove the correctness of the algorithms and test the implementations against brute-force solutions using small synthetic datasets.
- Likewise, we compare our algorithms with an existing solution [11] that uses an attack model based on Shannon entropy, and discuss how the two

approaches are related in terms of the type of private information leakage that each attacker represents.

- The code is publicly available at [10]. It includes not only the algorithms we propose, but also the reimplementation of the algorithms of [11] to support flexible padding constraints, multiple files having the same size, and sparse matrix representations.

## 1.2 Related Work

The model of attacker we use has been well investigated in the field of *Quantitative Information Flow* (QIF), which is a branch of security aimed at studying inference attacks, namely attackers that try to infer the value of the secret from related observations. The QIF theory actually formalizes a variety of models, each of them characterized by parameters that represent the capabilities and the goal of the attacker. For a detailed coverage of the topic we refer to [1].

This paper is strongly related with the work of Reed and Reiter [11], in which the authors consider the same problem with a different attack model, based on Shannon entropy, and more specifically, on measuring the leakage in terms of Shannon mutual information. Shannon mutual information is a well known notion that has been shown to be very useful in the several scientific fields. In security and privacy, however, it does not seem the right notion for modeling the attacker. Indeed, its operational interpretation corresponds to an attacker that can try to guess the exact secret by making an unbound number of attempts, and his objective is to minimize the expected number of attempts before he identifies it correctly. This seems a less natural model of attacker than those of QIF (and hence than the one we use, based on Rényi min-entropy), and it also sometimes leads to conclusions that are contrary to common sense. For a detailed discussion about this issue, refer to [14].

Reed and Reiter [11] propose three padding algorithms, called `PrpSh`, `PopSh` and PwoD (padding without a distribution), for finding padding schemes that minimize Shannon leakage under different bandwidth constraints. These algorithms do not support, however, multiple files having the same size nor flexible padding constraints as defined in this paper. We re-implemented their algorithms with these additional details before comparing them with our proposed solutions, and we explained in terms of attack models and information leakage the core difference between them.

In [4] they consider the BREACH/CRIME [7] security attack in which the attacker observes sizes and can also control a malicious script that runs in the browser of the victim. By exploiting the greedy mechanism of the Huffman encoder in the compression stage of the cookies, the attacker is able to use repeatedly the size information to discover the cookie secret and impersonate the victim. As they show, random gaussian padding can be used and is better than uniform padding to reduce the attacker's probability of success from 1.0 to 0.0026. Although this paper is more related with security than privacy, it shows how important padding can be to obfuscate information.

Lastly, one of the main conclusions in [16] is that the optimal way to reduce information obtained by an attacker that monitors traffic is to modify the traffic patterns so that they are confused with other patterns. We draw a similar conclusion formally in our problem (Proposition 2), proving that it is optimal to pad messages to reach the sizes other existing files.

## 2 Problem formalization

The collection of public files is denoted as $E = \{e_1, e_2, \ldots, e_n\}$, where $E$ is sorted non-decreasingly by the sizes $|e_i| \in \mathbb{N}$. For the sake of generality, we allow different files to have the same size, hence the set of file sizes $S \stackrel{\text{def}}{=} \{|e| \,|\, e \in E\}$ has $m \leq n$ unique elements, which we enumerate in increasing order as $S = \{s_1, s_2, \ldots, s_m\}$.

A *padding function* or padding scheme is a function $f : E \to \mathbb{N}$ respecting $f(e_i) \geq |e_i|$ that tells to what size each file should be padded. The padding constraints are expressed with the proposition $\forall i,\ f(e_i) \in [|e_i|, b_i]$, where each $[|e_i|, b_i] = \{|e_i|, |e_i| + 1, ..., b_i\}$ is an integer interval.

The sequence of users with their respective choices is modelled as a sequence of i.i.d. samples coming from the marginal distribution of the files. File $e_i$ is chosen with frequency $p_i \in [0, 1]$, where $\sum_{i=1}^{n} p_i = 1$. We let $X$ be a random variable satisfying $\mathbb{P}(X{=}e_i) \stackrel{\text{def}}{=} p_i$, thus, a sequence of users with choices can be represented as a sequence of i.i.d. choices following the distribution of $X$.

The attacker will predict, upon seeing a download of size $z \in \mathrm{Im}(f)$ (where the image $\mathrm{Im}(f) \stackrel{\text{def}}{=} \{z \in \mathbb{N} \mid \mathbb{P}(f(X){=}z) > 0\}$ denotes the set of possible outputs of $f$), that the secret value of $X$ is the file $e_i$ that maximizes $\mathbb{P}(f(e_i){=}z)$. To do this, he uses the public information he has access to and the information he can infer. The files and their sizes before padding are public, and he can determine the padding scheme by requesting each of the files himself, possibly multiple times in case of a randomized padding scheme. In addition, considering the worst-case scenario, we assume that he knows or has estimated the frequencies $p_i$ with which files are chosen on average. With this information, the attacker can always find a file $e_i$ that maximizes $\mathbb{P}(f(e_i){=}z)$ for the observed $z$, and his expected probability of success is therefore

$$\sum_{z \in \mathrm{Im}(f)} \max_{i \in [1..n]} \mathbb{P}(X{=}e_i \wedge f(X){=}z) = \sum_{z \in \mathrm{Im}(f)} \max_{i \in [1..n]} p_i \cdot \mathbb{P}(f(e_i){=}z). \tag{2.1}$$

The objective is to find a padding function $f : E \to \mathbb{N}$ that minimizes the accuracy of the attacker while respecting the given padding constraints. In addition, two scenarios are considered separately: *per-object-padding* (POP) refers to the case when $f$ is deterministic, hence the files are padded once and forever; *per-request-padding* (PRP) refers to the case when the padding is done on demand and $f$ is probabilistic.

## 2.1 Presentation in terms of privacy leakage

The objective of minimizing the attacker accuracy can equivalently be presented in terms of minimizing privacy leakage. There are several definitions for leakage $\mathbb{I}(|X|, f(X))$ of a padding function $f : E \to \mathbb{N}$. Particularly, Rényi min-entropy leakage [14], which we call *Rényi leakage* in this paper, is defined using Rényi min-entropy $\mathbb{H}_\infty$ as follows:

$$\mathbb{I}_\infty(f) \stackrel{\text{def}}{=} \mathbb{I}_\infty(|X|, f(X)) = \mathbb{H}_\infty(|X|) - \mathbb{H}_\infty(|X| \mid f(X)), \tag{2.2}$$

$$\mathbb{H}_\infty(|X|) = -\log_2 \max_{z \in \mathrm{Im}(f)} \mathbb{P}(|X| = z), \tag{2.3}$$

$$\mathbb{H}_\infty(|X| \mid f(X)) = -\log_2 \sum_{z \in \mathrm{Im}(f)} \max_{i \in [1..n]} (p_i \cdot \mathbb{P}(f(e_i) = z)). \tag{2.4}$$

The importance of Rényi leakage in more general contexts can be found in [9] and [14]. Basically, Rényi leakage is a special case ($\alpha = \infty$) of a family of leakages $\mathbb{I}_\alpha$ based on $\alpha$-Rényi entropy $\mathbb{H}_\alpha$. Since Rényi-min entropy $\mathbb{H}_\infty(|X|)$ is constant in regard to the padding-scheme, minimizing Equation (2.2) is equivalent to maximizing Equation (2.4), which is in turn equivalent to minimizing Equation (2.1). Therefore, Rényi leakage is in direct one-to-one correspondence with the probability of success of the attacker.

Another important case ($\alpha = 1$) is Shannon leakage, which is given by: $\mathbb{I}(|X|, f(X)) = \sum_{i,z} p_i \, \mathbb{P}(f(e_i){=}z) \log_2 \frac{\mathbb{P}(f(e_i)=z)}{\mathbb{P}(f(X)=z)}$. With some effort, this leakage can also be interpreted in terms of an attacker that we call Shannon attacker. The Shannon attacker is assumed to have access to an oracle that answers queries of the type "is the file in *this set of files*?" for each user, and his objective is to find the right files using the minimal number of queries, as in a 20Q game. Although the oracle assumption makes the Shannon attacker unrealistic, defenses against him are useful against the Rényi attacker of this paper because, intuitively, the more queries the Shannon attacker needs, the harder it is to guess the correct file in a single try.

For this particular application, the direct pragmatic connection between Rényi leakage and a simple adversary success makes it more appealing than the Shannon attacker. The same argument is used in [3], whose privacy measure is closely related with ours. More generally in the privacy community, leakage functions are better described in terms of their associated attacker rather than their information theoretic properties [2, 12].

## 2.2 Why not differential privacy?

Differential privacy [5], is one of the most prevalent formalizations of privacy. For this particular problem, a padding scheme $f$ satisfies $\epsilon$-differential privacy if and only if for all input files $e_i, e_j \in E$ and all output sizes $z \in \mathrm{Im}(f)$, we have $\mathbb{P}(f(e_1){=}z) \leq \exp(\epsilon)\, \mathbb{P}(f(e_2){=}z)$.

This notion of privacy represents an attacker whose success function is given by how much more likely one input file is *with respect to another one* for a

given observation. However, this is excessively strong for the problem under consideration. Indeed, as Theorem 1 shows, differential privacy can only be achieved at the total detriment of bandwidth use.

**Theorem 1.** *For any $\epsilon > 0$, the padding scheme that satisfies $\epsilon$-differential privacy and minimizes bandwidth is the one that pads all input files to the size of the largest one.*

*Proof.* Fix $\epsilon > 0$ and let $e_j \stackrel{\text{def}}{=} \arg\max_{e_i \in E} |e_i|$ be the largest file in $E$. For all sizes $z < |e_j|$, we have $\mathbb{P}(f(e_j){=}z) = 0$ because $e_j$ can not be padded to smaller sizes than $|e_j|$. Moreover, the differential privacy constraint forces every other file $e_i \neq e_j$ to satisfy $\mathbb{P}(f(e_i){=}z) \leq \exp(\epsilon)\mathbb{P}(f(e_j){=}z) = 0$ whenever $z < |e_j|$. In other words, all files must be padded to sizes at least as large as $|e_j|$, i.e. $\mathbb{P}(f(X) \geq |e_j|) = 1$. Among all the mappings $f$ that have this property, the one that minimizes bandwidth is the one that pads all files exactly to the largest file size $|e_j|$, and it satisfies $\epsilon$ differential privacy trivially because it is a constant function. □

Theorem 1 is the reason why we exclude differential privacy from the analysis and focus on the privacy notions discussed in the previous section. This theorem is a direct consequence of the inevitable fact that padding can only enlarge files and not reduce their sizes. Apart from putting in evidence the abusive overhead required by differential privacy, this theorem also shows that its parameter $\epsilon$ is irrelevant as a measure of privacy for the problem under consideration, making it inappropriate.

## 2.3 Simplification of the output set

We conclude this section by proving that optimal padding functions always map to sizes in $S$. This is a key-fact for the derivation of the algorithms and their proofs. Intuitively, if a set of files can be padded to a common certain size $z$, but can also be padded to $z-1$, we can pad them to $z-1$ and win some bandwidth without leaking any additional information. This forces the optimal padding functions to always pad to the sizes $z$ for which it is not possible to pad to $z-1$ without sacrificing privacy, which are precisely the sizes in $S$. The same holds true for padding schemes that minimize Shannon leakage, as shown in [11].

**Proposition 2.** For any padding-scheme $f : E \to \mathbb{N}$, there exists a padding-scheme $f^* : E \to S$ such that $\mathbb{I}(f^*) \leq \mathbb{I}(f)$. Moreover, $\mathbb{P}(f^*(X) \leq f(X)) = 1$, hence $f^*$ uses less padding (bandwidth) than $f$.

*Proof.* Define $f^*$ as the composition $f^* \stackrel{\text{def}}{=} g \circ f$, where $g(z) = \max\{s \in S : s \leq z\}$, that is, $f^*(X) = g(f(X))$. The function $g$ is defined only for $z \geq \min S$ and $f^*$ is well-defined because the padding constraints force $\mathbb{P}(f(X) \geq \min S) \leq \mathbb{P}(f(X) \geq |X|) = 1$. By definition, $g(z) \leq z$, thus $\mathbb{P}(f^*(X) \leq f(X)) = 1$. Let us now show, regarding privacy leakage, that $\mathbb{I}(f^*) \leq \mathbb{I}(f)$. Let $I^*_{xs}$ denote $\mathbb{P}(X{=}x \wedge f^*(X){=}s)$ and $I_{xz}$ denote $\mathbb{P}(X{=}x \wedge f(X){=}z)$. We will show that the

accuracy of the attacker (Eq. 2.1) is smaller or equal for $f^*$ than for $f$. This can be expressed as $\sum_s \max_x I^*_{xs} \leq \sum_s \sum_{z:g(z)=s} \max_x I_{xz}$. On the left and right-hand sides, we have summations on $s \in S$, so it suffices to prove that this inequality holds for each fixed $s$. At each $s \in S$, since $I^*_{xs} = \sum_{z:g(z)=s} I_{xz}$, the inequality becomes $\max_x \sum_{z:g(z)=s} I_{xz} \leq \sum_{z:g(z)=s} \max_x I_{xz}$, which is necessarily true. Indeed, letting $x^{(s)} \stackrel{\text{def}}{=} \arg\max_x \sum_{z:g(z)=s} I_{xz}$ for the left-hand side, we have for each $z$ with $g(z) = s$ that $I_{x^{(s)}z} \leq \max_x I_{xz}$. □ □

Proposition 2 can be seen as an instance of the Data Processing Inequality, which can be found as Theorem 8 of [6], or more generally for privacy contexts in [8].

**Corollary 3.** *A padding function that has minimal leakage must pad each file to the size of another file in the initial set.*

Having Corollary 3 in mind, the padding scheme $f$ can be represented as an obfuscation channel matrix $P$ where $p_{ij} = \mathbb{P}(f(e_i){=}s_j)$, in which case, the problem can be specified as shown below, and the attacker accuracy becomes

$$\sum_j \max_{i \in [1..n]} p_i \cdot p_{ij}. \tag{2.5}$$

**Problem input**: (1) A set $E$ of $n$ files $\{e_i \,|\, i \in [1..n]\}$ with frequencies $p_i$, sorted sizes $|e_i|$ and set of unique sizes $S = \{s_1, ..., s_m\}$. (2) Padding constraints of the form $\forall i,\ s_{l_i} \leq f(e_i) \leq s_{r_i}$, parametrized with pairs of indices $l_i, r_i \in [1..m]$.

**Desired output**: A padding function $f : E \to S$ in the form of a channel matrix $p_{ij} = \mathbb{P}(f(e_i){=}s_j)$ that minimizes Rényi leakage $\mathbb{I}_\infty(f)$ or equivalently Eq. (2.5). Depending on the problem variant, $f$ must be deterministic (POP) or randomized (PRP).

# 3 Algorithms

In this section, we derive the algorithms `PopRe` and `PrpRe` that minimize the Rényi leakage (2.2) for the POP and PRP cases respectively. They contrast those for Shannon mutual information minimization found in the paper [11], denoted here as `PopSh` and `PrpSh`. The complexities of these algorithms are summarized in Table 1.

Algorithm `PrpSh` is an approximation algorithm and has a runtime complexity that depends on the degree of accuracy imposed by the user and the limit number of iterations ITERS allowed. Also, the complexities of the dynamic programming algorithms correspond to the theoretical worst-case and might overestimate the actual implementations. For instance, although `PopRe` has two parameters varying in $[1..n]$, not all combinations need to be calculated in a top-down implementation.

| Algorithm | Minimizes | WC Runtime complexity | Memory |
|---|---|---|---|
| `PopRe` | Rényi leakage | $O(n^2\,\bar{b})$ | $n\,\bar{b}$ |
| `PrpRe`, `PrpReBa` | Rényi leakage | $O(n\,\bar{b})$ | $n\,\bar{b}$ |
| `PopSh` | Shannon leakage | $O(n\,\bar{b})$ | $n\,\bar{b}$ |
| `PrpSh` | Shannon leakage | $O(\text{ITERS} \cdot n\,m)$ | $n\,m$ |

Table 1: Complexities, where $\bar{b} \stackrel{\text{def}}{\coloneqq} (1/n)\sum_{i=1}^{n} r_i - l_i + 1$ is the matrix average band size. For practical reference, with reasonable padding constraints, if the files are diverse with a large and spread spectrum of sizes, one expects $\bar{b} \ll m \approx n$.

## 3.1 Per-object-padding scenario, `PopRe`

In this section we develop the algorithm that minimizes Rényi leakage in the POP variation, in which the matrix $P$ is constrained to $p_{ij} \in \{0, 1\}$. Before describing the algorithm, we will prove Remark 4, which will be used as the main update of the entries of the channel-matrix.

**Remark 4.** Let $f$ be a Rényi optimal padding-scheme and $e_i$ be the file with the highest associated frequency $p_i$, and assume that $p_{ij} = 1$ for some $j \in [1..m]$. Then there exists a padding-scheme $f^*$ with the same Rényi leakage such that $p_{kj} = 1$ for all $k \in [1..n]$ such that $j \in [l_k..r_k]$.

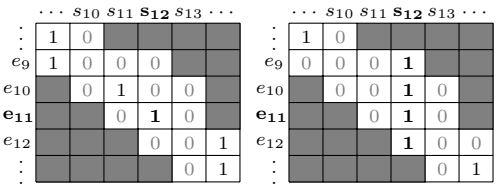


Figure 3.1: Remark 4: if the file with maximal frequency is $e_{11}$ and the left matrix ($f$) is optimal, the right one ($f^*$) must be as well.

*Proof.* We consider the padding-scheme $f$ to be represented as the channel-matrix between the secrets and the observables. When we want to minimize (2.5) we sum over each column of the matrix $P$. In particular, on the column $j$ we have $\max_{a\in[1..n]}(p_a \cdot p_{aj}) = p_i$ since $p_i$ is the highest frequency among the frequencies of the files and $p_{ij} = 1$. Now, let us consider the padding-scheme $f^*$ whose matrix $P^*$, consists on moving every 1 that we can to column $j$:

$$p^*_{ab} = \begin{cases} p_{ab} & \text{if } b \neq j \text{ and } a \in [1..n] \text{ such that } j \notin [l_a..r_a] \\ 1 & \text{if } b = j \text{ and } a \in [1..n] \text{ such that } j \in [l_a..r_a] \\ 0 & \text{otherwise} \end{cases}$$

On the column $j$ of the matrix $P^*$ we will still have $\max_{a\in[1..n]}(p_a \cdot p^*_{aj}) = p_i$ because the padding-scheme $f^*$ preserves the maximum on column $j$. Moreover, on the rest of the columns, the maximum either decreases or stays the same since we created more entries $p^*_{ab} = 0$, which means that the product $p_a \cdot p^*_{ab} = 0$. However, we chose $f$ to be the Rényi optimal padding-scheme and with the remarks above, $f$ and $f^*$ give the same leakage. □ □

**Algorithm 1** Per-object-padding pseudocode. This implementation uses recursion both for computation and reconstruction.

**procedure** RENYI POP ▷ Main function
  MEMO $\leftarrow \{\}$ ▷ Empty map
  $p_{ij} \leftarrow 0$ ▷ A matrix $p$ full of zeros
  renyi $\leftarrow$ RECONSTRUCT$(0, n)$
  **return** $(p, \text{renyi})$ ▷ Output matrix $p$ and its renyi leakage
**end procedure**
**procedure** RECONSTRUCT$(a, b)$
  $(\text{renyi}, k, a^\star, b^\star) \leftarrow f(a, b)$
  **for** $j = a^\star..b^\star$ **do** $p_{jk} \leftarrow 1$ **end for**
  **if** $a < a^\star$ **then** RECONSTRUCT$(a, a^\star)$ **end if**
  **if** $b^\star < b$ **then** RECONSTRUCT$(b^\star, b)$ **end if**
  **return** renyi
**end procedure**
**procedure** $f(a, b)$
  **if** $(a, b) \in$ MEMO **then return** MEMO$[(a, b)]$ **end if**
  **if** $a = b$ **then return** $(0, \infty, a, b)$ **end if**
  best $\leftarrow (\infty, \infty, \infty, \infty)$
  $i_{\max} \leftarrow \arg\max_{i=a..b} p_i$
  **for** $k = l_{i_{\max}}..r_{i_{\max}}$ **do**
    $j_{\min}, j_{\max} \leftarrow$ range of files $e_{j_{\min}}..e_{j_{\max}}$ that can be padded to size $s_k$
    $a^\star \leftarrow \max(a, j_{\min})$
    $b^\star \leftarrow \min(b, j_{\max})$
    renyi $\leftarrow f(a, a^\star)[0] + p_{i_{\max}} + f(b^\star, b)[0]$ ▷ Index [0] is the renyi component
    this $\leftarrow (\text{renyi}, k, a^\star, b^\star)$
    best $\leftarrow \min(\text{best}, \text{this})$ ▷ Lexicographic (compares first by renyi)
  **end for**
  $(\text{renyi}, k, a^\star, b^\star) \leftarrow$ best ▷ Unpack tuple
  MEMO$[(a, b)] \leftarrow (\text{renyi}, k, a^\star, b^\star)$
  **return** $(\text{renyi}, k, a^\star, b^\star)$
**end procedure**

Figure 3.1 depicts an example of a sub-matrix of $P$ as described in Remark 4. In the figure, we have exactly one entry equal to 1 in each line because the channel-matrix is stochastic, and we are in the POP case. Additionally, the quantity in (2.5) represents the sum of the maximum over columns where each 1 counts for the frequency of the file. Then, the update does not increase the (2.5) because the 1 with maximal frequency dominates its column, and moving all possible 1's above or below it does not increase Rényi leakage.

Using Remark 4 we can divide the padding problem into sub-problems that minimize (2.5) and leverage dynamic programming: $\forall a \leq b \in [1..n]$, we define

$$D[a][b] = \min_{\text{P channel matrix}} \sum_{j \in [1..m]} \max_{i \in [a+1..b]} (p_i \cdot p_{ij}),$$

i.e. $D[a][b]$ gives the minimal leakage for the sub-problem that pads files from $e_{a+1}$ to $e_b$, under the general constraints.

By convention, we consider $D[i][i] = 0$, which will be the base case. To write the recurrence formula, we need to take the file $e_{i_{\max}}$ with maximum frequency $p_{i_{\max}}, i_{\max} \in [a+1, b]$. We go through every size index $k \in [1..m]$ such that $e_{i_{\max}}$ can be padded to the size of $s_k$, and we update the channel-matrix according to Remark 4, i.e. add 1's on $k$-th column if we can (taking into consideration the padding constraints) and complete the lines that have a fixed 1 with 0's on the remaining entries. Then, we apply the recurrence on the rows which are not updated, i.e. from $a$ to $a^* \stackrel{\text{def}}{:=} \max(a, \max_{i \in [1..n]}\{i | r_i < k\})$, and, respectively, from $b^* \stackrel{\text{def}}{:=} \min(k, b)$ to $b$. Hence,

$$D[a][b] = p_{i_{\max}} + \min_{k \in [l_{i_{\max}}..r_{i_{\max}}]} (D[a][a^*] + D[b^*][b])$$

After applying the dynamic algorithm program with the aforementioned recurrence, we get the minimization of (2.5) in $D[0][n]$, from which we can compute the minimal Rényi leakage. If we want to recover the channel-matrix itself, in $D[a][b]$ we pass on the index $k$ for which the maximum happens, as an argument. In case of a tie, we choose the smallest index $k \in \{1, \ldots, n\}$ in order to reduce average padding. Hence, we know in each sub-interval $[a, b]$ what we pad everything to, so the information is enough to recover the channel matrix. A pseudocode summarizing all the logic is shown in Algorithm 1. A concrete optimized implementation can be found in [10].

In Figure 3.2 we depict the channel-matrix of the files with sizes $S = \{1000, 1050, 1100, 1110, 1120, 1140\}$ and associated frequencies $\{22\%, 5\%, 23\%, 12\%, 18\%, 20\%\}$. As shown in the visual representation of the padding-scheme in the right, we observe that, for both of the existing padded sizes, there are multiple files that are padded to the same element, making them indistinguishable for an attacker. Moreover, the blue bars on the graph

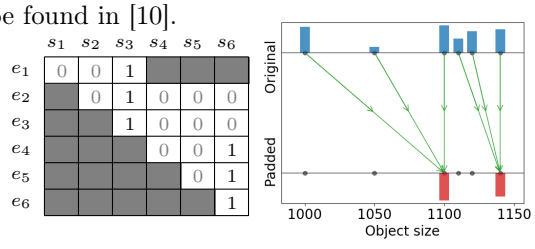


Figure 3.2: PopRe on a dataset of 6 files.

indicate the frequencies of the files, and the red bars, the maximum frequency among the frequencies of the files padded to each specific size. The red bars are effectively highlighting the terms of the sum (2.5).

### 3.2 Per-request-padding scenario, `PrpRe`

In this section, we treat the case of Per-Request-Padding and provide an algorithm for finding the probabilistic channel-matrix $P$ which minimizes the Rényi leakage. We will look at the joint distribution matrix $I$ with entries $I_{ij} = p_i \cdot p_{ij}, \forall i \leq n, j \leq m$, for which $\sum_{j=1}^{m} I_{ij} = p_1$ for each $i \in [1..n]$.

We proceed by finding iteratively, for each of the $m$ columns, starting from the last one, the Rényi optimal manner of setting the entries of $I$ given the padding constraints. Furthermore, we define the *optimal distribution of $p_i$ across the $i$-th row,* $1 \leq i \leq n$ to be the way we fill in the entries $p_{i1}, \ldots, p_{im}$ such as to obtain the minimum sum of the type (2.5) and preserve the relation $p_{i1} + \ldots + p_{im} = p_i$.

The proof of our algorithm requires us to consider sub-problems in which the sequence $(p_i)_{1 \leq i \leq n}$ is updated at each step of the algorithm, thus being different from the initial set of frequencies associated to each file. Hence, we rewrite the problem as a more general one in terms of a *budget* sequence $(b_i)_{1 \leq i \leq n}$ of length $n$ (initialized as $(p_i)_{1 \leq i \leq n}$), which dictates the remaining value to be distributed across each row $i$, for $i \in [1..n]$. The general problem is "Given a non-negative budget sequence $(b_i)_{i=1}^{k}$ of length $k \in [1..n]$, find a solution matrix $I_{k \times m}$ that minimizes Equation (2.5), under the padding constraints for rows $i \in [1..k]$, namely the set $\{[l_1, r_1], \ldots, [l_k, r_k]\}$ and $\sum_{j=1}^{m} I_{ij} = b_i$".

We will design the algorithm to solve the general problem recursively by returning the matrix $I$ for the budget sequence $\{p_1, \ldots, p_n\}$ with $n$ terms. The recurrence relationship can be described using the following observation that is used when creating the probabilistic channel-matrix for the padding-scheme $f$:

**Remark 5.** The solution $I_{k \times m}$ for a given $(b_i)_{i=1}^{k}$ that minimizes Rényi leakage satisfies the recurrence relationship

$$I_{ij} = \begin{cases} b_i & \text{if } j = m \text{ and } i \in [1..k], |e_i| = s_m \\ b_i - b_i^{'} & \text{if } j = m \text{ and } i \in [1..k-1], |e_i| \not= s_m, \\ & m \in [l_i..r_i] \\ I_{ij}^{'} & \text{otherwise} \end{cases}$$

where $I_{(k-t) \times (m-1)}^{'}$ is the solution to the same minimization problem for the sequence $(b_i^{'})_{i=1}^{k-t}$ of length $k - t$, $t$ = number of files from $E$ which can be padded to $s_m$, such that for any $i \in [1..k-t]$, it is defined as:

$$b_i^{'} = \begin{cases} \max(b_i - b_{t_{\max}}, 0) & \text{if } m \in [l_i..r_i] \text{ and} \\ & b_{t_{\max}} = \max\{b_i || e_i| = s_m\} \\ b_i & \text{otherwise} \end{cases}$$

*Proof.* If there are no files among $\{e_1, \ldots, e_k\}$ which can be padded to $s_m$, we set $t = 0$ and solve the minimization problem for the same budget sequence and for the set of $m-1$ sizes $\{s_1, \ldots, s_{m-1}\}$.

If there are files that can be padded to $s_m$, then due to the padding constraints, the element $e_i$ can only be padded to $s_m$, so the entry $I_{im}$ must necessarily be equal to $b_i$, for all $i$ such that $|e_i| = s_m$. Let us denote by $T = \{k-t+1, \ldots, k\}$ the set of indices satisfying $|e_i| = s_m, \forall i \in T$ and $b_{t_{\max}} = \max\{b_i | i \in T\}$. Clearly, for every $i \in T$, $I_{ij} = 0, \forall j \in \{1, \ldots, k-1\}$. On the $m$-th column of the matrix $I$, we have $\max_{i \in [1..k]} I_{im} \geq b_{t_{\max}}$.

In order to minimize the sum (2.5) and taking into consideration that the maximum entry on column $m$ is at least $b_{t_{\max}}$, we aim to distribute for every $i$ such that $e_i$ can be padded to $s_m$ and $|e_i| \neq s_m$, a quantity equal to $b_{t_{\max}}$ (or, if $b_i < b_{t_{\max}}$, then we distribute the whole $b_i$) on the entry $I_{im}$, so that we preserve the maximum on this last column to be $b_{t_{\max}}$. This way, we can assure that, among the other columns, we'll have to distribute a smaller fraction of $b_i$, which means that the maximum on each column between 1 and $m-1$ will decrease, and so will (2.5).

The problem reduces to find the optimal sub-matrix $I^{'}_{(k-t)\times(m-1)}$ to complete the first $k-t$ rows of $I$, and with the aforementioned remark, we can actually consider $I^{'}$ to be the solution given the updated sequence $(b^{'}_i)_{1 \leq i \leq k-t}$ which is defined, for every $i$ such that file $e_i$ that can be padded to $s_m$, as either 0, if $b_i \leq b_{t_{\max}}$, or as $b_i - b_{t_{\max}}$, if $b_i \geq b_{t_{\max}}$. When we reconstruct the matrix $I$, on the $m$-th column we will have the value $I^{'}_{im} + b_{t_{\max}}$ or $I^{'}_{im} + b_i$ (depending on whether $b_i$ is smaller, respectively larger, than $b_{t_{\max}}$).

Now, let us show that, for the sub-matrix $I^{'}$, we have 0's on every entry of the $m$-th column. By definition, $I^{'}$ must be a Rényi optimal solution for the updated sequence of $b^{'}_i$'s. Using Proposition 2, there exists a Rényi optimal padding-scheme $f^{'}$ which maps $e_i, i \in [1..k-t] \to \{s_1, \ldots, s_{k-t}\}$, for any set of files $\{e_1, \ldots, e_{k-t}\}$ with the associated frequencies $\{b^{'}_1, \ldots, b^{'}_{k-t}\}$. Consequently, for every $i \in [1..k-t], \mathbb{P}(f^{'}(e_i) = s_m) = 0 \Rightarrow I^{'}_{im} = 0$. □ □

Therefore, we have proved that the matrix $I$ can be recursively expressed using the sub-matrices obtained when we update the budget sequence accordingly, at each step decreasing by 1 the number of columns and by at least 1 the number of rows of the matrix returned from the algorithm, until we reduce a problem to finding the Rényi optimal scheme for a budget sequence with a single element. Since we want to minimize (2.5) in the case of $n$ files with frequencies $\{p_1, \ldots, p_n\}$ and the associated set of sizes $\{s_1, \ldots, s_m\}$, we proceed the induction on the number of rows and columns as described in Remark 5 and eventually fill in all the entries of the solution $I_{n \times m}$. The channel-matrix $P$ is then computed as $p_{ij} = I_{ij}/p_i$, and this is the output of `PrpRe`.

This algorithm is presented in Algorithm 2 in the form of pseudocode, and it is implemented in [10] with some optimizations.

**Algorithm 2** Per-request-padding pseudocode.

**procedure** RENYI PRP
  $\forall i, b_i \leftarrow p_i$ ▷ budget array
  $I \leftarrow$ Joint prob. matrix of zeros
  **for** j=m, m-1, ..., 1 **do**
    $t_{\max} = \arg\max_{\{i \,||e_i|=s_j\}} b_i$
    **if** $b_{t_{\max}} > 0$ **then**
      $j_{\min}, j_{\max} \leftarrow$ range of files $e_{j_{\min}}..e_{j_{\max}}$ that can be padded to size $s_j$
      **for** $i = j_{\max}, j_{\max} - 1, ..., j_{\min}$ **do**
        $I[i, j] \leftarrow \min(b_{t_{\max}}, b_i)$
        $b_i = b_i - I[i, j]$
      **end for**
    **end if**
  **end for**
  $P \leftarrow$ channel matrix after dividing each row $i$ of $I$ by $p_i$
  **return** $P$
**end procedure**

### 3.2.1 Bandwidth minimization

Once `PrpRe` has found a channel matrix that minimizes Rényi leakage, it is still possible to use heuristics to search for other channel matrices with the same (minimal) leakage but with less bandwidth use. We call `PrpReBa` to be the algorithm that runs `PrpRe` and the bandwidth reduction heuristics afterwards.

Let the list $C$ of maximums on each column after running `PrpRe`, i.e. $C = \{\max_{i\in[1..n]} I_{ij} | j \in [1..m]\}$, where $C_j = \max_{i\in[1..n]} I_{ij}$ for every $j \in [1..m]$. Define a *move* to be a change in the matrix $I$ performed on two of the entries of the matrix at line $i$, for some $i \in [1..n]$ such that $(I_{ia}, I_{ib})$ becomes $(I_{ia} - \alpha, I_{ib} + \alpha)$ while keeping the entries of $I$ positive, i.e. $\alpha \leq I_{ia}$.

Now, we will describe an *update* on the line $i$, which will consist of a series of *moves* and will return a new matrix $I^*$. We start with $I^*$ to be the matrix $I$, but with 0's on the $i$-th line. Since the sum on row $i$ is equal to $p_i$, we start with this quantity and go through the columns in order from $j = 1$ to $j = m$. For each column, we set:

$$I_{ij} = \begin{cases} C_j & \text{if } C_j + \sum_{k=1}^{j-1} I_{ik} \leq p_i \\ p_i - \sum_{k=1}^{j-1} I_{ik} & \text{otherwise} \end{cases}$$

# 4 Experiments and Comparison

Several experiments were carried out for three distinct purposes, namely, (1) to test the correctness of the implementations against brute-force algorithms for small sized problems, (2) to corroborate the direct link between Rényi leakage

and the success rate of an attacker and (3) to compare the runtime, bandwidth and leakages of all the algorithms on a public dataset. The code of all the experiments is available in [10].

## 4.1 Brute-force tests for correctness

To complement and corroborate the theory developed in this paper, all the algorithms were tested against brute-force implementations for small datasets (with at most 10 elements). More precisely, for each randomly generated test case of file sizes and frequencies, we explored (exhaustively) all the POP padding schemes satisfying the constraints, and chose among them, the ones that minimized Rényi leakage, Shannon leakage or bandwidth, with the purpose of comparing them with the solutions returned by our algorithms.

We ran ten thousand experiments (code available in [10]), all corroborating that: among all POP schemes, `PopRe` achieves minimal Rényi leakage, `PopSh` achieves minimal Shannon leakage, and `PrpRe` leaks at most the Rényi leakage of `PopRe`.

## 4.2 Attacker test for illustration

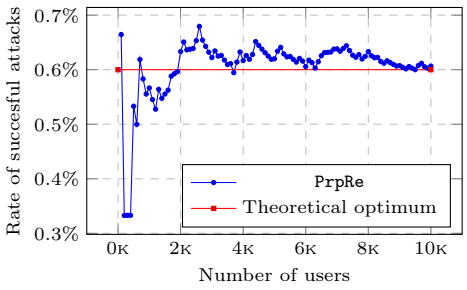


Figure 4.1: Attacker's success convergence.

We simulated the attacker described in this paper by Equation (2.1), who always guesses the original file with maximum probability given the priors and the padding scheme. Figure 4.1 shows that as the number of user increases, the success rate of the attacker against the padding proposed by `PrpRe` approaches the expected theoretical minimal possible success rate. This is a direct consequence of the law of large numbers as well as the equivalence between minimizing the expected success of the attacker (2.1) and the Rényi leakage, via Eq. (2.5).

## 4.3 Dataset tests for comparison

We used the dataset of NodeJS, proposed originally in [11]. This dataset consists of a list of 423,450 javascript packages provided by NPM for browser and nodeJS applications, each with its associated file size and access frequency, as of August 2021. Taking into account the large number of files and the availability of the access frequencies, we used the NodeJS dataset to benchmark the algorithms.

We used two versions of the NodeJS dataset: the *large* NodeJS dataset is the original dataset with 423,450 files, and the *small* consists of only the 1000 most frequently accessed files. The small NodeJS dataset allowed us to benchmark and compare the algorithms with large complexity, which timed-out on the

large dataset. In all experiments, we parametrize the padding constraints with a single constant $c > 0$ that represents the constraint $|X| \leq f(X) \leq (1+c) \cdot |X|$.

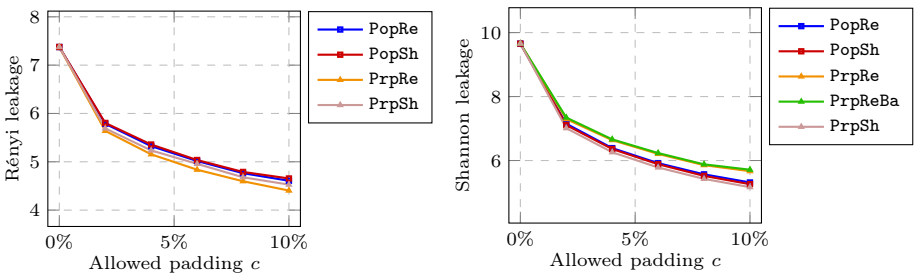


Figure 4.2: Rényi and Shannon leakage on the small dataset.

Figure 4.2 depicts the variation of privacy leakage as a function of $c$ on the small dataset. The trend is approximately equal in the large dataset, except that `PopRe` times out. The Rényi plot does not include `PrpReBa` to reduce redundancy, as it coincides with `PrpRe`. In the figure, we can appreciate the expected trend that larger $c$ allows for more padding and less leakage of privacy, both in Rényi and Shannon definitions. It can also be verified that the algorithms tuned to minimize Rényi leakage, have a very small (but not minimal) Shannon leakage, and vice-versa. For instance, the differences between `PopRe` and `PopSh` in both leakages are inferior to 2%. This is a consequence of the information theoretical connection between the two types of leakage.

The bandwidth increase generated by the padding of the files can be analyzed in Figure 4.3. For reference, the average file size in the dataset, weighted by frequency is 52.5 KB, so 1% increase, means around 5.3 additional kilobytes. Several observations can be made out of Figure 4.3. First, as anticipated, the larger the $c$, the larger the paddings on average. Second, the algorithms do not pad as much as they are allowed. Instead, when 10% is allowed, the optimal paddings lie at around 2% for the small dataset and 4% for the large dataset. For this particular example, the algorithms used more of the available padding on the large than in the small dataset, but we did not explore in depth in our experiments whether this pattern holds in general. Third, the improvements of `PrpReBa` over `PrpRe` can be corroborated, and estimated to approximately 20% less bandwidth use with the same Rényi leakage. Lastly, it appears empirically that the solutions that minimize Rényi leakage use less padding on average than those that minimize Shannon leakage.

Furthermore, the box plots in Figure 4.3 show that the padding use (with respect to the average file size) is most often below its average, meaning that there are a few files that contribute significantly more than the others to bandwidth excess. These files must be the largest, as they are the files for which the additional bandwidth can be the largest compare, even possibly exceeding the average file size. Note that the computation of confidence intervals can not be made for privacy leakages (Figure 4.2), as they are global guarantees of privacy that do not make sense for individual files.

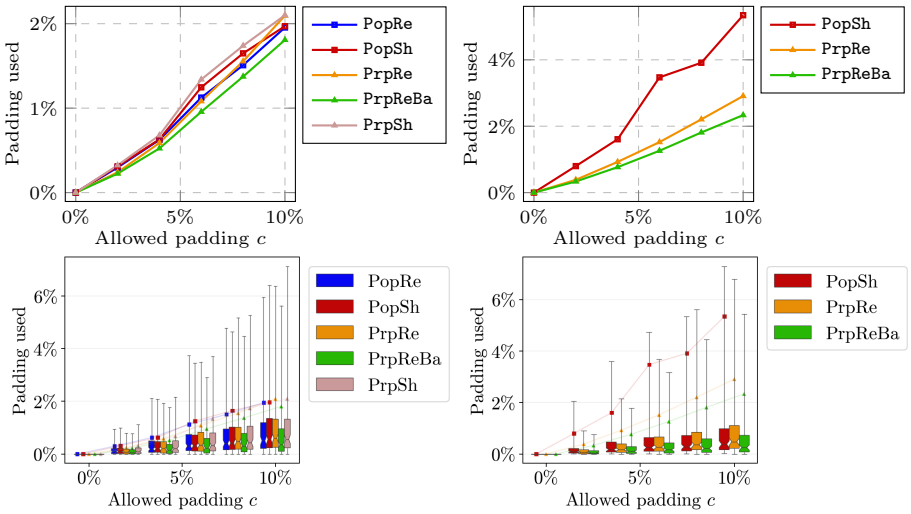


Figure 4.3: Bandwidth increase on the small (left) and large (right) datasets. The top plots show expected values and the bottom plots show, in addition, confidence intervals for a single random request. For each box, the body (Q1 and Q3 quartiles) corresponds to 50% confidence and the whisker (5% and 95% quantiles) to 90%.

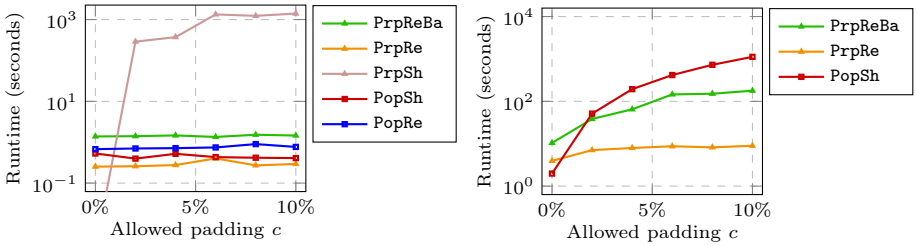


Figure 4.4: Runtime plots on small (left) and large (right) datasets. The plots ignore the 7 additional seconds needed for JIT compilation.

Figure 4.4 depicts the runtime of the algorithms under analysis. We refer the reader to Table 1 about the runtime complexities for a richer analysis of the plots. The analysis could have been even richer, if we included confidence intervals as in Figure 4.3, but we did not do it because of time constraints (the execution of `PopSh` in the large dataset is in the order of several hours of CPU time), and because the added value in this case is very little, as we already have a theoretical derivation of the complexities.

The left plot in Figure 4.4 does not have a clear tendency of longer executions for more relaxed padding constraints (higher $c$, thus also higher $\bar{b}$), meaning that for small datasets, all algorithms are suitable. In this regime, the runtime is not yet affected significantly by the growth of $\bar{b}$, possibly due to large constants that

are masked by the complexity class and implementation details, especially for `PrpReBa`. Nevertheless, the difference between `PopRe` versus `PrpRe` and `PopSh` is already visible, and indeed, `PopRe` times out (several hours) for the large dataset. The right plot highlights the scalability of the algorithms. For all values of $c$ plotted in this graph, the runtime for `PrpRe` is under 7 seconds, which makes it the fastest algorithm. `PrpReBa` peaks at $c = 10\%$ with around 3 minutes while `PopSh` needed 15 minutes. In this regime, the effect of increasing $\bar{b}$ via $c$ on the runtime is clear.

# 5 Conclusion

We designed and proved the optimality of several algorithms (`PopRe`, `PrpRe`, `PrpReBa`) that minimize the expected success rate of an attacker. The algorithms were compared with existing solutions (`PopSh`, `PrpSh`) that consider a different attack model. The comparison was done both numerically via experiments and theoretically via privacy leakage.

Prioritizing scalability, we recommend using either `PrpRe` or `PrpReBa` for the PRP problem, as they are much faster and provide protection against a more reasonable attacker than the existing solutions (`PopSh`, `PrpSh`). Nevertheless, for the POP problem, we recommend any of either the existing solution `PopSh` or our algorithm `PopRe` that minimizes Rényi leakage, because even though our attack model is more realistic, the complexity of `PopSh` makes it more practical.

In general terms, the two attack models are correlated in the sense that the optimizing against one of them results in a strong, though not optimal, protection against the other one (with empirical differences of less than 2%). In more detail, however, the Rényi attacker is more realistic than the Shannon attacker, and the padding schemes that minimize Rényi leakage seem to use less bandwidth in practice, making our proposed algorithms even more appealing.


## Acknowledgements

This work was supported by the European Research Council (ERC) project HYPATIA under the European Union's Horizon 2020 research and innovation programme. Grant agreement n. 835294.


# References


[1] Alvim, M.S., 0001, K.C., McIver, A., Morgan, C., Palamidessi, C., 0001, G.S.: The Science of Quantitative Information Flow. Information Security and Cryptography, Springer (2020)

[2] Alvim, M.S., Chatzikokolakis, K., Palamidessi, C., Smith, G.: Measuring information leakage using generalized gain functions. In: Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF). pp. 265–279 (2012).

https://doi.org/http://doi.ieeecomputersociety.org/10.1109/CSF.2012.26, http://hal.inria.fr/hal-00734044/en

[3] Cherubin, G.: Bayes, not naïve: Security bounds on website fingerprinting defenses. Proceedings on Privacy Enhancing Technologies **2017**(4), 215–231 (oct 2017). https://doi.org/10.1515/popets-2017-0046, https://doi.org/10.1515%2Fpopets-2017-0046

[4] Degabriele, J.P.: Hiding the lengths of encrypted messages via gaussian padding. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1549–1565 (2021)

[5] Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming. pp. 1–12. Springer (2006)

[6] Espinoza, B., Smith, G.: Min-entropy leakage of channels in cascade. In: International Workshop on Formal Aspects in Security and Trust. pp. 70–84. Springer (2011)

[7] Gluck, Y., Harris, N., Prado, A.: Breach: reviving the crime attack (2013). Dostupné také z: http://css. csail. mit. edu/6 **858** (2015)

[8] M'rio, S.A., Chatzikokolakis, K., Palamidessi, C., Smith, G.: Measuring information leakage using generalized gain functions. In: 2012 IEEE 25th Computer Security Foundations Symposium. pp. 265–279. IEEE (2012)

[9] Palamidessi, C., Romanelli, M.: Feature selection with rényi min-entropy. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition. pp. 226–239. Springer (2018)

[10] Pinzón, C., Petrui, C., Simon, S.: min-leakage-padding. https://github.com/caph1993/min-leakage-padding (2022), accessed: August 2022

[11] Reed, A.C., Reiter, M.K.: Optimally hiding object sizes with constrained padding (2021). https://doi.org/10.48550/ARXIV.2108.01753, https://arxiv.org/abs/2108.01753

[12] Romanelli, M.: Machine learning methods for privacy protection: leakage measurement and mechanisms design. Ph.D. thesis, Institut Polytechnique de Paris; Università degli studi (Sienne, Italie) (2020)

[13] Schindler, W.: A timing attack against rsa with the chinese remainder theorem. In: Cryptographic Hardware and Embedded Systems—CHES 2000: Second International Workshop Worcester, MA, USA, August 17–18, 2000 Proceedings 2. pp. 109–124. Springer (2000)

[14] Smith, G.: On the foundations of quantitative information flow. In: de Alfaro, L. (ed.) Proceedings of the 12th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2009). LNCS, vol. 5504, pp. 288–302. Springer, York, UK (2009)

[15] Song, D.: Timing analysis of keystrokes and ssh timing attacks. In: Proc. of 10th USENIX Security Symposium, 2001 (2001)

[16] Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: An efficient defense against statistical traffic analysis. In: NDSS. vol. 9. Citeseer (2009)