On Fingerprinting Attacks and Length-Hiding Encryption

Kai Gellert, Tibor Jager, Lin Lyu, and Tom Neuschulten

Bergische Universität Wuppertal, Germany
{kai.gellert,tibor.jager,lin.lyu,tom.neuschulten}@uni-wuppertal.de

Abstract. It is well known that already the *length* of encrypted messages may reveal sensitive information about encrypted data. *Fingerprinting attacks* enable an adversary to determine web pages visited by a user and even the language and phrases spoken in voice-over-IP conversations.
Prior research has established the general perspective that a length-hiding padding which is long enough to improve security significantly incurs an unfeasibly large bandwidth overhead. We argue that this perspective is a consequence of the choice of the security models considered in prior works, which are based on classical indistinguishability of *two* messages, and that this does not reflect the attacker model of typical fingerprinting attacks well.
Therefore we propose a new perspective on length hiding encryption, which aims to capture security against fingerprinting attacks more accurately. This makes it possible to concretely quantify the security provided by length-hiding padding against fingerprinting attacks, depending on the real message distribution of an application. We find that for many real-world applications (such as webservers with static content, DNS requests, Google search terms, or Wikipedia page visits) and their *specific* message distributions, even length-hiding padding with relatively small bandwidth overhead of only 2–5% can already significantly improve security against fingerprinting attacks. This gives rise to a new perspective on length-hiding encryption, which helps understanding how and under what conditions length-hiding encryption can be used to improve security.

1 Introduction

"Secure encryption" is today a very well-understood concept. However, standard cryptographic security definitions for encryption schemes (symmetric and asymmetric alike) consider a security experiment where an adversary chooses two plaintext messages m_0 and m_1, receives back an encryption of a randomly

Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823. We thank Hans-Jörg Bauer and Michael Simon, ZIM of University of Wuppertal, for their assistance with determining a real-world DNS host name distribution.

chosen message $m^* \in \{m_0, m_1\}$, and then has to determine which of the two messages was encrypted. A common, crucial restriction of such security definitions is that the two messages must have *equal length.* Otherwise it may be trivial to determine the encrypted message based on the length of the given ciphertext, such that security in this sense is impossible to achieve.

There are other standard security notions, such as "real-or-random" definitions, where an adversary has to distinguish between an encryption of a chosen message m and a random string of equal length or simulation-based semantic security, which requires the existence of an efficient simulator that produces the same output as the adversary given only the length of a ciphertext (e.g., [4,5,22,23]). All these definitions have in common that they do not provide security against attacks that are based on the length of messages.

Due to their simplicity and generality, these definitions have been extremely useful for building a general theory of secure encryption. However, assuming that only messages of equal length are encrypted is a necessary theoretical idealization and unrealistic from a practical perspective. In most real-world applications already the length of messages may reveal sensitive information.

The real-world relevance of hiding message lengths. Well-known examples of attacks leveraging message lengths consider a passive adversary that merely observes the encrypted network traffic and can identify web pages visited by a user [7,14,20,28,30,41], or the language and even phrases spoken in an encrypted voice-over-IP conversation [43,44], all this without breaking the expected security of the underlying encryption scheme. Even revealing the length of user passwords makes it possible to identify individual users in TLS-encrypted sessions and provides an advantage in password guessing attacks [17], in particular when passwords are re-used across different services. Hence, it is a desirable goal to hide the length of transmitted messages in practice.

Impossibility of hiding message lengths in cryptographic theory. Tezcan and Vaudenay [39] considered the *asymptotic* setting commonly used in theoretical cryptography and showed essentially that efficiently hiding the length of messages is impossible for arbitrary message distributions. Concretely, an exponential-sized padding is necessary, if the adversary in a standard security experiment is allowed to choose arbitrary messages of different lengths, and one aims at achieving a *negligible* distinguishing advantage. This suggests that *in theory* it is impossible to hide plaintext length efficiently, which supports the common belief that a considerable bandwidth overhead incurred by length-hiding padding is inevitable.

Dependence of encrypted messages and length hiding padding. In order to overcome this impossibility, Paterson, Ristenpart, and Shrimpton introduce the notion of length-hiding encryption (LHE) [32]. Essentially, LHE augments the encryption algorithm with an additional *length-hiding parameter* ℓ, which is specified by an application calling the encryption algorithm. The length-hiding parameter determines the amount of length-hiding padding used for a particular message. Secure LHE in the sense of [32] essentially guarantees security for plaintexts of different lengths, provided that the length hiding parameter ensures that the corresponding ciphertexts have equal size. However, [32] does not yet explain how ℓ can be chosen in order to obtain any security guarantees for realistic message distributions.

Furthermore, this work considers a classical "two-message indistinguishability" security model, where an adversary outputs two challenge messages m_0 and m_1 whose length difference must be *bounded* by some Δ, i.e., it holds that $0 \leq \big||m_0| - |m_1|\big| \leq \Delta$. Here, Δ depends on ℓ. Note that this requires the messages in the security experiment to be chosen depending on the length hiding parameter used by the underlying encryption scheme.

We argue that in order to determine suitable length-hiding padding to protect against fingerprinting attacks on a given application layer protocol, such as HTTP, DNS, etc., we do not want to make the distribution of application-layer messages dependent on the used padding scheme, but rather the other way around. That is, we want to determine a suitable length hiding parameter for the given message distribution of the application. Therefore we propose a security definition which does not mandate any *a priori* length difference Δ of messages, but rather quantifies the security of a certain padding length for a given message distribution (or an approximation thereof).

Quantifying the security of length-hiding encryption. There is currently no methodology that makes it possible to concretely assess and *quantify* the security of a given length-hiding padding scheme against fingerprinting attacks. In order to understand under which circumstances length-hiding schemes can reduce the effectivity of fingerprinting attacks *without* very large performance penalty, we have to analyze which *concrete* security guarantees can be obtained by length-hiding schemes with reasonable (i.e., non-exponential-sized) length-hiding parameters.

We know that suitable choice of a padding length must depend on the message distribution of an application (more precisely, on the distribution of the lengths of encrypted messages), as otherwise security is known to be not achievable [39]. In order to determine a suitable padding length for a given application in practice, it is therefore necessary to determine the message distribution of the given application. This can be achieved, for instance, by implementing a server-side monitoring algorithm that records (an approximation of) the distribution. This algorithm could run in set intervals and update the length-hiding parameter on the fly, if the distribution changes over time (e.g., due to changed web site contents or access patterns).

Our Contributions We develop a methodology that makes it possible to *concretely quantify* the effect of LHE on the security of a given application. To this end, we introduce a new cryptographic security model, which aims to preserve the simplicity and generality of classical models, while capturing security against fingerprinting attacks in order to reflect such security requirements of applications. Based on this definition, we describe a methodology to concretely quantify the effect of LHE for a given application.

In a next step, we demonstrate the feasibility of our approach by applying it to different types of fingerprinting attacks. Each of these scenarios cover multiple application-based aspects such as which block mode is used for encryption, or whether compression for transmitted data is enabled. Since the security impact of such aspects are often very subtle, we provide a more detailed explanation in Appendix 2. We summarize our results as follows:

Simple webpage fingerprinting. As a first example, we consider a website consisting of many static HTML pages, a user that visits one page, and an adversary that tries to determine the visited page based on the size of encrypted data. Since we want to base our analysis on a publicly-available web site with static contents, we used the IACR Cryptology ePrint archive at `https://eprint.iacr.org/2020/`.[1]

We find that switching from counter mode to block mode encryption already decreases the advantage of the adversary from 0.74 to 0.14 (where an advantage of 1 means that the adversary can uniquely determine a web page, while 0 means that the size of the transmitted data reveals no information about the visited page). This makes fingerprinting much less effective, without noticeable bandwidth overhead.

Reducing the advantage to 0 costs about 95% bandwidth overhead *without* compression, however, by additionally using compression and advantage to 0 can even be achieved *without* any overhead, by slightly reducing the amount of data transmitted by 0.3%. Hence, from a website fingerprinting perspective, it seems to make sense to enable LHE, possibly in combination with compression, on this server.

Web page fingerprinting with patterns. In order to analyze more complex fingerprinting attacks, we again consider the IACR Cryptology ePrint Archive and an adversary that tries to determine the visited page based on the size of encrypted data. This time, we consider a user that first visits the web page of a random paper from the year 2020, and then downloads the corresponding paper (a pdf file). Note that this yields a much more distinguishable pattern, in particular due to the highly varying size of pdf documents, and the fact that pdf files are not as easily compressible as text-based web pages.

We find that the advantage in counter mode is 1, that is, all papers are *uniquely* identifiable, such that the encryption provides no security at all against such attacks. This can be reduced to 0.12 by applying length-hiding padding with a bandwidth overhead of only about 2.4%. Hence, LHE can significantly improve security against fingerprinting at negligible overhead, which refutes the common belief that a significant overhead is necessary in order to achieve a considerable security improvement.

[1] We also considered basing this analysis on other web sites, such as Wikipedia and a user that accesses a certain Wikipedia page. However, the IACR ePrint server also enables us to easily consider a natural extension to more complex access pattern, see below.

Google search term fingerprinting. Here we consider the scenario that one user is searching some term in a search engine. A passive adversary observes the encrypted traffic and tries to determine which search term the user is searching for. We used 503 most popular search terms from the daily search trends published by Google at `https://trends.google.com/trends/` in a time period of one month in Spring 2021.

We find that without LHE an adversary achieves very high advantage of almost 1. LHE with only 2% bandwidth overhead can reduce this very significantly to only 0.07. In combination with compression, the advantage can be reduced to 0.006, while *reducing* the amount of transmitted data by 50%.

Simple Wikipedia fingerprinting. All the three application examples above consider a uniform message distribution, which does not necessarily capture the message distribution in the real-world applications. Obtaining real-world message distributions, e.g. by capturing Internet traffic, is difficult (for practical reasons, as well as due to privacy concerns). However, the Wikimedia Foundation publishes statistics of the Wikipedia website since May 2015, which provides us with the real distribution of visited webpages of the Wikipedia website. To better demonstrate the feasibility of our approach with respect to *real-world message distributions*, we carry out a webpage fingerprinting analysis for the Wikipedia webpages in simple English language, based on the real webpage visit distribution of May 2021.

We find that without LHE an adversary achieves very high trivial advantage (0.875). LHE with only 2% bandwidth overhead can reduce this to 0.13. In combination with compression, the advantage can be reduced to 0.0058, while *reducing* the amount of transmitted data by 50%.

DNS fingerprinting. Here we consider the *Domain Name System* (DNS) protocol and DNS request/response pairs. This setting is particularly interesting because there is an increasing trend to encrypt DNS protocol messages to hide the requested domain names. However, it turns out that the length of DNS requests/response pairs exhibit a very distinctive pattern, which make it easy to determine the requested domain name from the ciphertext length. We consider two different settings:

1. A user that issues a DNS request for a randomly chosen host name from 1,000 most popular hosts according to the *Majestic Million* list.
2. In collaboration with the IT department of a medium-sized university (with 23k students and 3.5k staff members), we collected the host names of DNS requests performed by staff and students within a 24 hr time interval in July 2021. The data collection was carried out under supervision of the university data privacy officer and in accordance with applicable data protection laws. In particular, only the hostnames and their frequency were collected, but not the requesting IP addresses or any other personal data. This provides us with a real-world message distribution that makes it possible to determine the security and appropriate padding sizes for this particular DNS service.

In both cases, the adversary that tries to determine the requested host name based on the size of request and response.

In the *Majestic Million* case, we find that the advantage of an adversary can be reduced from 0.644 (in counter mode without compression) down to 0.01 with a bandwidth overhead of about 79% without compression, or 57% with. For the university DNS case, the advantage of an adversary can be reduced from 0.551 (in counter mode without compression) to below 0.01 with a bandwidth overhead of about 54% without compression.

In summary, we find that LHE can, contrary to the common belief, improve security against fingerprinting attacks very significantly, often with minor bandwidth overhead. This also confirms the recent tendency to switch from block-mode to counter-mode encryption indeed makes fingerprinting attacks much more effective.

We support our theoretical model and calculations with a proof-of-concept implementation in form of an Apache module. Our implementation consists of two parts. One part is a server-side monitoring algorithm that records (an approximation of) the message distribution and computes a suitable length-hiding parameter. The other part applies the length-hiding parameter as input to the encryption procedure of server responses. We used this proof-of-concept implementation to validate the results of our theoretical analysis.

Application of our results. Some standards and implementations of cryptographic protocols already provide means to conceal the length of plaintexts, in order to prevent fingerprinting attacks or reduce their effectivity. One such example is the TLS 1.3 standard [34], which directly supports the use of length-hiding padding and functions as basis of our implementation.

Another example of an Internet standard that supports length-hiding padding is DNS. RFC 8467 [29] describes *block-length padding*, which recommends to pad all DNS requests to a multiple of 128 bytes and all DNS responses to a multiple of 468 bytes (as originally proposed in [13]). We remark that these numbers are derived from an empirical analysis conducted specifically for DNS, and that we do not yet have a clear methodology to quantify to which degree they improve security concretely.

Related Works The work of Boldyreva *et al.* [8] focuses on hiding message boundaries in a ciphertext stream with fragmented message transmission. This work is similar in spirit to ours in the sense that it tries to find a balance between the conflicting aims of keeping the generality and simplicity of traditional security definitions on the one hand, and developing an approach that can be used to provide meaningful provable security analyses of practical schemes on the other hand.

A very recent work on length hiding encryption is due to Degabriele [10], which will appear at ACM CCS 2021. We compare our approach with the one in [10] in **??**.

Fingerprinting attacks have been intensively studied in the context of web page fingerprinting, deanonymization of Tor private channels, and other applications such as LTE/4G. Early approaches were based on the length of encrypted messages, as well as their direction and the frequency of messages [19,20,28]. More recent approaches use additional features advanced analysis techniques based on machine learning [18,26,31,38]. Also active attacks have been considered [26,37]. Countermeasures to fingerprinting attacks were proposed [42,45], but could be broken with refined analysis methods [12,38]. Cai *et al.* [9] give a systematic analysis of attacks and defenses to understand what features convey the most information.

We note that many works on the feasibility of fingerprinting attacks make use of side-channels beyond message lengths and we discuss about this in Appendix A.

Outline of this paper. In Section 3 we describe a new perspective on LHE, which is more suitable for our approach, and conveniently yields schemes that follow the *standard* syntax of symmetric encryption schemes. Furthermore, we establish a definitional framework to analyze and quantify the concrete effect of LHE. Section 4 contains the results of an empirical analysis of different real-world message distributions. In Section 5 we present our implementation of LHE and compare its performance with our empirical results.

2 Further Context and Discussion

This section provides further context by discussing the impact of recent developments in secure communication protocols like TLS on the effectiveness of fingerprinting attacks. We argue that the increased use of Counter Mode encryption (such as AES-GCM, for instance) and the deprecation of compression in response to attacks such as CRIME, TIME, BREACH, and HEIST [3,11,33,40] has made fingerprinting attacks more efficient. In this light, we also discuss the controversial question whether compression before encryption is recommendable or not. We also survey related works on compression and encryption.

Increasing use of counter mode aids attacks based on message lengths. Older TLS versions preferred *block mode* encryption algorithms, such as CBC mode (e.g., the only mandatory cipher suite in TLS 1.2 is based on CBC). Since such block modes already need to pad a plaintext to a multiple of the block size of the underlying block cipher, the exact length of plaintexts is somewhat concealed, which provides some very minimalistic form of LHE.

However, in the past few years the percentage of TLS connections using *counter mode* encryption grew very significantly. A recent large-scale study [27] showed that about 80% of TLS connections use counter mode encryption (mostly AES-GCM). On the one hand, given the performance and security advantages of the authenticated encryption mode GCM over previous modes, this is a positive development. On the other hand, since the optional length-hiding padding is currently rarely used, counter mode encryption provides a passive adversary with very precise information about the length of a plaintext message, as it reveals the *exact* length of a plaintext to a passive adversary. This makes attacks that infer confidential information from the plaintext length much more effective, and additionally emphasizes the need for LHE and a methodology to select concrete length-hiding parameters for LHE schemes.

Deprecation of compression and attacks based on ciphertext lengths. Compression of data (with an algorithm like gzip) before encryption ("compress-then-encrypt") may equalize the size of ciphertexts of different plaintexts to some degree, and thus may contribute to concealing the length of the plaintext. Compression of plaintexts was optionally possible and often used (until a few years ago) in TLS versions up to 1.2. However, today it is considered good practice *not* to perform compression of data before encryption, and the option to compress plaintexts was deprecated in TLS 1.3. This is mainly due to *partially chosen-plaintext attacks* [25], where an adversary injects chosen data that get compressed along with secret data before encryption. The length of the resulting ciphertexts reveals information about the redundancy in the adversarially-chosen and the secret data, which allows the adversary to draw conclusions about this secret data. The practical relevance was demonstrated by attacks like CRIME, TIME, BREACH, and HEIST [3,11,33,40], which may allow an *active* adversary to steal HTTP session cookies, for instance. These attacks require an adversary that is able to perform a *partially chosen-plaintext attack*, which in some applications is very realistic, but in others is not.

While the deprecation of compression prevents such active attacks in applications where partially chosen-plaintext attacks are practically feasible, at the same time it provides a *passive* adversary with more accurate information about the size of plaintext data, even in applications where the partially chosen plaintext attacker model does not apply because an adversary is *not* able to inject data that gets encrypted along with confidential data. Hence, the effect of countermeasures may depend on the concrete attacker model at hand. If partially chosen-plaintext attacks provide a practical threat in a given application context, then disabling compression is necessary. However, if the design of an application does not allow such attacks, then disabling compression provides a passive adversary with more precise information about the size of encrypted data, which may be harmful to security in another attacker model.

Attacks like CRIME and its variants [3, 11, 33, 40] targeted HTTP session cookies in HTTP versions 1 and 1.1. Compression was performed on the underlying cryptographic protocol layer, by TLS. Now, due to the significant bandwidth gain, the most recent version HTTP/2 [6] re-introduces compression on the application layer. The possibility of partially chosen-plaintext attacks is explicitly considered, and the HTTP/2 standard [6] requires to disable compression in applications where such attacks are possible, or alternatively to use a separate compression dictionary for potentially adversarially-controlled data. Hence, for HTTP/2 a passive website fingerprinting attacker still needs to be considered, while active partially chosen-plaintext attacks based on compression are already addressed by the standard. Hence, we consider it interesting to analyze the security impact of compression on the effectivity of fingerprinting attacks.

We note also that the passive attacker model is particularly relevant when considering a "big brother" adversary, which monitors Internet traffic at such a large scale that widespread active attacks at this scale become practically infeasible. Such an adversary may store large amounts of encrypted data for later analysis. If the analysis takes place after the encrypted session has finished, then an active attack is not possible anymore.

On using compress-then-encrypt. Our analysis raises the controversial question whether compression before encryption is recommendable or not. From a website fingerprinting perspective with passive attacker model, compression appears to be useful, as this improves security against such attacks while even *saving* bandwidth. We stress, however, that we do not generally recommend the use of compression. Compression must only be used if it is clear that active attacks do not apply. This may either be the case if compression-based attacks are already considered and prevented on the application layer, as in some applications of HTTP/2, for instance, or in applications where active attacks seem not to apply, due to the nature or design of the considered application, or if they are considered out of scope since a different attacker model (e.g. the "big brother" adversary) is considered. Still, we conclude that the *general* advice to disable compression, independent of a given attacker model, can be misleading and harmful to security. It is necessary to carefully consider the concrete security requirements of the application at hand.

Related work on compress-then-encrypt The possibility of using compress-then-encrypt securely has been considered in [2, 24]. However, the security notion in [24] does not yet capture the aforementioned real-world attacks [3, 11, 33, 40]. Alawatugoda *et al.* [2] give security notions that focus on the security of HTTP Cookies in compress-then-encrypt schemes (which is the main target of [3,11,33,40]). They consider the trivial countermeasure of simply *excluding* the cookie from compression, which requires the compress-then-encryption scheme to "know" which input is "secret" and which not, which is essentially the approach followed by HTTP/2 [6]. They also consider the approach of using a restricted class of compression algorithms with fixed dictionary, but also show that this yields a rather bad compression rate for complex plaintext data.

3 A New Perspective on Length-Hiding Encryption

Now we can describe our new perspective on length-hiding encryption (LHE), which makes it possible to concretely quantify the effect of length-hiding padding on security. We first introduce a new syntactical notion of LHE in Section 3.1 and a corresponding security experiment in Section 3.2. Based on these foundations, we can then define the *trivial success probability* and the *trivial advantage* to concretely quantify the security of an encryption scheme with respect to different length-hiding parameters.

In Appendix B we also discuss k-anonymity as an alternative approach and explain why we consider it as not very suitable for our purposes.

3.1 A New Syntactical Definition

We first recap the formal definition of symmetric-key encryption.

Definition 1 (Symmetric-key encryption). *A* symmetric-key encryption scheme SE *consists of two algorithms* E *and* D*. The (possibly) randomized encryption algorithm* E *takes input a secret key* sk *and a plaintext* $m \in \{0,1\}^*$*, and outputs a ciphertext* ct*. The deterministic decryption algorithm* D *takes input a secret key* sk *and a ciphertext* ct *then outputs a plaintext* m *or a symbol* $\perp$*. The correctness of* SE *requires that* $\mathsf{D}_{\mathsf{sk}}(\mathsf{E}_{\mathsf{sk}}(m)) = m$ *for all* $\mathsf{sk} \in \{0,1\}^k$ *and* $m \in \{0,1\}^*$*.*

Note that, in the above definition, the ciphertext length is implicitly defined by the encryption algorithm and cannot be altered or controlled outside the algorithm. Paterson *et al.* introduced the concept of *length-hiding encryption* in [32], which makes it possible to control the ciphertext length outside the encryption algorithm. In this paper we will work with a slightly different perspective, which we consider as more practical.

Recall that in [32] the length-hiding parameter ℓ is the total *ciphertext length* and it is an explicit input to the encryption algorithm. The parameter can be controlled by the adversary in the security experiment, the only restriction is that it must be at least as large as the largest of the two messages submitted by the adversary in the indistinguishability security experiment. We believe this does not capture real-world attacks based on ciphertext lengths very well, because it considers only the indistinguishability of two messages that are encrypted with respect to the same ciphertext length ℓ.

To protect against such attacks, we find it more practical to view the length hiding parameter as a fixed system parameter and to make the ciphertexts length dependent on both ℓ and the size of encrypted messages. More precisely, the global parameter ℓ (of a symmetric-key encryption scheme) defines a fixed *function* which maps plaintext length to certain ciphertext length.

Now, for any symmetric-key encryption scheme SE, we formalize a new symmetric-key encryption scheme $\mathsf{SE}^{(\ell)}$ as the scheme SE instantiated with length hiding parameter ℓ as follows.

Definition 2 (Symmetric-key encryption with length-hiding parameter ℓ). *Let* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *be a symmetric encryption scheme. Let* $\mathsf{pad}(m, \ell)$ *be a function that pads a plaintext* m *such that*

$$|\mathsf{pad}(m, \ell)| = \left\lceil \frac{|m|}{\ell} \right\rceil \cdot \ell.$$

That is, pad *applies length-hiding padding to the plaintext* m *such that the length of* $\mathsf{pad}(m, \ell)$ *is an integer multiple of* ℓ*. Let* pad^{-1} *be the function that removes the padding, that is,* $\mathsf{pad}^{-1}(\mathsf{pad}(m, \ell)) = m$ *for all* m *and* $\mathsf{pad}^{-1}(\perp) = \perp$*. We call* ℓ *the* length-hiding parameter *and define the length-hiding encryption scheme* $\mathsf{SE}^{(\ell)} = (\mathsf{E}^{(\ell)}, \mathsf{D}^{(\ell)})$ *as*

$$\mathsf{E}^{(\ell)}_{\mathsf{sk}}(m) := \mathsf{E}_{\mathsf{sk}}(\mathsf{pad}(m, \ell)) \quad \textit{and} \quad \mathsf{D}^{(\ell)}_{\mathsf{sk}}(m) := \mathsf{pad}^{-1}(\mathsf{D}_{\mathsf{sk}}(m)).$$

Note that in particular we have $\mathsf{SE}^{(1)} = \mathsf{SE}$.

Padding a message to a multiple of some parameter ℓ is a generalization of the *block-length padding* scheme proposed in [13] and recommended for encrypted DNS requests in RFC 8467 [29].

Note that this scheme allows to capture "perfect" length-hiding padding, where the length-hiding parameter ℓ is at least as large as the largest possible message in an application, such that all ciphertexts have identical length and no information is leaked through the length. However, if ℓ is smaller, then we will get weaker security guarantees. We will show that this is very useful to achieve a trade-off between improved resilience to attacks based on message length and bandwith overhead of the length-hiding padding. Furthermore, we will analyze the concrete impact of different length-hiding parameters on the security and bandwidth requirements of an application.

Security notions for symmetric-key encryption schemes. Since $\mathsf{SE}^{(\ell)}$ follows the standard syntax of encryption schemes for all $\ell \in \mathbb{N}$, standard security definitions (without considering length leakage) for symmetric-key encryption apply to it. There are many different ways to define security for symmetric-key encryption schemes. In this paper we will consider two flavors of standard IND-CCA security, the IND-M-CCA security and the IND-C-CCA security. Both provide protection for message privacy but none of them provides protection against length leakage. However, they serve as an important tool in our approach to capture the effectiveness of length-hiding encryption.

IND-M-CCA security. The first notion is the indistinguishability of encryptions of chosen *messages* from encryptions of random strings. It is equivalent to the classical "left-or-right" definition where an adversary outputs two messages m_0 and m_1, receives back an encryption of message m_b for a random $b \xleftarrow{\$} \{0,1\}$, and has to determine b.

Definition 3 (IND-M-CCA security). *A symmetric-key encryption scheme* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is* $\mathsf{IND\text{-}M\text{-}CCA}$ secure *if for any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{SE},\mathcal{A}}_{\mathsf{IND\text{-}M\text{-}CCA}}(1^k) := \left|2 \cdot \Pr\left[\mathsf{IND\text{-}M\text{-}CCA}^{\mathsf{SE},\mathcal{A}}(1^k) = 1\right] - 1\right|$$

is negligible, where game $\mathsf{IND\text{-}M\text{-}CCA}^{\mathsf{SE}}$ *is defined in Figure 1.*

IND-C-CCA security. The second notion we consider is indistinguishability of *ciphertexts* from random strings (the "IND" notion in [22]), which is commonly used in recent works, such as [15, 16]. This security considers the "ciphertext pseudorandomness" of the symmetric-key encryption scheme.

Definition 4 (IND-C-CCA security). *A symmetric-key encryption scheme* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is* $\mathsf{IND\text{-}C\text{-}CCA}$ secure *if for any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{SE},\mathcal{A}}_{\mathsf{IND\text{-}C\text{-}CCA}}(1^k) := \left|2 \cdot \Pr\left[\mathsf{IND\text{-}C\text{-}CCA}^{\mathsf{SE},\mathcal{A}}(1^k) = 1\right] - 1\right|$$

```
Proc. Initialize($1^k$):
    $b \xleftarrow{\$} \{0,1\}$
    $\mathsf{sk} \xleftarrow{\$} \{0,1\}^k$
    $\mathcal{C} \leftarrow \emptyset$
    Return

Proc. Enc($m$):
    Return $\mathsf{E}_{\mathsf{sk}}(m)$

Proc. Dec($\mathsf{ct}$):
    Return $\begin{cases} \mathsf{D}_{\mathsf{sk}}(\mathsf{ct}) & \text{If } \mathsf{ct} \notin \mathcal{C} \\ \perp & \text{Otherwise} \end{cases}$

Proc. Challenge($m$):
    $m_0 \leftarrow m$
    $m_1 \xleftarrow{\$} \{0,1\}^{|m|}$
    $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_{\mathsf{sk}}(m_b)$
    $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_{\mathsf{sk}}(m)$
    $r \xleftarrow{\$} \{0,1\}^{|\mathsf{ct}|}$
    $\mathsf{ret} \leftarrow \begin{cases} \mathsf{ct} & \text{If b=0} \\ \mathsf{r} & \text{Otherwise} \end{cases}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{ret}\}$
    Return $\mathsf{ret}$

Proc. Finalize($b'$):
    Output $(b' = b)$
```

Fig. 1. The security game $\mathsf{IND\text{-}M\text{-}CCA}^{\mathsf{SE}}$ and the security game $\mathsf{IND\text{-}C\text{-}CCA}^{\mathsf{SE}}$. The gray background codes only applies to game $\mathsf{IND\text{-}C\text{-}CCA}^{\mathsf{SE}}$.

is negligible, where game $\mathsf{IND\text{-}C\text{-}CCA}^{\mathsf{SE}}$ *is defined in Figure 1.*

Since the length-hiding parameter ℓ is a global system parameter and known to the adversary (the adversary does not have to break the scheme or make any encryption or decryption queries to know this parameter), we point out that the length-hiding property is orthogonal to the exact oracle query ability of the adversary. Our choice to consider security against chosen-ciphertext attacks should merely serve as a concrete example. For instance, CPA security is easily obtained by removing the decryption oracle, passive eavesdropping by additionally removing the encryption oracle. It is also possible to consider advanced security notions, such as misuse-resistant encryption [35], where the adversary is able to specify the nonce used by an encryption algorithm. Furthermore, it can be considered in both the symmetric-key and the public-key setting. We chose security against chosen-ciphertext attacks in order to demonstrate that the approach works also with this rather strong standard security definition.

3.2 New Security Model

Our objective is to define a model that is independent of a particular application, but capable of capturing complex application settings where multiple messages of varying lengths are encrypted using multiple keys and the adversary wants to deduce information about these messages from the ciphertexts. Therefore we parametrize our security model as follows:

Message distribution. Note that the information an adversary can deduce from the size of observed ciphertexts depends inherently on the message distribution of a given application, and possibly other application-specific properties, such as observable patterns of messages, for instance. Since we

want to preserve as much of the simplicity and generality of classical security models as possible, our new security definition is parametrized by a *message distribution* M of a given application.
For instance, in the web page fingerprinting setting, the distribution M would assign probabilities to different web pages on a server. If a client performs a search with an Internet search engine, then $\boldsymbol{m}^* \xleftarrow{\$} \mathsf{M}$ would correspond to the messages exchanged between a client and the search engine. Note that, in this case, the message distribution M is defined by the client and the server may not know this distribution M beforehand. However, we consider strong adversaries who knows M so that it obtains some a prior knowledge about this search term.

Specification of data to-be-protected. We introduce a function $\mathcal{P}$ which specifies the information that an adversary wants to learn about the encrypted data. For instance, if a client performs a search with an Internet search engine and the adversary wants to learn the search term, then the data exchanged between client and search engine would be $\boldsymbol{m}^* \xleftarrow{\$} \mathsf{M}$, and the goal of the adversary would be to determine the search term $\mathcal{P}(\boldsymbol{m}^*)$.

Multiple symmetric keys. We want to be able to consider complex fingerprinting attacks on applications transmitting encrypted data. For instance, in the context of Internet search engines a client might first connect to a search engine and to perform a search, then connect to a DNS server to make a DNS request for the first search result, and then connect to the server hosting the corresponding web site. We will consider a setting where all three connections are encrypted, such that this involves three different encrypted sessions with three different, independent symmetric keys.[2] In order to reflect this accurately, we need to define a security model which involves d symmetric keys.

Adversarial model. Finally, we need to specify the capabilities that we grant the attacker. We will consider *chosen-ciphertext attacks* (CCA), as already discussed in Section 3.1.

In Section 4, we will show how to define M, $\mathcal{P}$, and d for a particular given applications to obtain concrete security statements for this application. In most practical cases, $\mathcal{P}$ will be efficiently computable, but we do not have to demand this.

Security experiment. Motivated by this discussion, we define the $(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)$-$\mathsf{CCA}$ security experiment based on the game described in Figure 2. In this game, we consider an encryption scheme SE with $d \geq 1$ independent secret keys and a message distribution M that outputs $t \geq 1$ messages where $t = t_1 + \ldots + t_d$ for some $t_1, \ldots, t_d \in \mathbb{N}$. We model M as a probabilistic algorithm defining a message distribution over $(\{0,1\}^*)^t$. In the sequel it will be convenient to view the output of M as a tuple of tuples of messages $\boldsymbol{m} = ((m_{1j})_{j\in[t_1]}, \ldots, (m_{dj})_{j\in[t_d]}) \xleftarrow{\$} \mathsf{M}$.

[2] In practice the TLS protocol would be used for these three connections, where different keys are used for sending and receiving data, but we view these keys as a single symmetric key.

Here $(m_{dj})_{j \in [t_i]}$ is the tuple of messages encrypted under the i-th symmetric key sk_i. The function $\mathcal{P}$ maps $\mathcal{P} : (\{0,1\}^*)^t \to \{0,1\}^*$.

```
Proc. INITIALIZE($1^k$):
$\mathsf{sk}_1, \cdots, \mathsf{sk}_d \xleftarrow{\$} \{0,1\}^k$
$\mathsf{challenged} \leftarrow \mathtt{False}$
$\mathcal{C} \leftarrow \emptyset$
Return $(\mathsf{M}, \mathcal{P})$

Proc. ENC($i, m$):
Return $\mathsf{E}_{\mathsf{sk}_i}(m)$

Proc. DEC($i, \mathsf{ct}$):
If $(i, \mathsf{ct}) \in \mathcal{C}$:
    Return $\perp$
Return $\mathsf{D}_{\mathsf{sk}_i}(\mathsf{ct})$
```

```
Proc. CHALLENGE():
If $\mathsf{challenged} = \mathtt{True}$ return $\perp$
$\mathsf{challenged} \leftarrow \mathtt{True}$
$\underbrace{\big((m^*_{1j})_{j \in [t_1]}, \cdots, (m^*_{dj})_{j \in [t_d]}\big)}_{\boldsymbol{m}^*} \xleftarrow{\$} \mathsf{M}$
For $i \in [d], j \in [t_i]$:
    $\mathsf{ct}^*_{ij} \xleftarrow{\$} \mathsf{E}_{\mathsf{sk}_i}(m^*_{ij})$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(i, \mathsf{ct}^*_{ij})\}$
Return $(\mathsf{ct}^*_{ij})_{i \in [d], j \in [t_i]}$

Proc. FINALIZE($p$):
Output $(\mathcal{P}(\boldsymbol{m}^*) = p)$
```

Fig. 2. The $(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)$-$\mathsf{CCA}$ security game with respect to encryption scheme SE, message distribution M, property $\mathcal{P}$ and number of secret key d.

Definition 5. *For any adversary $\mathcal{A}$, define* the success probability *of $\mathcal{A}$ in game* $(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)$-$\mathsf{CCA}$ *as the probability that $\mathcal{A}$ is able to determine $\mathcal{P}(\boldsymbol{m}^*)$. Formally:*

$$\mathsf{RealSucc}(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d, \mathcal{A}) := \Pr\left[(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)\text{-}\mathsf{CCA}^{\mathcal{A}}(1^k) = 1\right].$$

Non-triviality of M *and* $\mathcal{P}$. In the sequel we will only consider M and $\mathcal{P}$ such that there exist $\boldsymbol{m}$ and $\boldsymbol{m'}$ such that $\Pr[\boldsymbol{m} \xleftarrow{\$} \mathsf{M}] > 0$, $\Pr[\boldsymbol{m'} \xleftarrow{\$} \mathsf{M}] > 0$, and $\mathcal{P}(\boldsymbol{m}) \neq \mathcal{P}(\boldsymbol{m'})$. This is necessary for technical reasons, but also sufficient because we would otherwise not get a meaningful notion of security (because otherwise it would be trivial to predict $\mathcal{P}$ with respect to M with success probability 1, even without seeing any ciphertexts).

Our security definition implies the IND-M-CCA and we discuss this in Appendix C.

3.3 Trivial Success Probability and Advantage

Trivial success probability. We introduce the *trivial success probability* of an adversary, which captures the success probability that an adversary trivially obtains from the *length* of ciphertexts, without "breaking" the underlying encryption scheme SE. Note that this depends on the degree to which SE hides the size of plaintexts, the message distribution M, and the function $\mathcal{P}$.

Definition 6. *The* trivial success probability *with respect to* $(\mathsf{SE}, \mathsf{M}, \mathcal{P})$ *is defined as the largest possible probability of guessing $\mathcal{P}(\boldsymbol{m}^*)$, given M, $\mathcal{P}$, and the* size *of the ciphertexts* $(|\mathsf{ct}^*_{ij}|)_{i\in[d],j\in[t_i]}$*, but not the ciphertexts themselves. More formally,*

$$\mathsf{TrivSucc}(\mathsf{SE},\mathsf{M},\mathcal{P}) := \max_{\mathcal{S}} \Pr\left[\mathcal{S}(\mathsf{M},\mathcal{P},(|\mathsf{ct}^*_i|)_{i\in[t]}) = \mathcal{P}(\boldsymbol{m}^*)\right]$$

where $\boldsymbol{m}^* = \Big((m^*_{1j})_{j\in[t_1]},\cdots,(m^*_{dj})_{j\in[t_d]}\Big) \xleftarrow{\$} \mathsf{M}$, $\mathsf{sk}_1,\cdots,\mathsf{sk}_d \xleftarrow{\$} \{0,1\}^k$ *and* $\mathsf{ct}^*_{ij} \xleftarrow{\$} \mathsf{E}_{\mathsf{sk}_i}(m^*_{ij})$ *for* $i\in[d], j\in[t_i]$.

The following property is necessary to avoid contrived and unnatural examples of encryption schemes, where the length of ciphertexts depends not only on the size of the message, but also on the secret key. To our best knowledge, this property is met by any concrete symmetric encryption schemes, except for contrived counterexamples.

Definition 7. *We say an encryption scheme* $\mathsf{SE} = (\mathsf{E},\mathsf{D})$ *has* secret key oblivious ciphertext length distribution *if for any* $m \in \{0,1\}^*$*, any* $\mathsf{sk},\mathsf{sk}' \in \{0,1\}^k$, $|\mathsf{E}_{\mathsf{sk}}(m)|$ *distributes identically to* $|\mathsf{E}_{\mathsf{sk}'}(m)|$.

If we would restrict to settings with deterministic padding length, then we could instead require $|\mathsf{E}_{\mathsf{sk}}(m)| = |\mathsf{E}_{\mathsf{sk}'}(m)|$. However, we will also consider randomized padding, so that it is necessary to require identical *distribution* of ciphertext lengths.

Relation between trivial and real success probability. We will now prove that $\mathcal{A}$'s real success probability $\mathsf{RealSucc}(\mathsf{SE},\mathsf{M},\mathcal{P},d,\mathcal{A})$ cannot exceed the trivial success probability $\mathsf{TrivSucc}(\mathsf{SE},\mathsf{M},\mathcal{P})$ significantly, provided that SE is IND-C-CCA secure in the sense of Definition 4 and satisfies Definition 7. This establishes that in order to minimize $\mathsf{RealSucc}(\mathsf{SE},\mathsf{M},\mathcal{P},d,\mathcal{A})$ it suffices to minimize $\mathsf{TrivSucc}(\mathsf{SE},\mathsf{M},\mathcal{P})$.

Theorem 8. *For any encryption scheme* SE *that has secret key oblivious ciphertext length distribution in the sense of Definition 7, any message distribution* M*, any property* $\mathcal{P}$ *and any adversary* $\mathcal{A}$*, we can construct an adversary* $\mathcal{B}$ *such that*

$$\mathsf{RealSucc}(\mathsf{SE},\mathsf{M},\mathcal{P},d,\mathcal{A}) \le d\cdot\mathsf{Adv}^{\mathsf{SE},\mathcal{B}}_{\mathsf{IND\text{-}C\text{-}CCA}}(1^k) + \mathsf{TrivSucc}(\mathsf{SE},\mathsf{M},\mathcal{P}).$$

The running time of $\mathcal{B}$ *is* $T(\mathcal{B}) \approx T(\mathcal{A})+T(\mathsf{M})+T(\mathcal{P})+(t+q_e)\cdot T(\mathsf{E})+q_d\cdot T(\mathsf{D})$*, where* $T(\mathsf{M})$ *is the time to sample messages from distribution* M*,* $T(\mathcal{P})$ *is the time to evaluate* $\mathcal{P}$*,* $T(\mathsf{E})$ *is the time to execute the encryption algorithm once,* $T(\mathsf{D})$ *is the time to execute the decryption algorithm once,* q_e *is the number of* ENC *queries made by* $\mathcal{A}$ *and* q_d *is the number of* DEC *queries made by* $\mathcal{A}$.

Due to space limitations, we put the proof of Theorem 8 in Appendix D.

Theorem 8 provides an upper bound of the adversary's success probability in our practical "real world" security model and this bound is a sum of the CCA advantage with the trivial success probability. So if the encryption scheme itself is secure in the standard sense (i.e., the IND-C-CCA advantage of any adversary is negligible, which is provided IND-C-CCA security of the scheme) then the sum is close to the trivial advantage. In this way, we can focus on reducing the trivial success probability by choosing a proper length-hiding encryption scheme to reduce the information that is leaked through ciphertext length.

In the following, we show a methodology of evaluating the "effectivness" of length-hiding encryption in reducing the trivial success probability. Intuitively speaking, we first define the "most likely probability" to capture prior information leaked by the distribution M known to the adversary. And this probability provides a lower bound for the trivial success probability. For any concrete encryption scheme, we compare its trivial success probability with this lower bound to evaluate its "effectivness" in hiding the length information. And we further quantify it by providing a new definition of *trivial advantage*.

Trivial advantage. Based on the trivial success probability, we define the *trivial advantage* with respect to encryption scheme SE, message distribution M, and property $\mathcal{P}$ as follows.

Definition 9. *For message distribution* M *and property* $\mathcal{P}$*, let*

$$\mathsf{PrMostLikely}(\mathsf{M}, \mathcal{P}) := \max_{\mathcal{S}} \Pr\left[\mathcal{S}(\mathsf{M}, \mathcal{P}) = \mathcal{P}(\boldsymbol{m}^*)\right]$$

denote the probability of the most likely output of $\mathcal{P}$ *on input* $\boldsymbol{m}^*$*, where* $\boldsymbol{m}^* = \Big((m^*_{1j})_{j\in[t_1]}, \cdots, (m^*_{dj})_{j\in[t_d]}\Big) \xleftarrow{\$} \mathsf{M}$*. The* trivial advantage *with respect to* $(\mathsf{SE}, \mathsf{M}, \mathcal{P})$ *is defined as difference between the trivial success probability and* $\mathsf{PrMostLikely}(\mathsf{M}, \mathcal{P})$*, scaled to an number that ranges from* 0 *to* 1 *for any* M *and* $\mathcal{P}$*:*

$$\mathsf{TrivAdv}(\mathsf{SE}, \mathsf{M}, \mathcal{P}) := \frac{\mathsf{TrivSucc}(\mathsf{SE}, \mathsf{M}, \mathcal{P}) - \mathsf{PrMostLikely}(\mathsf{M}, \mathcal{P})}{1 - \mathsf{PrMostLikely}(\mathsf{M}, \mathcal{P})}.$$

Recall that we consider M and $\mathcal{P}$ such that there exist $\boldsymbol{m}$ and $\boldsymbol{m'}$ such that $\Pr[\boldsymbol{m}^* = \boldsymbol{m}] > 0$, $\Pr[\boldsymbol{m}^* = \boldsymbol{m'}] > 0$, and $\mathcal{P}(\boldsymbol{m}) \neq \mathcal{P}(\boldsymbol{m'})$, as otherwise it is trivial to predict $\mathcal{P}$ with respect to M with success probability 1. Therefore we have $\mathsf{PrMostLikely}(\mathsf{M}, \mathcal{P}) < 1$.

4 Analysis of Real-World Message Distributions

In this section, we will consider different types of attacks, where an adversary observes encrypted communication and wants to determine the plaintext based on the communication pattern. As a concrete example we consider counter mode encryption (with AES-GCM, for instance) where the adversary is able to determine the size of the encrypted message precisely, as well as block mode encryption (using AES-CBC as an example) where the adversary is able to determine the size of the encrypted message up to the minimal message padding required for this mode. We apply our methodology to concretely calculate the trivial advantage for the considered message distribution M and function $\mathcal{P}$ that we define for each application. Then we compare the results to the same schemes, but using length hiding encryption with different length hiding parameters. This makes it possible to quantify the security gained by length-hiding encryption and compression precisely. Most interestingly, we will show that relatively small length-hiding parameters are able to significantly reduce the effectivity of fingerprinting attacks.

In all the subsequent parts of this work, we will use different units for the length hiding parameter for counter mode and block mode encryption. The reason for this is that counter mode ciphers (like AES-GCM) usually encrypt messages *byte-wise*, therefore the size of ciphertexts and the length-hiding parameter ℓ are measured in bytes. Block ciphers (like AES-CBC) encrypts messages *block-wise*, so we will measure the size of ciphertexts and ℓ in blocks, where one block corresponds to the block size of the underlying block cipher (16 bytes in case of AES).

4.1 Simple Website Fingerprinting

Adversarial model. We consider an adversary performing a *website fingerprinting* attack against the IACR ePrint archive[3] at `https://eprint.iacr.org/2020/`, based on the size of encrypted data. The ePrint page for the year 2020 links to 1620 different subpages, one for each archived research papers. We chose the year 2020 because it is the most recent year where the number of archived research papers was fixed at the time this paper is written, which makes the analysis more easily reproducible. Each subpage contains the title, list of authors, abstract, and some other metadata for one archived research paper.

Let $\mathcal{M} := \{m_1, \ldots, m_{1620}\}$ be the set of the 1620 different web pages. We assume that a user Alice picks one out of these 1620 subpages uniformly at random and then visits the corresponding page. Hence, the message distribution $\mathsf{M_e}$ that we consider here is the uniform distribution over $\mathcal{M}$. The adversary tries to guess the subpage visited by Alice. Hence, the adversary outputs an index $i \in \{1, \ldots, 1620\}$, such that we define function $\mathcal{P}$ as $\mathcal{P} : \mathcal{M} \to [1, 1620]$ for $m_i \mapsto i$.

Considered encryption schemes. The IACR ePrint server uses TLS 1.2 and allows the client and the server to negotiate one out of two different symmetric ciphers for the encryption of payload data:

Counter mode. The first option is AES-GCM, which is essentially a stream cipher and we refer it as $\mathsf{SE}^{(\ell)}_{\mathrm{CTR}}$. Note that in this case the attacker learns the size of the underlying plaintext *exactly* (in case of TLS the attacker merely has to subtract a known constant from the ciphertext size), provided that no length-hiding padding is used ($\ell = 1$).

Block mode. The second option is AES-CBC and we refer it as $\mathsf{SE}^{(\ell)}_{\mathrm{BLK}}$. Due to the padding required by CBC-mode encryption, the attacker learns only that the plaintext lies within a certain (small) range even if no length-hiding

[3] We use this server since URLs from the IACR ePrint archive are particularly easy to parse and analyse.

padding is used ($\ell = 1$). Since the block size of AES is 16 bytes, AES-CBC uses a padding of length between 1 and 16 bytes, this holds also for AES-CBC in TLS 1.2.

We decided to use these two algorithms as the basis of our analysis, since they cover both a stream cipher and a block mode cipher, and are very widely used in practice. We consider a length-hiding padding with length-hiding parameter $\ell \in \mathbb{N}$. Ciphertexts are padded to a multiple of ℓ *bytes* for $\mathsf{SE}^{(\ell)}_{\mathrm{CTR}}$, and ℓ *blocks* for $\mathsf{SE}^{(\ell)}_{\mathrm{BLK}}$. Thus, if $\ell = 1$, then no additional length-hiding padding is used.

We also consider enabled compression where the full plaintext message is compressed using the `gzip` algorithm, which implements DEFLATE compression (a combination of the LZ77 algorithm and Huffman coding). This is the algorithm standardized for use in TLS versions up to 1.2 [21], and therefore seems to be a reasonable choice for an analysis of real-world algorithms.

In order to indicate whether compression is used, we will refer to the encryption algorithm as $\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}$ where $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$ represents its mode and $\mathrm{C} \in \{\mathrm{True}, \mathrm{False}\}$ represents whether compression is applied before encryption.

Empirical data generation. We implemented a simple script that downloaded all subpages of `https://eprint.iacr.org/2020/` and determined the size of the page in both uncompressed and compressed form. These sizes are stored in a database and enable us to compute the size[4] of counter- and block-mode ciphertexts that encrypt the pages in compressed or uncompressed form, with and without length-hiding encryption, and with respect to different length hiding parameters.

Calculating the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e})$. We determined the number $|U|$ of uniquely identifiable web pages, the size $|S_{\mathsf{max}}|$ of the largest set of pages of identical size, the number $|\mathcal{S}|$ of different possible ciphertext lengths, and the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e})$ for both counter mode and block mode encryption, with and without compression, and with respect to different length hiding parameters, and with respect to the uniform distribution $\mathsf{M_e}$ over the ePrint 2020 webpages.

Recall that the trivial success probability is defined as

$$\mathsf{TrivSucc}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e}, \mathcal{P}) := \max_{\mathcal{S}} \Pr\left[\mathcal{S}(\mathsf{M_e}, \mathcal{P}, |\mathsf{ct}^*|) = \mathcal{P}(m^*)\right]$$

where the probability is over the random choice $m^* \xleftarrow{\$} \mathsf{M_e}$ and ct^* is the encryption of m^*. Since $\mathsf{M_e}$ is the uniform distribution, the trivial success probability for a given ciphertext length λ is equal to the number of messages $m^* \in \mathcal{M}$ that encrypt to a ciphertext of the given length $\lambda = |\mathsf{ct}^*|$. Hence, defining

[4] To make our calculation more realistic, we consider the ciphertext is split across TLS fragments with a maximum payload of 2^{14} bytes per fragment, whereby each fragment needs an additional 22 bytes reserved for the fragment header. The length for all the TLS headers is also considered and summed to the ciphertext length.

$\mathcal{M}_\lambda := \{m \in \mathcal{M} : |\mathsf{E}_{\mathsf{sk}}(m)| = \lambda\}$ as the set of messages that encrypt to a ciphertext of size λ, we have

$$\mathsf{TrivSucc}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e}, \mathcal{P}) = \sum_{\lambda \in \mathcal{S}} \frac{|\mathcal{M}_\lambda|}{|\mathcal{M}|} \cdot \frac{1}{|\mathcal{M}_\lambda|} = \frac{|\mathcal{S}|}{|\mathcal{M}|} = \frac{|\mathcal{S}|}{1620}.$$

Hence, the trivial advantage is $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e}, \mathcal{P})$

$$= \frac{\mathsf{TrivSucc}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e}, \mathcal{P}) - \mathsf{PrMostLikely}(\mathsf{M_e}, \mathcal{P})}{1 - \mathsf{PrMostLikely}(\mathsf{M_e}, \mathcal{P})}$$
$$= \left(\frac{|\mathcal{S}|}{1620} - \frac{1}{1620}\right) \div \left(1 - \frac{1}{1620}\right) = \frac{|\mathcal{S}| - 1}{1619}.$$

We consider length-hiding block mode ciphertexts with $\ell \in \{1, 5, 10, 25\}$ and counter mode ciphertexts with $\ell \in \{1, 80, 160, 400\}$. Table 1 summarizes the results of this analysis.

Achieving a trivial advantage of 0. Note that according to Table 1 an increasing length hiding parameter ℓ reduces the trivial advantage significantly, but not to 0. This may be sufficient to make traffic analysis attacks significantly less effective, but for some applications a trivial advantage of 0 may be desirable.

Rows 9,10,19 and 20 of Table 1 give the minimal length hiding parameter ℓ that is necessary to reduce the trivial advantage to 0.

Conclusions. For the message distribution considered in this section, we come to the following conclusions:

– A block mode of operation improves the indistinguishability of the considered web pages, compared to a counter mode when $\ell = 1$. In combination with length-hiding encryption and compression, the adversary's trivial advantage can be reduced from 0.7400 (in counter mode without compression) down to 0.0037 (in block mode with compression and length hiding parameter $\ell = 25$), together with a reduction in traffic by 45.278%.
– In most cases compression reduces the trivial advantage of the adversary by a factor of about 2 or better. The only exception is counter mode without length hiding padding, where the reduction is still by a significant factor of about 1.5.
– Compression significantly reduces the number $|U|$ of uniquely identifiable web pages in all cases by a factor of 2 or better, and increases the size $|S_{\mathsf{max}}|$ of the largest set of pages of identical size.
– To analyze the impact of different length-hiding parameters and compression on communication complexity, we also measured the total amount of data transferred from the server to the client when each of the 1620 web pages is accessed exactly once. Note that compression not only reduces the adversary's advantage by a factor around 2, but also reduces the communication complexity by a factor around 2.

Table 1. Analysis of simple website fingerprinting attacks. "CTR" refers to AES-GCM, "BLK" to AES-CBC. "Comp." indicates whether compression is enabled, ℓ is the length-hiding parameter. Recall that plaintexts are padded to a multiple of ℓ *bytes* for counter mode and ℓ *blocks* for block mode encryption, thus, if $\ell = 1$, then no additional length-hiding padding is used. $|U|$ is the number of uniquely identifiable pages, $|S_{\mathsf{max}}|$ is the size of the largest set of pages of identical size, $|\mathcal{S}|$ is the number of different ciphertext lengths that are possible for the considered message distribution. "TrivAdv" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M_e})$. The column "Total data" lists the total amount of data transferred from the server to the client when each of the 1620 web pages is accessed exactly once, "Overhead" the traffic overhead incurred by LHE when compared to the same encryption mode with $\ell = 1$ and without compression.

	Mode	ℓ	Comp.	$\|U\|$	$\|S_{\mathsf{max}}\|$	$\|\mathcal{S}\|$	TrivAdv	Total data	Overhead
1.	CTR	1	False	868	4	1199	0.7400	5.39 MB	—
2.	CTR	1	True	400	8	822	0.5071	2.65 MB	-50.888 %
3.	CTR	80	False	8	85	54	0.0327	5.45 MB	1.158 %
4.	CTR	80	True	3	205	24	0.0142	2.71 MB	-49.745 %
5.	CTR	160	False	6	166	30	0.0179	5.52 MB	2.311 %
6.	CTR	160	True	3	404	14	0.0080	2.77 MB	-48.610 %
7.	CTR	400	False	1	409	14	0.0080	5.70 MB	5.677 %
8.	CTR	400	True	2	826	7	0.0037	2.96 MB	-45.159 %
9.	CTR	8995	False	0	1620	1	0.0000	13.93 MB	158.346 %
10.	CTR	3458	True	0	1620	1	0.0000	5.38 MB	-0.294 %
11.	BLK	1	False	40	23	213	0.1309	5.40 MB	—
12.	BLK	1	True	17	46	100	0.0611	2.66 MB	-50.777 %
13.	BLK	5	False	8	85	54	0.0327	5.45 MB	0.938 %
14.	BLK	5	True	3	205	24	0.0142	2.71 MB	-49.854 %
15.	BLK	10	False	6	166	30	0.0179	5.52 MB	2.089 %
16.	BLK	10	True	3	404	14	0.0080	2.77 MB	-48.722 %
17.	BLK	25	False	1	409	14	0.0080	5.70 MB	5.448 %
18.	BLK	25	True	2	826	7	0.0037	2.96 MB	-45.278 %
19.	BLK	563	False	0	1620	1	0.0000	13.95 MB	158.157 %
20.	BLK	217	True	0	1620	1	0.0000	5.40 MB	-0.111 %

- Without compression, even for a relatively large LHE parameter ℓ, such as $400 = 25 \cdot 16$ for counter mode or 25 for block mode encryption, the added communication complexity is relatively small, in particular when compared to the corresponding reduction in the adversary's trivial advantage.
- Achieving a trivial advantage of 0 costs a traffic overhead 159% without compression and even reduces traffic with compression.

We also consider a more powerful fingerprinting attack on the IACR ePrint archive website based on *user patterns* and another fingerprinting attack on the Google search engine. Due to space limitations, these two sections can be found in Appendices E and F.

4.2 Simple Wikipedia Fingerprinting

Adversarial model. We consider webpage fingerprinting attack on the Wikipedia website. In this model, a user visits the Wikipedia website. A passive adversary observes the encrypted traffic and tries to determine the visited page.

Considered algorithms. Same as in Section 4.1.

Empirical Data Generation. The Wikimedia Foundation publishes *pageview statistics* of the Wikipedia website since May 2015. The pageview statistics contain the number of requests for each webpage in Wikipedia, which provides us with a real world message distribution of Wikipedia pages. We used the pageview data of May 2021 from `https://dumps.wikimedia.org/other/pageviews/readme.html`. We considered the data for 164270 webpages written in *simple English* and used a script issuing an HTTP HEAD-request for every webpage to determine the size, in uncompressed and in compressed form. These values are stored in a database and used to compute the size of counter- and block-mode ciphertexts for these schemes that encrypt the web page in either compressed or uncompressed form, with or without length-hiding encryption, and with respect to different length-hiding parameters. We note that the Simple Wikipedia project has a total number of 260478 webpages, as of May 2021, but not all webpages were accessed at least once during this period. Therefore we consider only the subset of webpages that were accessed in the considered time period (thus, the message distribution implicitly assigns a probability of zero to other pages).

Let $\mathcal{M} := \{m_1, \ldots, m_{164270}\}$ be the set of different Simple Wikipedia webpages accessed in May 2021. We assume that the user picks one out of the 164270 webpages according to the distribution (denoted by $\mathsf{M}_{\mathsf{Wiki}}$) based on the pageview statistics and then gets the corresponding encrypted webpage. The adversary tries to guess the webpage. Hence, the adversary outputs an index $i \in \{1, \ldots, 164270\}$, such that we define function $\mathcal{P}$ as $\mathcal{P} : \mathcal{M} \to [1, 164270]$, for $m_i \mapsto i$.

Calculating the trivial advantage. Table 2 summarizes the results of the analysis. We calculate $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{Wiki}})$ for $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$, $\mathrm{C} \in \{\mathrm{True}, \mathrm{False}\}$ and different length-hiding parameter ℓ.

Conclusions. Table 2 allows us to make the following observations:

- The adversary achieves very high trivial advantage (0.875) when $\ell = 1$. This indicates that the adversary could successfully identify the webpage with high probability only from the ciphertext length.
- Compression reduces the overhead incurred by length-hiding encryption, and also reduces the trivial advantage.
- A length-hiding parameter of 40K in the CTR mode (resp. 2.5K in the BLK mode) reduces the trivial advantage to 0.01 and increases the total data by about 23 % without compression, but reduces about 50 % total data with compression.

We believe that the example of Wikipedia webpage fingerprinting illustrates very well that length-hiding encryption with proper length-hiding parameter together with compression provides us with both security and bandwidth reduction for real-world message distributions.

Beyond the above analyses, we have also conducted to more analyses on DNS fingerprinting (one using random host names from the *Majestic Million* list, the other using real-world DNS data collected in cooperation with a medium-sized university), which can be found in Appendix G.

5 Implementation and Analysis

We have implemented our new approach as an Apache module, based on the most recent versions of an Apache HTTP server (Apache/2.4.38) and the OpenSSL library (version 1.1.1d) for Debian 10. The implementation is currently experimental, we plan to make a more stable version publicly available.

The implementation consists of two components. The first component monitors the server's requests and responses. It stores the block-length and and number of occurrences of requests to static URLs and the corresponding responses in a database. This database then provides an approximation of the server's message distribution, which will be used by the second component to determine an appropriate length-hiding parameter ℓ. Furthermore, this component applies length-hiding padding to plaintext messages. Since OpenSSL currently does not support length-hiding padding beyond the last fragment of a plaintext, even though this is possible according to the TLS 1.3 standard, our experimental implementation currently simulates padding beyond frame bounds by appending NULL-bytes to the plaintext.[5]

The second component of our implementation computes the length-hiding parameter ℓ, based on the request-response database and a given upper bound on the desired trivial advantage. It computes the smallest ℓ that satisfies this trivial advantage for the message distribution defined by the database. Currently, the computation proceeds iteratively, that is, at first it tests whether the parameter $\ell = 1$ satisfies the trivial advantage bound. If it does not, ℓ is incremented and the process repeated until an optimal value ℓ is found. Since the trivial advantage does not necessarily decrease monotonously for a given message distributions, this approach yields the smallest possible value ℓ. The resulting parameter ℓ is then used to define the size of length-hiding padding in the first component. Note that the second component can run in the background at predefined times (e.g., once per hour or once per day).

5.1 Validation of Pencil-and-Paper Analysis

To validate the theoretical pencil-and-paper analysis from Section 4 we configured the webserver as a proxy for the IACR ePrint server (resp. Google search

[5] As we will discuss below, we aim to extend OpenSSL to allow for length-hiding padding beyond the TLS fragment boundary, by appending further TLS fragments, if necessary.

engine). Then we requested each of the webpages (resp. search terms) once, to establish the same message distribution that was considered in Section 4. We used our implementation to record the message distribution, determine the LHE parameter ℓ, and then again accessed every web page once. We considered both AES-GCM and AES-CBC cipher suites. In all cases, we were able to accurately reproduce our results from Tables 1–4.

So far we have considered only the case without compression, for the following reason. The plaintext data is accessible from within an Apache module by accessing an appropriate "data bucket" of the Apache processing filter chain. This enabled us to circumvent the fact that OpenSSL currently does not support length-hiding padding beyond the last fragment of a plaintext, by padding the plaintext directly with NULL bytes. However, in order to apply the same approach to *compressed* plaintext data, we would need access to a bucket that contains the plaintext *after* compression, but *before* encryption. This seems not possible from an Apache module, and therefore we could not yet validate the pencil-and-paper analysis of compress-then-encrypt.

In future work we aim to perform further experiments, on a public web server with real-world message distributions and with compression. To this end, we have to extend OpenSSL to allow for length-hiding padding beyond the TLS fragment boundary, by appending further TLS fragments, if necessary. Such low-level modifications to OpenSSL and experiments on a public web server in production require significant additional care. Therefore we view this as out of scope of the present paper and leave this for future work.

6 Conclusions

Fingerprinting attacks are a very powerful technique to extract information from an encrypted channel, which are known to be very difficult to defend against. We have introduced a methodology that makes it possible to concretely quantify the security gained by length-hiding encryption. Our goal is to go beyond the assumption that the length of a ciphertext does reveal sensitive information, which is typically made implicitly in theoretical security models. In this sense, understanding security in presence of messages of different lengths contributes to our understanding of "secure encryption" in general.

We find that a surprisingly small amount of padding that incurs only a minor bandwidth overhead of a few percent can already reduce the advantage of an adversary very significantly. Even only the use of *block mode* encryption schemes, whose padding provides some very simple form of length-hiding encryption, may already reduce the concrete advantage of fingerprinting attacks very significantly. We also observe that the the general recommendation to disable compression before encryption in all applications seems overgeneralized. While useful and necessary in some applications (when active attacks are feasible), in some other cases (e.g., "big brother" attacks) it may be harmful to security. It is therefore necessary to consider the security requirements of the application at hand with a more detailed analysis.

References

1. Majestic Million List, 2020. January 28, 2020, `https://majestic.com/reports/majestic-million`.
2. Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Protecting encrypted cookies from compression side-channel attacks. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 86–106. Springer, Heidelberg, January 2015.
3. Tal Be'ery and Amichai Shulman. A perfect CRIME? only TIME will tell, 2013. `https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf`.
4. Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997.
5. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.
6. Mike Belshe, Roberto Peon, and Martin Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, May 2015.
7. George Dean Bissias, Marc Liberatore, David D. Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted HTTP streams. In George Danezis and David M. Martin Jr., editors, *PET 2005*, volume 3856 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2005.
8. Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. Security of symmetric encryption in the presence of ciphertext fragmentation. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 682–699. Springer, Heidelberg, April 2012.
9. Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 227–238. ACM Press, November 2014.
10. Jean Paul Degabriele. Hiding the lengths of encrypted messages via Gaussian padding, 2021. To appear at ACM CCS 2021, preliminary copy directly obtained from the author.
11. Thai Duong and Juliano Rizzo. The CRIME attack, 2012. `http://www.ekoparty.org/archive/2012/CRIME_ekoparty2012.pdf`.
12. Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346. IEEE Computer Society Press, May 2012.
13. Daniel Kahn Gillmor. Empirical DNS padding policy, 2017. `https://dns.cmrg.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf`.
14. Xun Gong, Nikita Borisov, Negar Kiyavash, and Nabil Schear. Website detection using remote traffic analysis. In Simone Fischer-Hübner and Matthew K. Wright, editors, *PETS 2012*, volume 7384 of *LNCS*, pages 58–78. Springer, Heidelberg, July 2012.
15. Shay Gueron and Yehuda Lindell. GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 109–119. ACM Press, October 2015.

16. Shay Gueron and Yehuda Lindell. Simpleenc and simpleencsmall – an authenticated encryption mode for the lightweight setting. Cryptology ePrint Archive, Report 2019/712, 2019. `https://eprint.iacr.org/2019/712`.
17. Benjamin Harsha, Robert Morton, Jeremiah Blocki, John Springer, and Melissa Dark. Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage, 2020.
18. Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1187–1203. USENIX Association, August 2016.
19. Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *CCSW*, pages 31–42. ACM, 2009.
20. Andrew Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul F. Syverson, editors, *PET 2002*, volume 2482 of *LNCS*, pages 171–178. Springer, Heidelberg, April 2002.
21. Scott Hollenbeck. Transport Layer Security Protocol Compression Methods. RFC 3749, May 2004.
22. Tibor Jager, Martijn Stam, Ryan Stanley-Oakes, and Bogdan Warinschi. Multi-key authenticated encryption with corruptions: Reductions are lossy. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 409–441. Springer, Heidelberg, November 2017.
23. Jonathan Katz and Moti Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology*, 19(1):67–95, January 2006.
24. James Kelley and Roberto Tamassia. Secure compression: Theory & practice. Cryptology ePrint Archive, Report 2014/113, 2014. `http://eprint.iacr.org/2014/113`.
25. John Kelsey. Compression and information leakage of plaintext. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 263–276. Springer, Heidelberg, February 2002.
26. Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. Lost traffic encryption: fingerprinting LTE/4G traffic on layer two. In *WiSec*, pages 249–260. ACM, 2019.
27. Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. Coming of age: A longitudinal study of tls deployment. In *Proceedings of the Internet Measurement Conference 2018*, IMC 2018, New York, NY, USA, 2018. Association for Computing Machinery.
28. Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 255–263. ACM Press, October / November 2006.
29. Alexander Mayrhofer. Padding Policies for Extension Mechanisms for DNS (EDNS(0)). RFC 8467, October 2018.
30. Brad Miller, Ling Huang, Anthony D. Joseph, and J. D. Tygar. I know why you went to the clinic: Risks and realization of HTTPS traffic analysis. In Emiliano De Cristofaro and Steven J. Murdoch, editors, *PETS 2014*, volume 8555 of *LNCS*, pages 143–163. Springer, Heidelberg, July 2014.
31. Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *WPES*, pages 103–114. ACM, 2011.

32. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 372–389. Springer, Heidelberg, December 2011.

33. Angelo Prado, Neal, and Harris Yoel Gluck. SSL, gone in 30 seconds: a BREACH beyond CRIME, 2013. `https://media.blackhat.com/us-13/US-13-Prado-SSL-Gone-in-30-seconds-A-BREACH-beyond-CRIME-WP.pdf`.

34. Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

35. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

36. Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Tech. Rep*, 1998.

37. Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. Robust website fingerprinting through the cache occupancy channel. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 639–656. USENIX Association, August 2019.

38. Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1928–1943. ACM Press, October 2018.

39. Cihangir Tezcan and Serge Vaudenay. On hiding a plaintext length by preencryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 345–358. Springer, Heidelberg, June 2011.

40. Mathy Vanhoef and Tom Van Goethem. HEIST: HTTP encrypted information can be stolen through TCP-windows, 2016. `https://tom.vg/papers/heist_blackhat2016.pdf`.

41. Tao Wang and Ian Goldberg. On realistically attacking tor with website fingerprinting. *PoPETs*, 2016(4):21–36, October 2016.

42. Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017*, pages 1375–1390. USENIX Association, August 2017.

43. Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *2008 IEEE Symposium on Security and Privacy*, pages 35–49. IEEE Computer Society Press, May 2008.

44. Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson. Language identification of encrypted VoIP traffic: Alejandra y roberto or alice and bob? In Niels Provos, editor, *USENIX Security 2007*. USENIX Association, August 2007.

45. Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS 2009*. The Internet Society, February 2009.

A Side-channels Beyond Ciphertexts Lengths

Many works on the feasibility of fingerprinting attacks make use of side-channels beyond message lengths. For example, Dyer *et al.* [12] demonstrate that efficient countermeasures against traffic analyses often fail in practice. They analyzed datasets obtained from a concrete implementation, which contain additional information about the visited websites, beyond the size of transmitted data. This includes, for instance, the pattern of packet bursts and the bandwidth of the server.

In comparison, our work follows a "bottom-up" approach, in the sense that we aim to determine what is in principle achievable, if one focuses on the length of encrypted messages for different padding lengths, but ignoring other features such as packet bursts that could be dealt with separately. This leaves us with a simpler setting, which focuses only on application-layer message lengths and eliminates other side channels that exist in current implementations.

It seems natural that our approach yields better security, and also suggests that fingerprinting attacks such as those considered in [12] exploit not merely the length of padded messages, but also very significantly other features such as the timing of message bursts, which could be dealt with separately, independent of length-hiding padding. We believe that this points to a possible approach to make fingerprinting attacks less effective, in particular when the "TLS setting" is considered:

1. Use length-hiding padding to conceal the size of application-layer messages. Our approach even yields concrete information-theoretic bounds that apply to any attack (as long as the attack is only based on ciphertext length, but does not break the security of the encryption scheme, of course).
2. Eliminate other side channels, such as message bursts and their timing, by sender-side buffering and ensuring that application-layer messages are send in single bursts.

Given the effectiveness of fingerprinting attacks pointed out by [12] and other works, and with the theoretical impossibility result by Vaudenay and Tezcan [39] in mind, which both do not raise too much optimism about LHE, we consider this work as a first possible step towards a meaningful application of LHE in practice.

B On k-Anonymity of Encrypted Messages

Another natural candidate to analyze the indistinguishability of encrypted messages would be to extend the notion of k-anonymity [36], which is a classical and well-studied approach for protecting privacy in datasets in general. Let $\mathcal{M}$ be the set of all messages that can be produced by message sampler M. In our context, k-anonymity would essentially require that for every message $m \in \mathcal{M}$ there exists at least $k-1$ further messages $m_1, \ldots, m_{k-1}$ that are encrypted to the same ciphertext length.

However, there are two main issues with this seemingly more natural approach. k-anonymity provides only meaningful message privacy if the message sampler M produces the uniform distribution over $\mathcal{M}$ (or is at least close to uniform. However, k-anonymity may fail completely to provide expected security guarantees, if M is not the uniform distribution.

For instance, consider the message space

$$\mathcal{M} = \{m_{1,0}, m_{1,1}, m_{2,0}, m_{2,1}\}$$

and assume that message $m_{i,j}$ yields a ciphertext of lengths ℓ_i, such that

$$|\mathsf{E}_{\mathsf{sk}}(m_{1,0})| = |\mathsf{E}_{\mathsf{sk}}(m_{1,1})| = \ell_1$$

and

$$|\mathsf{E}_{\mathsf{sk}}(m_{2,0})| = |\mathsf{E}_{\mathsf{sk}}(m_{2,1})| = \ell_2$$

with $\ell_1 \neq \ell_2$. This message space would achieve 2-anonymity.

However, now consider a message distribution M which does not produce a uniform distribution over $\mathcal{M}$, but we have

$$\Pr\left[m_{1,0} \xleftarrow{\$} \mathsf{M}\right] = \Pr\left[m_{2,0} \xleftarrow{\$} \mathsf{M}\right] = 0$$

and

$$\Pr\left[m_{1,1} \xleftarrow{\$} \mathsf{M}\right] = \Pr\left[m_{2,1} \xleftarrow{\$} \mathsf{M}\right] = 1/2.$$

Note that in this case the adversary can *uniquely* determine the encrypted message from the ciphertext length, even though 2-anonymity is achieved.

The message sampler M represents the *a priori* knowledge of an adversary of encrypted data. For instance, the adversary might already know that the targeted user visits a Wikipedia web page about some disease, where some are more likely than others, but might be uncertain which one. In this case, the distribution M would produce only the Wikipedia pages on diseases, each with a probability corresponding to the likelihood of this particular disease, rather than the uniform distribution over all Wikipedia pages. As another example, consider an adversary that wants to figure out which exact research paper was accessed by a user on `https://eprint.iacr.org/2020/`, knowing that the user is particularly interested in research on one particular research area. Again, the uniform distribution over all messages is not useful to capture this.

We believe a practically useful formal security model for length-hiding encryption should cover such *a priori* knowledge and therefore consider k-anonymity as not very suitable to analyze practical fingerprinting attacks, unless the distribution M is the uniform distribution. In this case, for a message space achieving k-anonymity, we obtain a trivial advantage of at most $1/k$, and exactly $1/k$ if and only if each ciphertext length corresponds to *exactly* k different messages.

In our experiments described below, we will also determine the k of k-anonymity achieved by different padding lengths, for comparison.

C On one-way security and indistinguishability.

At a first glance our security definition (Definition 5) may appear very weak, since it considers a "one-way" security definition where the adversary has to output $\mathcal{P}(\boldsymbol{m}^*)$ explicitly, rather than distinguish it from random. We point out that the definition is actually *not* weaker than standard IND-M-CCA security based on message-indistinguishability in the sense of Definition 3, but actually a generalization of this standard notion. To see this, let M be the distribution which outputs some fixed message m_0 with probability 1/2, or otherwise a random message m_1, and let $\mathcal{P}$ be the function such that $\mathcal{P}(m_0) = 0$ and $\mathcal{P}(m_1) = 1$ for all $m_1 \neq m_0$. Note that then the $(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)$-$\mathsf{CCA}$ is essentially the standard IND-M-CCA security experiment for symmetric encryption in the sense of Definition 3. The generalization makes it possible to capture applications where messages do not have identical lengths in a more accurate and natural way.

D Proof of Theorem 3.8

We first define experiment Game_u for $u \in \{0, 1, \ldots, d\}$ as shown in Figure 3.

```
Proc. INITIALIZE($1^k$):
$\mathsf{sk}_1, \ldots, \mathsf{sk}_d \xleftarrow{\$} \{0,1\}^k$
$\mathsf{challenged} \leftarrow$ False
$\mathcal{C} \leftarrow \emptyset$
Return $(\mathsf{M}, \mathcal{P})$

Proc. ENC($i, m$):
Return $\mathsf{E}_{\mathsf{sk}_i}(m)$

Proc. DEC($i, \mathsf{ct}$):
If $(i, \mathsf{ct}) \in \mathcal{C}$:
    Return $\perp$
Return $\mathsf{D}_{\mathsf{sk}_i}(\mathsf{ct})$
```

```
Proc. CHALLENGE():
If $\mathsf{challenged} =$ True return $\perp$
$\mathsf{challenged} \leftarrow$ True
$\underbrace{((m^*_{1j})_{j \in [t_1]}, \ldots, (m^*_{dj})_{j \in [t_d]})}_{\boldsymbol{m}^*} \xleftarrow{\$} \mathsf{M}$
For $i \in [d], j \in [t_i]$:
    $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_{\mathsf{sk}_i}(m^*_{ij})$
    $r \xleftarrow{\$} \{0,1\}^{|\mathsf{ct}|}$
    $\mathsf{ct}^*_{ij} \leftarrow \begin{cases} \mathsf{ct} & \text{If } i > u \\ r & \text{Otherwise} \end{cases}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(i, \mathsf{ct}^*_{ij})\}$
Return $(\mathsf{ct}^*_{ij})_{i \in [d], j \in [t_i]}$

Proc. FINALIZE($p$):
Output $(\mathcal{P}(\boldsymbol{m}^*) = p)$
```

Fig. 3. Game_u for $u \in \{0, 1, \ldots, d\}$ in proving Theorem 8.

It is clear that Game_0 is exactly the same with the $(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d)$-$\mathsf{CCA}$ game. So that we get

$$\Pr[\mathsf{Game}_0^{\mathcal{A}} = 1] = \mathsf{RealSucc}(\mathsf{SE}, \mathsf{M}, \mathcal{P}, d, \mathcal{A}). \tag{1}$$

For any $u \in [d]$, Game_{u-1} differs from Game_u only in the generation of $(\mathsf{ct}^*_{uj})_{j \in [t_u]}$. And we can prove the following claim which bounds their difference.

Claim. For any $u \in \{1, \ldots, d\}$, there exists an adversary $\mathcal{B}$ such that

$$\Pr[\mathsf{Game}_{u-1}^{\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{u}^{\mathcal{A}} = 1] \leq \mathsf{Adv}_{\mathsf{IND\text{-}C\text{-}CCA}}^{\mathsf{SE},\mathcal{B}}(1^k) \tag{2}$$

and the running time of $\mathcal{B}$ is $T(\mathcal{B}) \approx T(\mathcal{A})+T(\mathsf{M})+T(\mathcal{P})+(t+q_e)\cdot T(\mathsf{E})+q_d\cdot T(\mathsf{D})$, where $T(\mathsf{M})$ is the time to sample messages from distribution M, $T(\mathcal{P})$ is the time to evaluate $\mathcal{P}$, $T(\mathsf{E})$ is the time to execute the encryption algorithm once, $T(\mathsf{D})$ is the time to execute the decryption algorithm once, q_e is the number of Enc queries made by $\mathcal{A}$ and q_d is the number of Dec queries made by $\mathcal{A}$.

Adversary $\mathcal{B}$ picks $d-1$ secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{u-1}, \mathsf{sk}_{u+1}, \ldots, \mathsf{sk}_d \xleftarrow{\$} \{0,1\}^k$ and implicitly sets sk_u as the secret key sk that is chosen by the $\mathsf{IND\text{-}C\text{-}CCA}$ experiment. Then it starts $\mathcal{A}$ with input $(\mathsf{M}, \mathcal{P})$.

In order to respond to the first Challenge query of $\mathcal{A}$, $\mathcal{B}$ samples a message tuple $\boldsymbol{m}^* = \Big((m_{1j}^*)_{j\in[t_1]}, \ldots, (m_{dj}^*)_{j\in[t_d]}\Big) \xleftarrow{\$} \mathsf{M}$. For all $i \neq u$, $\mathcal{B}$ is able to perfectly generate ct_{ij}^* for $j \in [t_i]$ according to the specification of Game_u since $\mathcal{B}$ has the secret key sk_i. To generate ct_{uj}^* for $j \in [t_u]$, $\mathcal{B}$ queries m_{uj}^* to the Challenge oracle provided by the $\mathsf{IND\text{-}C\text{-}CCA}$ experiment and sets the return value as ct_{uj}^*. Then $\mathcal{B}$ returns $(\mathsf{ct}_{ij}^*)_{i\in[d],j\in[t_i]}$ to $\mathcal{A}$.

When $\mathcal{A}$ makes an Enc(i, m) or Dec(i, ct) query for $i \neq u$, $\mathcal{B}$ is able to answer it perfectly since it has the secret key sk_i. If $i = u$, then $\mathcal{B}$ is able to forward this query to the oracles provided by the $\mathsf{IND\text{-}C\text{-}CCA}$ experiment and forward the result to $\mathcal{A}$.

When $\mathcal{A}$ outputs p and terminates, $\mathcal{B}$ calculates $p^* \leftarrow \mathcal{P}(\boldsymbol{m}^*)$ itself and outputs $b' \leftarrow 1$ to the $\mathsf{IND\text{-}C\text{-}CCA}$ experiment if and only if $p = p^*$.

So, when $b = 0$ in the $\mathsf{IND\text{-}C\text{-}CCA}$ experiment, the ciphertexts $(\mathsf{ct}_{uj}^*)_{j\in[t_u]}$ are honestly generated and $\mathcal{B}$ perfectly simulates Game_{u-1} for $\mathcal{A}$. When $b = 1$, the ciphertexts $(\mathsf{ct}_{uj}^*)_{j\in[t_u]}$ are just random strings and $\mathcal{B}$ perfectly simulates Game_u for $\mathcal{A}$. So,

$$\begin{aligned}\Pr[\mathsf{Game}_{u-1}^{\mathcal{A}} = 1] -& \Pr[\mathsf{Game}_{u}^{\mathcal{A}} = 1]\\ &= \Pr[b' = b \mid b = 0] - \Pr[b' = b \mid b = 1]\\ &\leq \mathsf{Adv}_{\mathsf{IND\text{-}C\text{-}CCA}}^{\mathsf{SE},\mathcal{B}}(1^k),\end{aligned}$$

and the claim follows. Finally, we can argue information-theoretically that

$$\Pr[\mathsf{Game}_{d}^{\mathcal{A}} = 1] \leq \mathsf{TrivSucc}(\mathsf{SE}, \mathsf{M}, \mathcal{P}). \tag{3}$$

Note that $(\mathsf{M}, \mathcal{P}, (|\mathsf{ct}_{ij}^*|)_{i\in[d],j\in[t_i]})$ is all the information about $\mathcal{P}(\boldsymbol{m}^*)$ leaked to $\mathcal{A}$ in Game_d and inequality Equation (3) follows. In total, Theorem 8 follows from Equations (1) to (3).

E Website Fingerprinting with Communication Pattern

Adversarial model. Again, we consider a website fingerprinting attack against the IACR ePrint archive at `https://eprint.iacr.org/2020/`, based on the size of encrypted data. However, we consider a slightly more complex communication pattern:

1. Alice picks one out of the 1620 subpages uniformly at random.
2. Then she visits the corresponding subpage.
3. Then she additionally downloads the pdf file of the corresponding research paper.

Hence, the message distribution $\mathsf{M}_{\mathsf{e,f}}$ considered here is the uniform distribution over the 1620 tuples (page, pdf file) in the ePrint archive for 2020. The adversary observes two ciphertexts $\mathsf{ct}_{\mathsf{page}} \xleftarrow{\$} \mathsf{E}^{(\ell)}_{\mathsf{sk}_1}(\mathsf{page})$ and $\mathsf{ct}_{\mathsf{file}} \xleftarrow{\$} \mathsf{E}^{(\ell)}_{\mathsf{sk}_2}(\mathsf{file})$. Its goal is again to determine the accessed web page, based on the size of ciphertexts transmitted from the server to the client.

Note that the adversary now observes both the size of the subpage and the size of the pdf document. Hence, we expect that this yields a more distinguishable pattern. Our goal is to determine the effect length-hiding encryption (and optional compression) for this scenario. We expect that compression is less helpful, due to the higher variance of the size and compressibility of pdf documents, and that a larger length-hiding parameter is required in order to reduce the trivial advantage significantly.

Considered algorithms. We consider the same algorithms as in Section 4.1.

Empirical data generation As in Section 4.1, we implemented a script that downloads all subpages of `https://eprint.iacr.org/2020/` along with the corresponding pdf files. If no pdf exists, which may happen when the paper is withdrawn from the archive, then this yields a pdf file size of 0, which also yields a distinguishable pattern. Then we calculated the size of the page and the pdf file, each in uncompressed and compressed form. These values are stored in a database, from which we compute the size of counter- and block-mode ciphertexts for AES-GCM and AES-CBC, with and without compression, and with and without length-hiding encryption for different length hiding parameters.

Calculating the trivial advantage. Since $\mathsf{M}_{\mathsf{e,f}}$ is the uniform distribution over the 1620 tuples (page, pdf file) in the ePrint archive for 2020, we can calculate the trivial advantage as in Section 4.1.

Table 3 summarizes the results of our analysis. Just like before, we calculate the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{e,f}})$ for $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$, with and without compression, and for different length-hiding parameter ℓ. Since the considered pdf documents are usually significantly larger than the web pages considered in Section 4.1, we have used larger length-hiding parameters. Counter mode ciphertexts are padded to a multiple of $\ell \in \{1, 1600, 16000, 160000\}$ bytes, block mode ciphertexts to a multiple of $\ell \in \{1, 100, 1000, 10000\}$ blocks (with block size 16 byte).

Conclusions. For the message distribution considered in this section, we come to the following conclusions:

– Without length-hiding encryption, in counter mode *all* web site accesses are uniquely identifiable with perfect success probability. This holds even when

compression is enabled. Hence, encryption is ineffective to prevent website fingerprinting in this case.
– Compression reduces the trivial advantage in most cases (except for counter mode without LHE), but significantly less than in the example considered in Section 4.1. Hence, as expected, compression is particularly helpful if the transmitted plaintext data is easily compressible, such as text data, but less effective if other plaintexts are transmitted, such as pdf documents. The same holds probably for images, videos, music files, etc., too.
– A block mode of operation provides essentially the same indistinguishability of data as a counter mode, when relatively long plaintexts are encrypted.
– For both encryption modes, a large length hiding parameter, which pads ciphertexts to a multiple of 16 kB, reduces the trivial advantage from 1 to below 0.12, which we expect to make traffic analysis attacks much less practical. Even such a large length-hiding parameter incurs only a very small overhead, which is below 3% without compression.
– Huge amount of padding is needed to achieve zero trivial advantage.

Hence, the length-hiding padding incurs only a very small overhead, while achieving a significant reduction of the trivial advantage. Not surprisingly, if the plaintexts consist mostly of data that is not well compressible, then compression has no significant effect on the trivial advantage of a website fingerprinting adversary. However, in such applications a suitably chosen length-hiding parameter may significantly reduce the trivial advantage. Even the large length hiding parameters that we consider have a surprisingly small bandwidth overhead.

F Google Search Term Fingerprinting

Adversarial model. We consider the scenario that a user is searching some term in a search engine. A passive adversary observes the encrypted traffic and tries to figure out which search term the user is searching for.

Considered algorithms. We consider the same algorithms as in Section 4.1.

Empirical Data Generation. In order to obtain realistic data, we collected 503 search terms published by Google as the daily search trends at `https://trends.google.com/trends/` from April 06 to May 05, 2021. We use a script to search each term in the Google search engine and collected the size of the returned data, in uncompressed and in compressed form. These values are stored in a database and used to compute the size of counter- and block-mode ciphertexts for these schemes that encrypt the web page in either compressed or uncompressed form, with or without length-hiding encryption and with respect to different length hiding parameters.

Let $\mathcal{M} := \{m_1, \ldots, m_{503}\}$ be the set of the 503 different Google search result pages. We assume that the user picks one out of the 503 words uniformly at random and then gets the corresponding search result page. Hence, the message distribution $\mathsf{M}_{\mathsf{Google}}$ that we consider here is the uniform distribution over $\mathcal{M}$.

The adversary tries to guess the search term. Hence, the adversary outputs an index $i \in \{1, \ldots, 503\}$, such that we define function $\mathcal{P}$ as $\mathcal{P} : \mathcal{M} \to [1, 503]$, for $m_i \mapsto i$.

Calculating the trivial advantage. Table 4 summarizes the results of the analysis. We calculate $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{Google}})$ for $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$, $\mathrm{C} \in \{\mathrm{True}, \mathrm{False}\}$ and different length-hiding parameter ℓ.

Conclusions. Table 4 allows us to make the following observations:

– The adversary achieves very high trivial advantage (close to 1) when $\ell = 1$. This indicates that the adversary could successfully identify the search term with high probability only from the ciphertext length of the search result page.
– Compression reduces the overhead incurred by length-hiding encryption, and also reduces the trivial advantage.
– A length-hiding parameter of 160K in the CTR mode (resp. 10K in the BLK mode) reduces the trivial advantage to 0.01 and increases the total data by about 16 % without compression, but reduces about 50 % total data with compression.

We believe that the example of Google search term fingerprinting illustrates very well that the *general* advice to disable compression before encryption can be misleading and harmful to security. It is necessary to consider the concrete security requirements of the application at hand.

G DNS Fingerprinting

Adversarial model. Now we turn from web site fingerprinting to a different application. Recall that the *Domain Name System* (DNS) is an Internet service that converts between host names such as `eprint.iacr.org` and IP addresses. DNS is a public service and therefore public DNS records are public information. Still, the contents of a DNS query made by an Internet user should usually be considered private information, since a passive adversary observing these requests may obtain a very clear picture about the user's web browsing habits. Therefore there is currently a tendency towards encrypting DNS traffic with TLS.

A DNS query consists of a *request* from a client containing a host name and a matching *response* that contains this host's IP address and additional data. In this section, we consider an adversary performing a *DNS fingerprinting* attack:

1. Alice picks host name from a distribution of host names known to the adversary. For instance, this distribution may reflect some *a priori* knowledge of the attacker about commonly requested host names for a particular DNS service.
2. Then she makes an encrypted DNS request to a DNS server and receives an encrypted response.

We formalize this procedure as $(\mathsf{req}, \mathsf{res}) \xleftarrow{\$} \mathsf{M_{DNS}}$ where $\mathsf{M_{DNS}}$ is a distribution over the (request, response) tuples. Then the request and response are encrypted ($\mathsf{ct_{req}} \xleftarrow{\$} \mathsf{E}^{(\ell)}_{\mathsf{sk}_1}(\mathsf{req})$ and $\mathsf{ct_{res}} \xleftarrow{\$} \mathsf{E}^{(\ell)}_{\mathsf{sk}_2}(\mathsf{res})$). A *passive* adversary that monitors both the request and the response, records two ciphertexts ($\mathsf{ct_{req}}$,$\mathsf{ct_{res}}$), and then tries to guess the domain name for which Alice has requested an IP address.

Hence, our function $\mathcal{P}$ is defined as

$$\mathcal{P} : \mathcal{M} \to [1, N], \qquad \mathcal{P}(\mathsf{req}_i, \mathsf{res}_i) = i$$

where $\mathcal{M}$ is the set of all possible request/response pairs for the considered list of host names of length N, $(\mathsf{req}_i, \mathsf{res}_i)$ is any request/response pair for the i-the entry in the list, and $\mathsf{M_{DNS}}$ is a distribution over $\mathcal{M}$.

Considered algorithms. We consider the same algorithms as in Section 4.1.

Empirical Data Generation. For our empirical analysis, we collected two different data sets:

1. The first 1000 host names of the so-called *Majestic Million* list [1]. This is a public list of the 1,000,000 most popular (according to the creator of the list) web sites on the Internet.[6]
   In order to obtain realistic and publicly reproducible data, we used the Google DNS server available at the IP address `8.8.8.8`. We implemented a script making DNS requests for the first 1000 host names on the *Majestic Million* list and calculating the size of the request and the response, each both in uncompressed and in compressed form. These values are stored in a database, and used to compute the size of counter- and block-mode ciphertexts for these schemes that encrypt the web page in either encrypted or unencrypted form, with or without length-hiding encryption and with respect to different length hiding parameters. To determine the trivial advantage, we assumed that a user chooses the requested domain name uniformly at random, due to a lack of more precise information of a realistic distribution. We believe that this already draws a general picture of the impact of length-hiding padding on the security of DNS against fingerprinting attacks.
2. To address the fact that in the *Majestic Million* case we had to assume a uniform distibution, due to lack of more information about the frequency of requested host names, we also obtained a real-world distribution of DNS requests in collaboration with the IT department of a medium-sized university with 23k students and 3.5k staff members. We monitored the university DNS service for a 24 hr time interval in July 2021, and stored the list of requested host names and their frequency in order to obtain a real-world message distribution that makes it possible to determine the concrete security and appropriate padding sizes for this particular DNS service.

[6] The list is similar to the Alexa Top 1-M list, which is often used in academic research papers, but published under a Creative Commons license.

Ethical considerations and compliance with privacy regulation. In order to guarantee the protection of privacy of users and compliance with applicable data protection laws, the data collection was carried out under supervision of the responsible university data privacy officer. IP adresses and any other data, except for the requested host names, were automatically filtered out before storing a requested host name in a text file.

In both cases, we then determined the size of the request and the response, each both in uncompressed and in compressed form. These values are stored in a database, and used to compute the size of counter- and block-mode ciphertexts for these schemes that encrypt the web page in either encrypted or unencrypted form, with or without length-hiding encryption and with respect to different length hiding parameters.

Calculating the trivial advantage. Table 5 summarizes the results of the Majestic Million DNS analysis. We calculate $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{MajesticDNS}})$ for $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$, $\mathrm{C} \in \{\mathrm{True}, \mathrm{False}\}$, different length-hiding parameter ℓ and uniform message distribution $\mathsf{M}_{\mathsf{MajesticDNS}}$. Since DNS requests and responses are rather small, we use relatively small length-hiding parameters. Counter mode ciphertexts are padded to a multiple of $\ell \in \{1, 64, 128, 256\}$ *bytes*, block mode ciphertexts to a multiple of $\ell \in \{1, 4, 8, 16\}$ *blocks* (with block size 16 byte).

Table 6 summarize the results of the university DNS analysis. We calculate $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{UniversityDNS}})$ for $\mathrm{M} \in \{\mathrm{CTR}, \mathrm{BLK}\}$, $\mathrm{C} \in \{\mathrm{True}, \mathrm{False}\}$, different length-hiding parameters ℓ for DNS request/response and real-world DNS request distribution $\mathsf{M}_{\mathsf{UniversityDNS}}$. Similarly, we consider relatively small length-hiding parameter $\ell \in \{1, 64, 128, 512\}$ *bytes* for counter mode and $\ell \in \{1, 4, 8, 32\}$ *blocks* for block mode.

Achieving trivial advantage 0*.* For the *Majestic Million* dataset, achieving a trivial advantage of 0 requires an overhead of 216.509% in CTR mode with compression. For BLK mode, we obtain 188.626%.

For the university DNS dataset, achieving a trivial advantage of 0 incurs 495.914% overhead in CTR mode with compression. For BLK mode, we obtain 441.625%.

Conclusions. Table 5 and Table 6 allows us to make the following observations:

– Since DNS requests and responses are relatively short, block mode encrypted improves the indistinguishability of (request, response)-tuples significantly, even without compression and length-hiding encryption.
– Compression reduces the trivial advantage but does not always reduces the overhead incurred by length-hiding encryption. The reason is that DNS requests and response contains little redundancy because of the short length, which may lead to a length growth after compression.
– In both of the two analysis, a moderate length-hiding encryption parameter of 8 for block mode (resp. $128 = 8 \cdot 16$ for counter mode) in combination

with compression reduces the trivial advantage to below 0.01. For counter mode, this incurs a traffic overhead of at most 72%, for block mode at most 59%.

Table 2. Analysis of Simple Wikipedia article fingerprinting. "CTR" refers to AES-GCM, "BLK" to AES-CBC. "Comp." indicates whether compression is enabled, ℓ is the length-hiding parameter. Ciphertexts are padded to a multiple of ℓ bytes for counter mode, and ℓ blocks for block mode encryption, $\ell = 1$ means that no additional length-hiding padding is used. $|U|$ is the number of uniquely identifiable pages, $|S_{\mathsf{max}}|$ is the size of the largest set of pages of identical size, $|\mathcal{S}|$ is the number of different ciphertext lengths that are possible for the considered message distribution. "$\mathsf{TrivAdv}$" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{Wiki}})$. The column "Total data" contains the total amount of data transferred from the sever to the client when each of the 164270 web pages exactly once, "Overhead" the overhead incurred by LHE when compared to the same encryption mode with $\ell = 1$ and without compression.

	Mode	ℓ	Comp.	$\lvert U\rvert$	$\lvert S_{max}\rvert$	$\lvert\mathcal{S}\rvert$	$\mathsf{TrivAdv}$	Total data	Overhead
1.	CTR	1	False	26597	232	62524	0.8750	400.24 GB	0.000%
2.	CTR	1	True	12908	81757	25689	0.5635	72.93 GB	-81.780%
3.	CTR	400	False	293	2251	1081	0.2775	401.14 GB	0.226%
4.	CTR	400	True	132	81757	507	0.1557	73.82 GB	-81.556%
5.	CTR	4000	False	27	21876	193	0.1354	409.24 GB	2.248%
6.	CTR	4000	True	26	81757	100	0.0701	81.76 GB	-79.573%
7.	CTR	40000	False	4	75284	34	0.0499	493.69 GB	23.349%
8.	CTR	40000	True	6	158278	20	0.0058	199.47 GB	-50.162%
9.	CTR	160000	False	2	160609	11	0.0091	828.22 GB	106.930%
10.	CTR	160000	True	2	164063	7	0.0012	728.39 GB	81.987%
11.	CTR	2290976	False	0	164270	1	0.0000	10.14 TB	2493.687%
12.	CTR	1331408	True	0	164270	1	0.0000	5.89 TB	1407.331%
13.	BLK	1	False	3283	238	10540	0.6023	401.35 GB	0.000%
14.	BLK	1	True	1457	81757	4901	0.3763	74.40 GB	-81.462%
15.	BLK	25	False	293	2251	1081	0.2775	402.22 GB	0.217%
16.	BLK	25	True	132	81757	507	0.1557	75.26 GB	-81.250%
17.	BLK	250	False	27	21876	193	0.1354	410.31 GB	2.232%
18.	BLK	250	True	26	81757	100	0.0701	83.19 GB	-79.272%
19.	BLK	2500	False	4	75284	34	0.0499	494.67 GB	23.251%
20.	BLK	2500	True	6	158278	20	0.0058	200.75 GB	-49.983%
21.	BLK	10000	False	2	160609	11	0.0091	828.85 GB	106.514%
22.	BLK	10000	True	2	164063	7	0.0012	728.99 GB	81.632%
23.	BLK	143186	False	0	164270	1	0.0000	10.14 TB	2486.567%
24.	BLK	83213	True	0	164270	1	0.0000	5.89 TB	1403.497%

Table 3. Analysis of website fingerprinting with patterns. "CTR" refers to AES-GCM, "BLK" to AES-CBC. "Comp." indicates whether compression is enabled, ℓ is the length-hiding parameter. $|U|$ is the number of uniquely identifiable pages, $|S_{\mathsf{max}}|$ is the size of the largest set of pages of identical size, $|\mathcal{S}|$ is the number of different ciphertext lengths that are possible for the considered message distribution. "$\mathsf{TrivAdv}$" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{e,f}})$. The column "Total data" contains the total amount of data transferred from the sever to the client when each of the 1620 web pages and corresponding pdf documents is accessed exactly once, "Overhead" the overhead incurred by LHE when compared to the same encryption mode with $\ell = 1$ and without compression.

	Mode	ℓ	Comp.	$\|U\|$	$\|S_{\mathsf{max}}\|$	$\|\mathcal{S}\|$	$\mathsf{TrivAdv}$	Total data	Overhead
1.	CTR	1	False	1620	1	1620	1.0000	1.33 GB	—
2.	CTR	1	True	1620	1	1620	1.0000	1.24 GB	-6.840 %
3.	CTR	1600	False	632	12	989	0.6103	1.33 GB	0.186 %
4.	CTR	1600	True	538	12	923	0.5695	1.24 GB	-6.643 %
5.	CTR	16000	False	74	57	191	0.1174	1.36 GB	2.359 %
6.	CTR	16000	True	68	62	178	0.1093	1.27 GB	-4.289 %
7.	CTR	160000	False	13	387	45	0.0272	1.68 GB	26.927 %
8.	CTR	160000	True	12	397	43	0.0259	1.59 GB	20.201 %
9.	CTR	91923338	False	0	1620	1	0.0000	278 GB	20836 %
10.	CTR	89829769	True	0	1620	1	0.0000	272 GB	20359 %
11.	BLK	1	False	1618	2	1619	0.9994	1.33 GB	—
12.	BLK	1	True	1609	4	1613	0.9957	1.24 GB	-6.840 %
13.	BLK	100	False	632	12	989	0.6103	1.33 GB	0.184 %
14.	BLK	100	True	538	12	923	0.5695	1.24 GB	-6.645 %
15.	BLK	1000	False	74	57	191	0.1174	1.36 GB	2.357 %
16.	BLK	1000	True	68	62	178	0.1093	1.27 GB	-4.291 %
17.	BLK	10000	False	13	387	45	0.0272	1.68 GB	26.925 %
18.	BLK	10000	True	12	397	43	0.0259	1.59 GB	20.199 %
19.	BLK	5745209	False	0	1620	1	0.0000	278 GB	20835 %
20.	BLK	5614361	True	0	1620	1	0.0000	272 GB	20358 %

Table 4. Analysis of Google search term fingerprinting. "CTR" refers to AES-GCM, "BLK" to AES-CBC. "Comp." indicates whether compression is enabled, ℓ is the length-hiding parameter. Ciphertexts are padded to a multiple of ℓ bytes for counter mode, and ℓ blocks for block mode encryption, $\ell = 1$ means that no additional length-hiding padding is used. $|U|$ is the number of uniquely identifiable pages, $|S_{\mathsf{max}}|$ is the size of the largest set of pages of identical size, $|\mathcal{S}|$ is the number of different ciphertext lengths that are possible for the considered message distribution. "$\mathsf{TrivAdv}$" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{Google}})$. The column "Total data" contains the total amount of data transferred from the sever to the client when each of the 503 web pages exactly once, "Overhead" the overhead incurred by LHE when compared to the same encryption mode with $\ell = 1$ and without compression.

	Mode	ℓ	Comp.	$\lvert U\rvert$	$\lvert S_{\mathsf{max}}\rvert$	$\lvert\mathcal{S}\rvert$	$\mathsf{TrivAdv}$	Total	Overhead
1.	CTR	1	False	503	1	503	1.0000	214.08 MB	—
2.	CTR	1	True	501	2	502	0.9980	71.07 MB	-66.800 %
3.	CTR	16	False	485	2	494	0.9821	214.08 MB	0.002 %
4.	CTR	16	True	460	4	480	0.9542	71.08 MB	-66.799 %
5.	CTR	1600	False	80	10	188	0.3725	214.46 MB	0.176 %
6.	CTR	1600	True	28	17	99	0.1952	71.45 MB	-66.625 %
7.	CTR	16000	False	9	56	35	0.0677	217.94 MB	1.801 %
8.	CTR	16000	True	2	124	17	0.0319	75.08 MB	-64.927 %
9.	CTR	160000	False	1	361	6	0.0100	248.45 MB	16.054 %
10.	CTR	160000	True	2	363	4	0.0060	98.71 MB	-53.892 %
11.	CTR	1294396	False	0	503	1	0.0000	621.76 MB	190.435 %
12.	CTR	513311	True	0	503	1	0.0000	246.57 MB	15.178 %
13.	BLK	1	False	485	2	494	0.9821	214.08 MB	—
14.	BLK	1	True	460	4	480	0.9542	71.08 MB	-66.799 %
15.	BLK	100	False	80	10	188	0.3725	214.46 MB	0.175 %
16.	BLK	100	True	28	17	99	0.1952	71.45 MB	-66.626 %
17.	BLK	1000	False	9	56	35	0.0677	217.94 MB	1.799 %
18.	BLK	1000	True	2	124	17	0.0319	75.08 MB	-64.927 %
19.	BLK	10000	False	1	361	6	0.0100	248.45 MB	16.052 %
20.	BLK	10000	True	2	363	4	0.0060	98.71 MB	-53.893 %
21.	BLK	80900	False	0	503	1	0.0000	621.77 MB	190.432 %
22.	BLK	32082	True	0	503	1	0.0000	246.57 MB	15.176 %

Table 5. Analysis of DNS fingerprinting. "CTR" refers to AES-GCM, "BLK" to AES-CBC. "Comp." indicates whether compression is enabled, ℓ is the length-hiding parameter. Ciphertexts are padded to a multiple of ℓ bytes for counter mode, and ℓ blocks for block mode encryption, $\ell = 1$ means that no additional length-hiding padding is used. $|U|$ is the number of uniquely identifiable pages, $|S_{\mathsf{max}}|$ is the size of the largest set of pages of identical size, $|\mathcal{S}|$ is the number of different ciphertext lengths that are possible for the considered message distribution. "TrivAdv" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{MajesticDNS}})$. The column "Total data" contains the total amount of data transferred between the client and the DNS server when each of the 1000 websites is accessed exactly once, "Overhead" the overhead incurred by LHE when compared to the same encryption mode with $\ell = 1$ and without compression.

	Mode	ℓ	Comp.	$\|U\|$	$\|S_{\mathsf{max}}\|$	$\|\mathcal{S}\|$	TrivAdv	Total data	Overhead
1.	CTR	1	False	469	13	644	0.644	157.97 KB	—
2.	CTR	1	True	363	10	590	0.590	145.90 KB	-7.644 %
3.	CTR	64	False	1	536	6	0.005	222.88 KB	41.084 %
4.	CTR	64	True	1	827	5	0.004	197.38 KB	24.942 %
5.	CTR	128	False	0	549	3	0.002	309.62 KB	95.998 %
6.	CTR	128	True	0	836	2	0.001	270.50 KB	71.231 %
7.	CTR	256	False	0	974	2	0.001	506.50 KB	220.623 %
8.	CTR	256	True	0	1000	1	0.000	500.00 KB	216.509 %
9.	BLK	1	False	7	130	36	0.035	173.23 KB	—
10.	BLK	1	True	1	246	19	0.018	161.30 KB	-6.891 %
11.	BLK	4	False	0	535	6	0.005	223.44 KB	28.980 %
12.	BLK	4	True	1	814	5	0.004	198.69 KB	14.693 %
13.	BLK	8	False	0	546	3	0.002	310.00 KB	78.948 %
14.	BLK	8	True	0	819	2	0.001	272.62 KB	57.374 %
15.	BLK	16	False	0	974	2	0.001	506.50 KB	192.378 %
16.	BLK	16	True	0	1000	1	0.000	500.00 KB	188.626 %

Table 6. Analysis of DNS fingerprinting for real-world university DNS request distribution $\mathsf{M}_{\mathsf{UniversityDNS}}$. "TrivAdv" is the trivial advantage $\mathsf{TrivAdv}(\mathsf{SE}^{(\ell)}_{\mathrm{M,C}}, \mathsf{M}_{\mathsf{UniversityDNS}})$. The others stay the same with Table 5.

	Mode	ℓ	Comp.	$\lvert U\rvert$	$\lvert S_{\mathsf{max}}\rvert$	$\lvert\mathcal{S}\rvert$	TrivAdv	Total data	Overhead
1.	CTR	1	False	2734	18646	10295	0.551	73.49 MB	—
2.	CTR	1	True	1679	11865	8364	0.584	80.81 MB	9.956 %
3.	CTR	64	False	1	273098	16	0.024	96.35 MB	31.103 %
4.	CTR	64	True	2	190619	13	0.037	102.79 MB	39.866 %
5.	CTR	128	False	0	338153	5	0.009	124.24 MB	69.051 %
6.	CTR	128	True	0	311134	4	0.004	126.30 MB	71.850 %
7.	CTR	512	False	0	448467	1	0.000	437.96 MB	495.914 %
8.	CTR	512	True	0	448467	1	0.000	437.96 MB	495.914 %
9.	BLK	1	False	13	68962	147	0.146	80.86 MB	—
10.	BLK	1	True	6	91914	87	0.081	88.25 MB	9.145 %
11.	BLK	4	False	2	269943	18	0.024	96.68 MB	19.569 %
12.	BLK	4	True	1	174666	13	0.030	105.21 MB	30.109 %
13.	BLK	8	False	0	335348	6	0.009	124.64 MB	54.149 %
14.	BLK	8	True	0	293682	4	0.004	128.43 MB	58.831 %
15.	BLK	32	False	0	448465	2	0.000	437.96 MB	441.626 %
16.	BLK	32	True	0	448467	1	0.000	437.96 MB	441.625 %