Look What's There! Utilizing the Internet's Existing Data for Censorship Circumvention with OPPRESSION

Sebastian Zillien
szillien@hs-worms.de
Worms University of Applied Sciences
Worms, Germany

Tobias Schmidbauer
tobias.schmidbauer@th-nuernberg.de
Nuremberg Institute of Technology
Nuremberg, Germany

Mario Kubek
mkubek@gsu.edu
Georgia State University
Atlanta, GA, USA

Jörg Keller
joerg.keller@fernuni-hagen.de
FernUniversität in Hagen
Hagen, Germany

Steffen Wendzel
wendzel@hs-worms.de
Worms University of Applied Sciences
Worms, Germany
FernUniversität in Hagen
Hagen, Germany

ABSTRACT

An ongoing challenge in censorship circumvention is optimizing the stealthiness of communications, enabled by covert channels. Recently, a new variant called *history* covert channels has been proposed. Instead of modifying or mimicking legitimate data, such channels solely *point* to observed data matching secret information. This approach reduces the amount of secret data a sender explicitly must transfer and thus limits detectability. However, the only published history channel is only suitable for special scenarios due to severe limitations in terms of bandwidth. We propose a significant performance enhancement of history covert channels that allows their use in real-world scenarios through utilizing the content of online social media and online archives. Our approach, which we call OPPRESSION (*Open-knowledge Compression*), takes advantage of the massive amounts of textual data on the Internet that can be referenced by short pointer messages. Broadly, OPPRESSION can be considered a novel encoding strategy for censorship circumvention.

We further present and evaluate our open source proof-of-concept implementation of OPPRESSION that can transfer secret data by pointing to popular online media, such as Twitter (now "X"), news websites, Wikipedia entries, and online books. The pointer itself is transmitted through existing censorship circumvention systems. Our approach minimizes the amount of traffic to be concealed in comparison to existing works, even in comparison to compression.

CCS CONCEPTS

• **Security and privacy → Network security**; *Distributed systems security*; *Information flow control*; Pseudonymity, anonymity and untraceability; • **Social and professional topics → Computer crime**.

KEYWORDS

Steganography, Information Hiding, History Covert Channel, Censorship Circumvention, Encoding, Compression

ACM Reference Format:
Sebastian Zillien, Tobias Schmidbauer, Mario Kubek, Jörg Keller, and Steffen Wendzel. 2024. Look What's There! Utilizing the Internet's Existing Data for Censorship Circumvention with OPPRESSION. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3634737.3637676

1 INTRODUCTION

Censorship systems aim at restricting the access to online resources for citizens and face a growing trend. For several years, many countries applied censorship systems [14, 21, 42] with their methods becoming increasingly sophisticated [32]. So-called *censorship circumvention* systems enable the free communication of individuals despite the presence of censorship systems [25]. Censorship circumvention systems typically bypass censorship by mimicking legitimate behavior, tunneling traffic, providing proxies or utilizing steganography methods, including covert channels.

Addressed problem: A major challenge of any kind of censorship circumvention system is that the more information is transmitted, the more likely is its detection as an anomaly, leading to limitation and blocking. For this reason, we propose a novel censorship circumvention approach called OPPRESSION (*Open-knowledge Compression*). OPPRESSION is based on the utilization of existing censorship circumvention systems, but radically reduces the amount of transferred data using the history covert channel concept [55]. We do that by only sending short pointers to longer text chunks that are available via the Internet, thus combining ideas from encoding, compression and steganography.

The volume of messages that appear in online social media increases since years. According to 2022 statistics, there are for instance 575k new tweets sent on Twitter (now "X") and 66k new photo and video shares on Instagram per minute [24]. Further, Internet-based archives and databases, such as website archives, Wikipedia and digitized open books continue to grow. The English

Wikipedia encyclopedia, for example, currently contains around 6.5M articles. A large fraction of online content on current popular social media platforms as well as archives contains textual code. For this reason, we considered these massive content resources attractive for transferring secret texts through censors. As long as *at least one* legitimate textual resource of the public Internet can be accessed by a sender and a receiver, OPPRESSION can be applied.

If a public dataset was used once for OPPRESSION, it can still be used even in case a censor blocks the access to the particular resource after some time.

Our **key contributions** are as follows:

(1) Presentation of OPPRESSION (*Open-knowledge Compression*), a new censorship circumvention methodology which reduces the amount of traffic required for the transfer through systems drastically, thus minimizing the chances for detection and blocking of traffic by a censor.
(2) Detailed theoretical evaluation of our method
(3) Provision of an open source prototype implementation,
(4) Evaluation of effectiveness and robustness of the implementation with different Internet data sources such as Wikipedia, Twitter[1], ebooks, etc. OPPRESSION can be used with any available censorship circumvention tool.

The reminder of this paper is structured as follows. Sect. 2 discusses related work. Our approach and methodology are introduced in Sect. 3. In Sect. 4, we describe in detail the dictionary generation and document choices for the evaluation. We perform an evaluation of our approach regarding robustness, bandwidth and detectability in Sect. 5, in which we also discuss the limitations of our work. We conclude in Sect. 6 that also gives an outlook on future work.

2 RELATED WORK

Censorship Circumvention. A plethora of work on censorship circumvention was published during the last decades. Several authors disguised and/or multiplexed secret communications using either anonymization systems, such as *Tor*, traffic mimicking, or direct utilization of (network) steganography [29, 35]. After censorship resistance and circumvention became more prominent in the early 2000's [26], sophisticated methods and tools emerged. Popular early examples are *StegoTorus* [54], *SkypeMorph* [37], *ScrambleSuit* [83] and *FreeWave* [26].

Already in 2013, the imitation of legitimate protocols by available censorship circumvention tools was shown to be weak [20, 25], i.e., available imitations were detectable. Succeeding systematic analyses of censorship circumvention tools can be found in [29] and [40]. Newer and more sophisticated approaches aimed at improving the mimicry of existing protocols as well as the tunneling and embedding of steganographic messages in existing protocol traffic (including the replacement of traffic) [35], see, e.g., *WebRTC* [5], *Stegozoa* [14], *MassBrowser* [40] and *OUStralopithecus* [32] for recent examples. In [14], the authors investigate the throughput of secret messages and also evaluate the detectability compared to transmission efficiency. Therein, the detectability is significantly reduced by decreasing throughput. So for some censorship circumvention tools, a key criterion for maximizing stealthiness remains through the size of the transmitted message: the shorter the message, the smaller the influence on a carrier and thus less content to observe for a censor. For this reason, minimizing the amount of transmitted secret data as proposed by our work appears to be advantageous for censorship circumvention.

History Covert Channels and Covert Channel Message Size Amplification. The only method known by the authors that is based on secret data represented through *pointers* to existing (*historic*) online data is *DYST* (*Did You See That?*) [55]. DYST observes live network traffic and points a receiver to recently seen traffic data that matches a (piece of a) secret message. DYST introduces a methodological advancement in the sense that it minimizes the amount of covert traffic of a sender, i.e., it enables an amplification of the covert channel's message size. However, DYST has shown limitations in terms of practical applicability as waiting times for matching secret traffic are rather long. In comparison to DYST, our approach OPPRESSION utilizes already existing Internet-based archives with massive volumes of data accessible to both, sender and receiver. For this reason, we do not rely on finding randomized matches in live traffic. Instead, we match secret natural text with existing natural text fragments, which leads to significantly more matches and better performance. Further, an early method, HICCUPS corrupts frame checksums to refer to a packet's payload containing secret data [33]. Note that HICCUPS works in a just-in-time fashion and does not employ our idea of pointing to previously seen data. Further, HICCUPS places the secret data directly in the payload instead of referring to third-party data.

Cover Selection and Coverless Steganography. Two image steganography concepts (*cover selection* [16] and *coverless image steganography* [88]) rely on previously built databases of images that, when they match a secret message, are used for a transmission, or are partially used through image patches. One recent publication by Liu et al. [31] is somewhat similar to ours in the sense that it also builds a dictionary. However, the authors use image features for their directory and their methodology differs as it relies on a modification of the so-called *bag-of-words* method.

Linguistic Steganography. Current methods of linguistic steganography rely on the generation of new and modification of existing texts [1]. Often, text generation is conducted with the help of machine learning methods, see, e.g., RNN-Stega [86] and VAE-Stega [87] for recent examples. A work with some familiarity to ours is *Facade*, which utilizes web searches, where search terms are mapped to a secret dictionary of words [28]. In comparison to our work, Facade relies on the generation of its own cover traffic (instead of pointing to it or minimizing it), is bound to the *OpenSearch* protocol, and performs a standard text steganography approach where search terms are substituted with secret terms during interpretation.

Tries. A *trie* [11, 15] is an m-way tree to store a set of words over an alphabet of size m. Each node in the trie except the root represents one letter. The letters on the way from the root (marked with ε) to a node represent the beginning (prefix) of a word. The successor nodes represent the corresponding possibilities to continue the word, thus each node has outdegree at most m. Tries are particularly suitable as a data structure due to their compact data storage and because of the fast access to the information. A variant

[1]Twitter is now called X. The data used in this paper was collected in late-2022. The Twitter-API was limited after the tweets were fetched for this paper. However, the principle is still adoptable to other similar services like Mastodon, Threads or Hive and other social media platforms based on user-created textual content.

of these tries, called *Patricia tries*, was first mentioned in [38]. They reduce the number of nodes by merging each node that has a single child with this child node. A further variant is the so-called Compact Patricia Trie (CPT) used for classification purposes that only stores necessary subtrees, removes redundant subtrees, and prunes strings in leaves that are longer than 1 [84]. While tries concern the storage of a set of words, we consider storing multiple sequences of words. The idea of a trie can be extended by attributing each node with one word instead of one letter. As the set of possible words over an alphabet can be infinite for arbitrarily long words, there is no more bound on the outdegree of nodes. However, except for the root, the outdegree will be small in practice. For practical reasons, we will restrict the depth of the tree, i.e., only accept word sequences of length at most l. This trie variation can efficiently store and search for sequences of words and exhibits the mentioned space-optimization property, too. Furthermore, and depending on the application, it is possible to add metadata like a sentence number or document number to the nodes. Therefore, we resort to this variant of a trie.

Encoding and Compression. Encoding translates information into another format, for example binary numbers, in such a way the process is reversible. Coding theory distinguishes encoding into four types [23]: Source coding, error control, cryptographic coding, and line coding, whereby only source coding is relevant for the remainder of this paper. An example for source coding is character encoding (like i.a. ASCII and Unicode), where each single character is represented by one specific binary sequence, defined by a fixed table. Also, compression is considered to be source coding. A famous example is the lossless GZIP compression algorithm DEFLATE [12], which is a combination of LZ77 [89] and Huffman coding [27]. For DEFLATE, compression is performed by two steps: (1) Pointers to already encoded matching strings up to 32 kB (LZ77), and (2) utilizing weighted symbols according to the frequency of use (Huffman Coding). DEFLATE therefore crafts a self creating, text-specific codebook, containing characters and sequences of characters that are pointed to. The codebook need not be transferred and is self-created by the message during compression and decompression. However, still the self-creating codebook is implicitly part of the compressed message transmitted.

3 METHODOLOGY

In this section, first we introduce the threat scenario, followed by a general description of OPPRESSION. Second, we describe two different feasible realizations in detail, each comprising a high-level description and a detailed description of dictionary generation and encoding. Also, the encoding of absence words, not covered by the dictionary, is described. An overview of the notations used in this paper is given in Tab. 1.

3.1 Threat Scenario

In our threat scenario, two communication parties aim to exchange textual messages. The parties are separated through means of Internet censorship, in the sense that their communication is both observed (to determine illicit content) and influenced (e.g., limited or blocked) by a censor. We do not discriminate whether the censor has limited or even state-level capabilities. If the censor detects an illicit communication between the two parties, the censor will try to block their communication. In addition, one or both of the parties may be threatened or punished if the communication is detected.

In general, the more traffic the two parties exchange directly, the more traces they leave for potential analysis through the censor. Furthermore, the amount of information to be transmitted can be limited by the censor. Usually, censorship algorithms require some minimum amount of traffic containing illicit content to successfully detect censorship circumventing communications. For this reason, our work aims at minimizing the amount of traffic that the two parties need to exchange. To this end, both parties agree on web-based content in advance to generate a codebook as described in the next section. The web content must be accessible to both of them and can be any type of website—it can even be a fake news website of the censor itself to which both parties have access. It might also be a Wikipedia page, a Twitter account, or online-accessible books. We assume that the parties can access the web content at times *they* can choose, e.g., slowly downloading the content over a period of months so that it that would be considered unsuspicious by a censor. In our scenario, it is practically infeasible for a censor to ensure that the communication parties cannot access *any* common online resource that they can use to assemble the codebook. This has practical reasons as in most scenarios (e.g., [14, 83]) even censors are forced to keep the Internet connection somewhat functional so that citizens can use parts of it, at least due to economical constraints. We further assume that the pre-computation time required by the parties before the communication phase can start is not an obstacle, and that the censor has no direct access to the end-user's device.

3.2 The Approach in General

An overview of our circumvention method is given in Fig. 1. We assume that both parties, Alice and Bob, use some available circumvention tool to establish the circumvention channel from inside a censored network (where Alice resides) to the public Internet (where Bob resides). They use the circumvention channel to exchange the address of a publicly accessible website (1). Or, sender and receiver could agree on a predefined search query which results in a fixed set of articles or tweets. Further, sender and receiver may agree on the latest article of a website, like news articles or blog entries, at a predefined moment. Both parties download content from that website (2a)/(2b), i.e., the monitored flow 2a is a *legitimate* one. On the basis of the retrieved web content, they each generate a local codebook. The codebooks are consistent as both use the same deterministic algorithm for the generation. This access only needs to be performed once. Afterward, they start to exchange small pointers which point to longer strings in the codebook (3). This method minimizes the amount of transferred traffic and keeps both parties' communication stealthier in comparison to state-of-the-art attempts, as well as the utilization of compression algorithms. In contrast to LZ77, the approach is based upon a previously existing external codebook, and not on a codebook generated on the fly during compression and decompression.

To transfer the actual pointers, the communicating parties utilize third-party censorship circumvention software that can be exchanged at any time (e.g., once one tool becomes obsolete because it is blocked or detectable). We thus assume that at any given point in

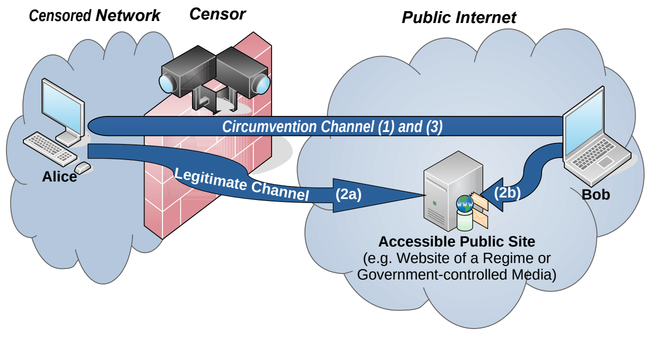

Figure 1: High-level Functioning of OPPRESSION

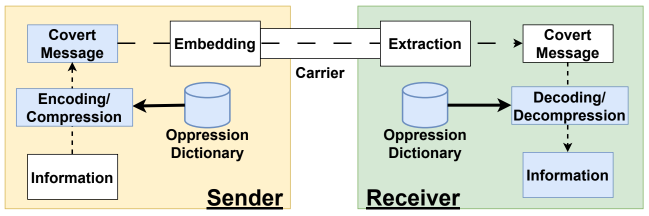

Figure 2: Contribution of OPPRESSION

time, at least a small fraction of information (the above-mentioned pointer) can be transferred through existing circumvention software without raising the attention of the censor. The contributions of OPPRESSION are depicted in Fig. 2 by blue boxes. On the side of the sender, OPPRESSION is utilized to create the covert message by encoding information. On the side of the receiver, the given covert message is decoded by OPPRESSION, extracting the transmitted information. The steps not covered by OPPRESSION, i.e., embedding and extraction of information in and of a carrier have to be performed by censorship circumvention software. For this reason, the utilized circumvention software itself is outside our scope as we focus on the steps performed by OPPRESSION.

3.3 Approach 1 - Document Pointers

3.3.1 High-level Description. Assume that sender and receiver have agreed on a set of documents $D_1, D_2, \ldots$. Each document D_i is seen as a sequence of words $w_{i,0}, w_{i,1}, \ldots$, where extra characters like commas, quotation marks, etc. are ignored for the moment.

By a *pointer*, we denote a tuple

$$P = (i, o, k) \tag{1}$$

with the meaning: utilize document D_i and look at the words from position w_o to w_{o+k-1}; the pointer represents the sequence of k words starting from a position at an offset o in the document i.

To transfer a secret message comprised of words $a_1, \ldots, a_n$, the sender tries to locate words $a_1, \ldots, a_{k_0}$ in the document D_{i_0} at positions o_0 to o_0+k_0-1, i.e. $w_{i_0,o_0+j} = a_{0+j}$ for $j = 0, \ldots, k_0-1$ so that k_0 is maximized. Then the sender tries to find words $a_{k_0+1}, \ldots, a_{k_0+k_1}$ in the document D_{i_1} at positions o_1 to $o_1 + k_1 - 1$, and so on until all words of the secret message are located. To ensure that any message can be sent, the sender can use an additional document D_{root} that contains all possible words. The sender might create such a document and transmit this once to the receiver, or it might use a document such as https://github.com/dwyl/english-words. In contrast to other documents, the words here are only considered isolated. Alternatively, sender and receiver could agree on a scheme to encode not defined words, like names or abbreviations for instance, as described in Section 3.3.4.

The secret message then consists of a sequence of pointers $(i_0, o_0, k_0), (i_1, o_1, k_1), \ldots$ such that $\sum_j k_j = n$. The advantage is that the pointers are compact, e.g., 4 bytes or fewer per pointer, and the words are large, e.g., 5 characters on average. A message comprised of 100 words of 5 characters each (plus 100 white spaces), i.e., 600 characters long, can theoretically be sent in 20 pointers or 60 bytes, which in numerous instances is better than usual lossless compression like GZIP. Thus, our approach might be seen as a variant of text compression applied to text steganography or alternatively, our approach might also be seen as a very efficient kind of encoding, that points on existing words.

The receiver, upon receiving a pointer (i_j, o_j, k_j), locates words w_{i_j,o_j} to w_{i_j,o_j+k_j-1} in the document D_{i_j}, which it accepts as the next part of the secret message.

To quickly locate message parts in known documents, the sender initially generates a kind of dictionary for lookup, which we will describe next.

3.3.2 Dictionary Generation. Let $W = \cup_{i,o}\{w_{i,o}\}$ be the set of all different words in the documents, and for each $w \in W$, let $O_w = \{(i, o) \mid w_{i,o} = w\}$ the set of all occurrences of w.

For each $w \in W$, and each $(i, o) \in O_w$, we denote by $s_{i,o} = w_{i,o}, \ldots, w_{i,o+l-1}$ the sentence of length l starting with the word $w = w_{i,o}$ in the document D_i at the position with offset o. If o is

Table 1: Notations used in this paper.

Symbol	Definition
a_i	Secret word i (that is to be transferred)
n	number of secret words to be transferred
D_i	Document i (D_{root} is the root document)
ϵ	The empty word
k	Length of a sequence that is pointed to (in number of words)
L	List of words
L_{simple}	Length of binary pointer encoding
l	Length of a sentence to be added to tree (equals tree depth)
m	Number of documents
O_w	Set of all occurrences of a word w in a document
o	Offset parameter for words in a document
P	Pointer $P = (i, o, k)$, i.e., points to the words from positions w_o to w_{o+k-1} within document D_i
S_w	Set of all sentences of a given length that start with word w
T	A trie
T_w	Trie variant over S_w
v_i	Vertex (node) i of a path in a Trie
W	Set of all words in a document
$w_{i,j}$	Word j of document D_i

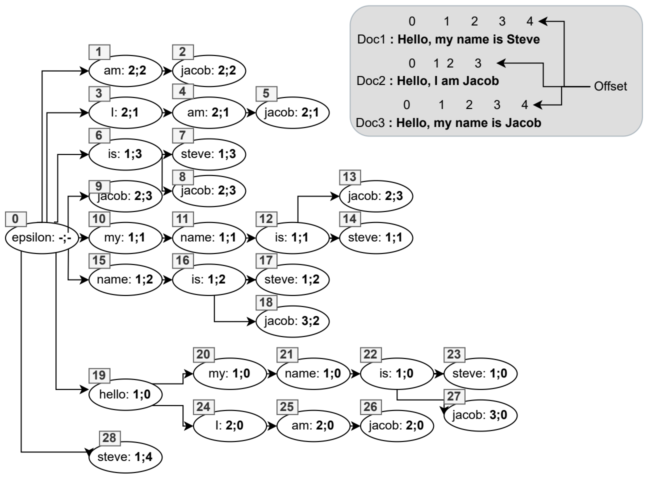

Figure 3: Document-pointer Tries

nearer than l to the end of the document, we shorten the sentence accordingly, but will not mention this in the following to simplify presentation. Then, $S_w = \{s_{i,o} \mid (i,o) \in O_w\}$ denotes the set of all sentences of length l that start with a word w and occur in one document D_i.

For each $w \in W$, we build a trie variant T_w over S_w, i.e., a prefix tree over all sentences that start with w. Each leaf of the tree, which corresponds to the end of a sentence $s_{i,o}$, is annotated with (i,o), i.e., the occurrence of that sentence. If there are multiple occurrences of the same sentence, we choose the first that is analyzed. We are aware, that one may also be picked randomly, but we decided not to implement this randomized approach for our proof of concept.

We might also view the trees T_w as a single tree by adding an artificial root node for the empty word ε that may "occur" in any document at any offset, and having the roots of all trees T_w as children of that new root node. The tree generation thus happens as shown in Alg. 1 for the next approach. Figure 3 depicts 3 documents and the trees constructed from them. The pointers allocated at each node point at the first root-node appearance and the sentence to be found in the specific document. The word *Hello* has three leaves, pointing at the word appearance of the root node.

If we are going to send a secret message comprised of a sequence of words $a_1, \dots, a_n$, then we search for the maximum k_0 such that $a_1, \dots, a_{k_0}$ is present in the tree T_{a_0}. If $k_0 < l$, then we do a depth first search in the tree, starting at a_{k_0}, to find a leaf annotation (i_0, o_0). The first pointer to be sent is (i_0, o_0, k_0). We repeat this with $a_{k_0+1}, \dots, a_n$ to find further pointers, until the whole message is covered. So, the example sentence *Hello, I am Steve* (not represented in the original documents) can be encoded as follows: $(1,0,3); (0,4,1)$, which means, take the words of document 1 from position 0 to position $2 = 0 + 3 - 1$ and the word at position 4 from document 0. It also demonstrates that the coding is not unique. As we always search for the maximum k_j, we use a greedy strategy.

The form of the trees and a suitable depth l depend both on the documents to be chosen and on the messages to be sent. Furthermore, the choice of the documents may depend on the messages to be sent. If the messages cannot be predicted, it might be useful to choose a set of documents such that a broad range of sentences is covered. If a specific set of messages is covered, it might be useful to choose a small set of documents to get shorter pointers.

Table 2: Pointer Structure (Approach 1)

Encoding	Total Bytes	Number of Docs (bit)	Number of Words (bit)	Number of Depth (bit)
General	5	262,144 (18)	262,144 (18)	16 (4)
Books	3	8 (3)	131,072 (17)	16 (4)
Tweets	5	1,048,576 (20)	256 (8)	16 (4)
Standard	3	32 (5)	32768 (15)	16 (4)

An alternative data structure would be a graph with the set W of all words as the node set, and two nodes w and w' connected by an arc (w, w') if words w and w' occur in sequence, i.e., if there exists a document D_i and an offset o such that $w = w_{i,o}$ and $w' = w_{i,o+1}$. Each arc (w, w') must be annotated with all supporting occurrences (i, o). Following a path to find a secret message $a_1, \dots, a_n$ in such a graph is cumbersome: we establish the longest path from a_0 to a_{k_0} such that all arcs (a_j, a_{j+1}) exist *and* the intersection of all occurrence sets is non-empty, i.e. there really is a document D_i with a sequence of words $w_{i,o}, \dots, w_{i,o+k_0-1}$. Intersection here means that two occurrences in succeeding arcs are considered equal if they refer to the same document and their offsets differ by 1. However, we did not explore this variant in our current implementation.

3.3.3 Encoding. To encode the pointers, we might use a static encoding: if there are m documents with offsets less than o_{max}, then we can use binary encodings of i_j, o_j and k_j with a total length of

$$L_{simple} = \lceil \log_2 m \rceil + \lceil \log_2 o_{max} \rceil + \lceil \log_2 l \rceil \quad (2)$$

bits. We see that it is useful if the number m of documents and the depth l of the trees is a power of 2. If, e.g., we use $m = 2^{18}$ documents with maximum offset less than $o_{max} = 2^{18}$ and tree of depth $l = 2^4$, we can encode a pointer in 40 bits or 5 bytes, which can be considered as a general coding approach. For some scenarios (few documents for example), shorter pointers may also be considered, allowing better compression rates. This leads to an optimization problem, however the focus of our experiments is to show, that *even if the pointer is not optimized*, Open-knowledge Compression allows compression rates better than classical compression. So, besides this general approach, the concrete number of documents and pointers depends on the state of the underlying documents. For books, fewer document pointers m are needed than for shorter texts like tweets. Nevertheless, larger offsets o are needed for books, as they contain more words than tweets. The pointer structures and the number of addressable elements for Approach 1 are presented in Tab. 2. We assume that for some scenarios, even a pointer consisting of at most two bytes can be a good choice.

If the average word has 5 characters (followed by a white space) and the average length k_j of a sentence found is 4, then we replace 20 characters of the secret message on average by a pointer of 4 bytes, i.e., we achieve a ratio of 5.

If we know distributions over the choice of documents and the choice of offsets in the pointers used for the intended messages, we might choose other encodings that produce shorter codes for the most frequently used pointers. If, e.g., $m = 2^{15}$, but $i_j < 128$ in 50% of the cases, then we might use a one byte code (with the uppermost bit 0) for these, and a two byte code (with the uppermost bit 1 in the first byte) for the remaining document indices, somewhat similar to Unicode UTF8 [52]. This leads to 1.5 byte on average to encode the document index, as in the case of a static binary encoding for 2^{12} documents. Also other choices for prefix codes, such as Huffman codes [27] could be considered, but are outside the scope of the present work. Finally, we might also use a static encoding of the pointers and apply a standard lossless compression on the sequence of pointers before sending them.

3.3.4 Absence Words. If no document D_{root} can be provided, the procedure as described would not allow the encoding as in Eq. (1) for words that are not present in the documents $D_1, D_2, \ldots$ the trees are based on. As words also can be encoded character by character, this can be used as a fallback method to encode names, abbreviations and other rare words not present in the trees. However, the sender needs to signal to the receiver how a word is encoded, which can still be achieved by using document index $i = 0$. followed by parameter k (4 bits) signaling of how many UTF-8 characters the word consists of, and a sequence of bytes representing the word's characters. If the word persists of more than 15 characters, an additional absence-word encoding has to be performed until all characters are transmitted. In comparison to UTF-8, the encoding of the word needs one additional byte, which is offset by the better encoding of multiple words. Single word encoding thus can be described as a pseudo-pointer

$$W = (0, k, C = c_0, \ldots c_{k-1}) \tag{3}$$

with a document pseudo-index 0, the number k of characters and the Word C, persisting of characters c_0 to c_{k-1}.

More compact encodings of the characters are possible, e.g. assigning less than 8 bits to frequent characters and more than 8 bits to seldom used characters. Deploying such strategies is an issue of future work, as we assume that such pseudo-pointers are not needed often and thus the influence will be small.

3.4 Approach 2 - Tree Nodes

3.4.1 High-Level Description. Similar to approach 1, sender and receiver have to agree on a set of documents $D_1, D_2, \ldots$ and communicate this set beforehand. Again, we have each document D_i consisting of words $w_{i,0}, w_{i,1}, \ldots, w_{i,n_i-1}$ where n_i is the length of the document. We then divide the document into sub-lists of length l by using a sliding window. The first list would contain $w_{i,0}, w_{i,1}, \ldots, w_{i,l-1}$, the second list $w_{i,1}, w_{i,2}, \ldots, w_{i,l}$, the third list $w_{i,2}, w_{i,3}, \ldots, w_{i,l+1}$ and so on. These lists are then used to create a prefix tree T. This sliding window allows us to include more different variations of word combinations and therefore cover more potential sequences. In this tree T, arcs hold no information and nodes only hold the word w_i that they were created with. A sequence of words, $w_0, \ldots, w_{k-1}$, with length k will correspond to a path consisting of vertices $(v_0, \ldots, v_{k-1})$ in the tree with length k. The maximum sequence length is determined by the parameter l, which is the maximum depth of T. Since paths in a tree are unique, and we always start from the 0 node, we can use the last node in the path to identify the entire path. To encode a message $a_1, \ldots, a_n$ with length n, the sender searches, starting with a_1, for the longest possible sequence of words that has a corresponding path in T. The sender then stores the node ID of the last node in this sequence and continues the search with the first a_i that was not yet included, until all words are encoded. The result is a list of node-IDs $[i_1, \ldots, i_k]$ which will be transmitted to the receiver.

To decode a message, the receiver starts with the first node-ID i_1 and traces the path from the root to this node while storing each word w_i from each node that is included in the path. Once all node-IDs are traced, the message is reconstructed as $w_1, \ldots, w_n$.

3.4.2 Tree Generation. We start with a list of lists of words $L = l_0, l_1, \ldots$ with $l_t = w_{i_t,o_t}, \ldots, w_{i_t,o_t+l-1}$. The list L contains all lists l_i generated by dissecting the documents with the sliding window explained before, and thus is identical to $\cup_w S_w$.

To generate the tree, we follow Alg. 1. As an example, Fig. 3 includes the node IDs, so that the sentence "Hello I am Steve" is encoded with node IDs 25 (Hello I am) and 28 (Steve).

Algorithm 1 Prefix Tree Generation

```
Initialize $T$ with root node $t_0$
for $l_t$ in $L$ do
    Focus $t_0$
    for $w_i$ in $l_t$ do
        if Focused node has child corresponding to $w_i$ then
            Focus corresponding child node
        else
            Add new child node corresponding to $w_i$
            Focus new child node
        end if
    end for
end for
```

3.4.3 Encoding. With this approach, each chunk of a message is only represented by the corresponding node ID of the tree, therefore we know the maximum possible ID that could to be used beforehand. So, we can choose a byte encoding format that supports integers up to the maximum node ID. So with up to 65,535 IDs we could use 2 bytes per ID, with up to 16,777,215 IDs we would use 3 bytes and so forth. This allows us to follow a simple encoding schema where each ID occupies the same space, which results in a simple byte alignment. The byte length of the encoding is dependent on the used tree and can easily be determined by sender and receiver after generating the tree. With this, we just concatenate the byte representations of all the node IDs to form the final message. Additionally, it would be easily possible to include an error correcting code on top of this raw encoding.

Absence Words. If a word can not be found in the tree, this approach allows for two options.

- *Spelling:* We write a node ID that is larger than the maximum node ID of the encoding tree to signal a spelled word. This is followed by the length of the word and then each letter of the word encoded as one byte. This allows for simple parsing and byte alignment of the resulting data.

- *NULL Pointer:* We write a null pointer with the correct size. This signals a missing word and will be skipped when decoding, while preserving the byte alignment of the data.

The number of absence words can be minimized (or eliminated) by including D_{root} in the tree generation and placing each word at tree depth one. This means, we can at least encode every isolated word with a single ID.

4 DICTIONARIES AND DOCUMENTS

In this section we describe how the dictionaries for our evaluation were chosen and further how they were grouped. Also, we discuss the potential usage of dictionary groups for different censorship circumvention scenarios.

4.1 Reference Groups for Dictionaries

As the compression rate of a text relies on the documents the dictionary is created from, we defined various scenarios to compare the compression rates with each other. Henceforth, we picked variable texts as foundation of our dictionaries and grouped them as follows:

- Books
- News articles
- Twitter tweets
- Wikipedia articles

Twitter had been recently renamed to X and the download of tweets had been limited after the data was collected. However, the principle of this group can still be adopted to other services similar to Twitter. Examples are Mastodon or Threads, which rely on publicly accessible user-generated content. Furthermore, the approach may be applied to other social media platforms like Facebook, Instagram, Pinterest, Reddit or Lemmy.

For each single source in each group, we created a dictionary and further created a joint dictionary consisting of all sources of a group. Additionally, we crafted a *super group*, containing all source documents. This procedure allows the evaluation of the compression ratios, depending on the number of total and distinct words of the sources. Further, this procedure allows the investigation of compression rates for texts that cannot be considered to be near a dictionary group. The static sources were collected at 11th of September 2022 from 10:00 AM UTC to 11:00 AM UTC. Twitter sources were collected on the 21st of September 2022 between 7:00 and 15:00 with a search query to include tweets from 1st January 2009 to the 21st September 2022. All sources of each group are presented in Appx. B in Tab. 3. For **books**, we picked three top 100 books from the Gutenberg project[2] in compliance of legal aspects. Each single book resulted in a dictionary of its own. Further, we created a joint books-dictionary. The **newspaper** groups were divided into two subgroups. We decided to create a dictionary for BBC-news articles and collected five top-articles available online, to cover short daily newspaper articles. In addition, we created a dictionary for longer articles created by Reuters Investigative[3] (further Reuters) to cover also background news-articles. Both subgroups are the fundamentals of the joint newspapers group. Also, we created dictionaries for publicly available social media and decided to utilize **Twitter**. We created a dictionary for each Twitter account referred in Appx. B Tab. 3 and created a joint dictionary for Twitter, based on all presented accounts. Finally, we utilized public **Wikipedia** articles and split them into the themed subgroups cities, history, scientists and sports. We decided to take advantage of four different setups to investigate the efficiency of OPPRESSION, even in themed dictionaries for texts, that are not covered by this specific topic. Further, we created a joint dictionary for Wikipedia, containing all referred articles. The supergroup, which created our super-dictionary, persists of all sources and documents.

[2]https://www.gutenberg.org/

[3]https://www.reuters.com/investigates/

4.2 OPPRESSION Modes and Dictionary Groups

We propose two **modes** of operation for OPPRESSION, that generally use it in the same way but under different circumstances:

1. Persistent In this mode, sender and receiver have to agree upon a set of documents that can be retrieved again and again, and so accessed at any time. The documents in the set themselves must also remain unchanged. Given these parameters, a message that has been encoded and sent today, can be decoded at any point in the future by referencing the unchanged set of documents.

This allows significant time delays between "sending" and "receiving" a message, that may be placed at a meeting point, reachable for both communication parties (i.e., a social media post, a comment on a website, a git repository, etc.). Such a delayed transmission concept has recently been described in [46] for the scope of indirect network covert channels and may easily be adopted. Data sources with a history feature or timestamps that can be used to reconstruct their state at a given time in the past lend themselves for this mode. Other feasible options include publicly available historical books.

2. Ephemeral In this mode, sender and receiver also have to agree upon a set of documents that can be retrieved by both of them. In this case, the persistence requirements no longer apply. Sender and receiver both take a "snapshot" of their document set at the same time, which ensures that both have, in fact, the same set. This set will then be used only for a limited amount of time, e.g., 24 hours. After that time, a new snapshot is taken and used for all messages during the next time period. Ideally, the document sets change significantly between the two snapshots, which means the encoded messages of two consecutive time slots are drastically different, even if the messages are the same. Therefore, an ever-changing data source is needed, such as a social media feed where a snapshot could be taken of the "1000 newest tweets for hashtag X" or the "news report of the day" from a news website. An added bonus would be if the state of the document set was *not* reconstructable, even if the time of the snapshot was known. As this would implement a mechanism of self-destruction for the messages, the reconstruction of secret content by the censor would be more difficult.

We further considered the following **groups** of source documents. The documents utilized, their according group, source and number of total words can be found in Appx. B in Tab. 4.

Books. As books do not change after being published, aside from errata, they can only be used in the persistent mode of OPPRESSION. If sender and receiver agree upon a certain edition of a book, it will be unchanged and still accessible in the future.

Newspapers. As we work with digital versions of newspapers, it depends on the setup of a particular news website which of the OPPRESSION modes can be used. The ephemeral mode can be implemented by a snapshot of the newest X articles at a given time. If articles are still freely available with a timestamp in the future, it is also possible to use a set of news articles for the persistent mode.

Twitter. Twitter can be used for both operation modes of OPPRESSION. The persistent mode can reference a fixed set of tweets from the past, which will not change (cf. Sect. 5.2). In the ephemeral mode, as long as a precise timestamp would be used by sender and receiver, Twitter data of a high-volume hashtag can be used to have a steady stream of new data for each snapshot.

Wikipedia. Wikipedia is mostly interesting for the persistent mode, as changes are less frequent compared to social media. Each article has a history of all changes that have been made. So, it is easily possible to reconstruct the state of each article to any given point in the past. However, Wikipedia also offers an "article of the day" which could be used for the ephemeral mode.

Compressed Documents. As we created dictionaries by utilizing documents from different sources and topics, we need to investigate if the approach can be applied with any type of dictionary. To ensure this, we decided to cross-check one additional document from each group and subgroup, not involved in the creation of the dictionaries.

5 EVALUATION

In this chapter, we evaluate the effectiveness and robustness of OPPRESSION. Afterward, we discuss its detectability, potential countermeasures and limitations of our work.

5.1 Effectiveness (Reduction of Transmission)

The compression performance of OPPRESSION is directly proportional to the amount of traffic that needs to be transmitted by the censorship circumvention tool. Therefore, we directly evaluated the compression performance.

To do so, we used reference texts and target messages from our scenarios (see Appx. B) and performed a cross validation of the datasets. We used the first 100, 200, ..., 1000 words of our target message, compressed them and saved the resulting file. To get the compression ratio, we divided the size of the compressed file by the size of the plain text file. Therefore, a lower ratio denotes better compression performance. Additionally, we used OPPRESSION in combination with GZIP by compressing the resulting file, which represents the transfer of a compressed pointer. For GZIP, we compressed the plain text file using the strongest compression options (`gzip -fk -9 -n <input file>`) and again calculated the compression ratio by the file sizes.

Fig. 4 and Fig. 5 show the results for approach 1 (Document Pointers) and approach 2 (Tree Nodes). Here we compare OPPRESSION (blue), OPPRESSION+GZIP (green) and GZIP (red) as baseline. Different shades of the color show the different tree depths that were used, a darker color signals a deeper tree. In the results, we find several interesting aspects:

(1) With approach 2, OPPRESSION outperforms GZIP significantly in many scenarios for shorter messages. The gap closes with longer messages.
(2) Approach 1 shows a similar pattern with slightly worse performance.
(3) The "combined" tree performs consistently well for all target messages, which is to be expected, as it contains the largest number of possible sentences.
(4) For approach 1, we can generally observe benefits of deeper trees.
(5) For approach 2, the tree depth does not generally influence the encoding performance significantly. A deeper tree can offer a better compression, as longer sentences can be encoded in a single node. On the other hand, it results in a larger tree with more nodes which forces us to use more bytes to identify a single node, which in turn results in a worse compression ratio. Additionally, finding a longer match becomes more and more rare. So even with a deep tree we might only ever encode three or four words at a time.

To get an overview of the compression performance with longer texts, we compressed messages up to 1,000 words for approach 1 and up to 100,000 words for approach 2 for all our scenarios and recorded the compression ratios. In Fig. 6 we can see that with longer texts, GZIP has a performance lead compared to approach 1 alone. But approach 1 in combination with GZIP closes the gap and can outperform GZIP. This plot shows again that deeper trees perform better for approach 1. This can be explained since with this approach, the pointer size does not increase with a deeper tree or with longer potential chunks. Therefore, we only gain performance with a deeper tree without suffering drawbacks.

In Fig. 7, we can see that for approach 2, OPPRESSION with a tree depth of 2 outperformed GZIP in most cases and gained an even bigger lead when combined with GZIP. This plot also shows the potential adverse effects of a too large tree depth for approach 2, as depth 15 performed worse than depth 2 in our tests.

Additionally, we investigated the match lengths to gauge the effectiveness of deeper trees. For approach 1, we compressed all reference documents (length set to 1000) using the combi-tree with maximum depth of 15. The results are shown in Fig. 8. For approach 2, we used the combined tree with a depth of 20 and compressed each target message up to 100,000 words and recorded the length of each sentence chunk that was encoded, cf. Fig. 9.

With approach 1, we mainly see matches of length 1 while the other lengths are significantly less frequent. As we do not have negative effects when using a deeper tree, it is still beneficial to use a deeper tree with this approach. With approach 2, we can see that most matches have a length of 1 or 2. While a length of 3 still shows some matches, anything above 4 is insignificant: There are even matches of length greater or equal to 10, but in total we only observed 8 of these. In our case, it would be beneficial to have a tree depth of at least 3. If the tree depth can be increased further without forcing a larger node encoding (e.g., 3 instead of 2 bytes per node ID), the tree depth could be increased further. This optimization depends on the specific tree that is generated from the chosen references but can be performed before deploying OPPRESSION.

All in all, OPPRESSION performed well in our tests.

We performed additional evaluations with different compression algorithms. The results can be found in App. C.

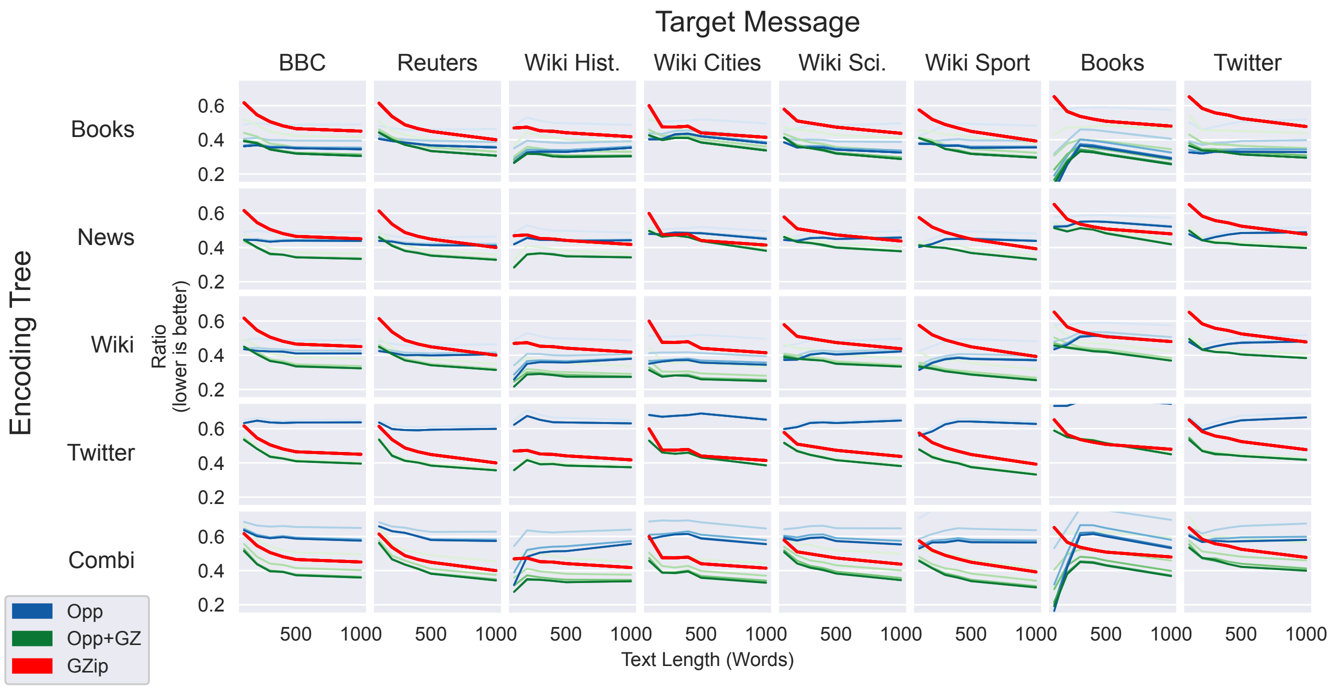

Figure 4: Evaluation Results: Doc. Pointer. Color shades indicate different tree depths (darker color signals a deeper tree).

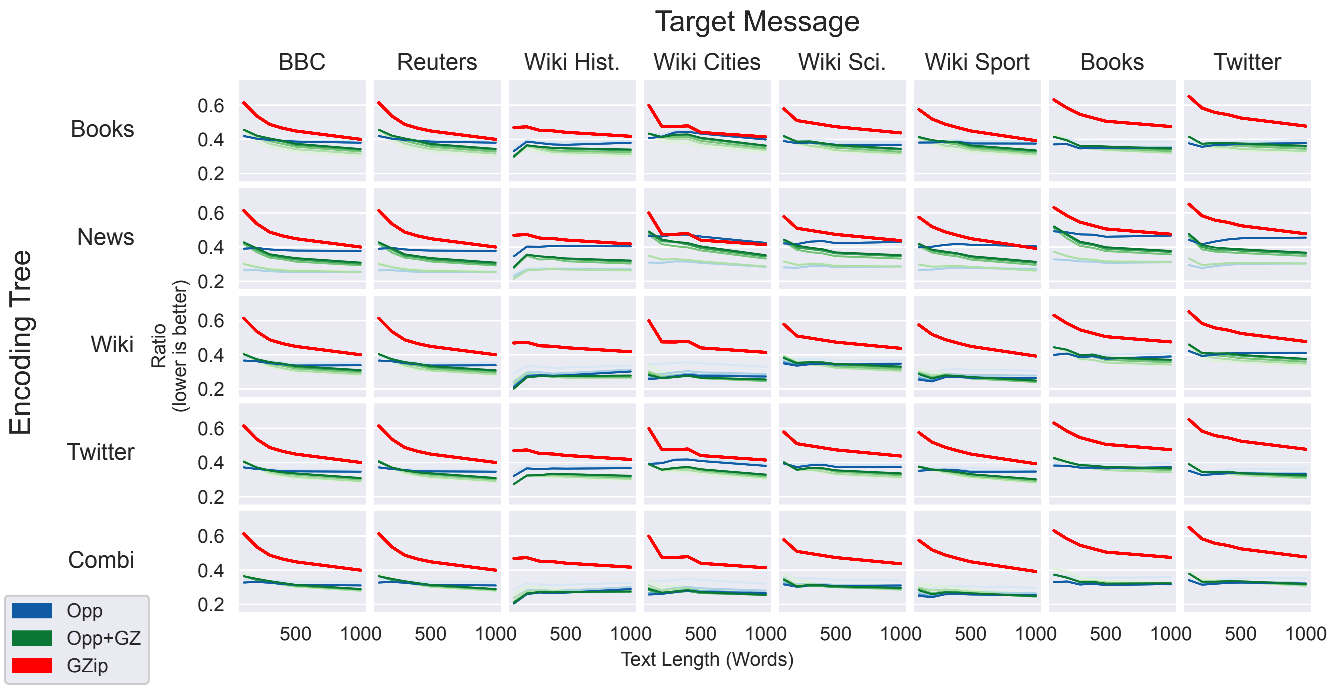

Figure 5: Evaluation Results: Node Tree. Color shades indicate different tree depths (darker color signals a deeper tree)

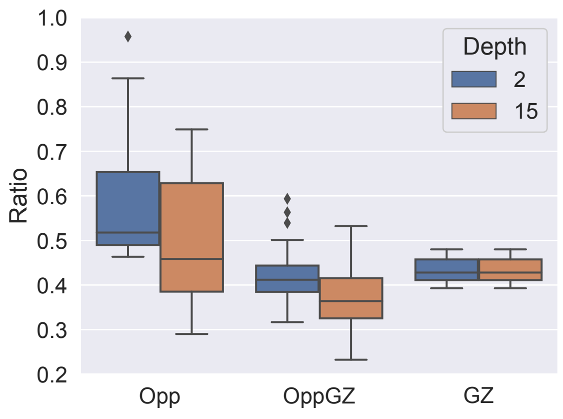

Figure 6: Doc Pointer: Compression Box Plot

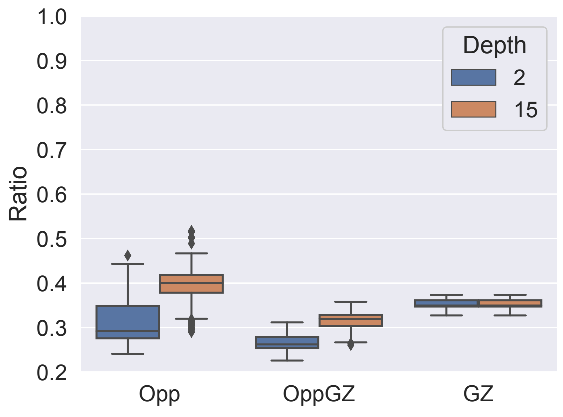

Figure 7: Node Tree: Compression Box Plot

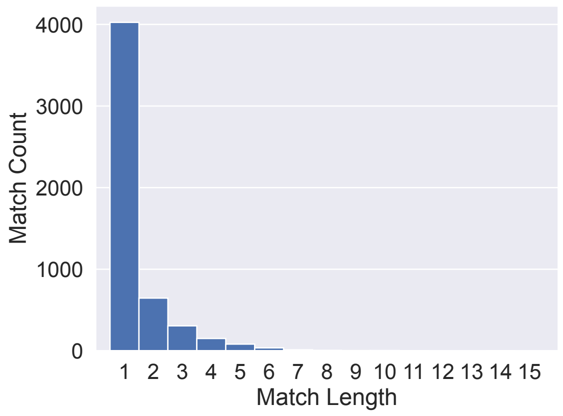

Figure 8: Doc Pointer: Match Lengths

5.2 Robustness

The robustness of OPPRESSION can be influenced by multiple factors, which we discuss in the following section.

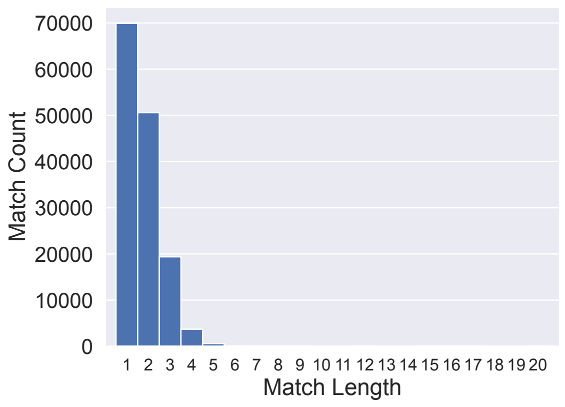

Figure 9: Node Tree: Match Lengths

5.2.1 Document Set. With OPPRESSION, we rely on the assumption that both, sender and receiver, can retrieve an identical set of documents to perform the encoding and decoding, crafting one and the same message. For the ephemeral mode, it is only important that sender and receiver can retrieve the same document set once, which allows a broader choice of data sources. However, the data sources still need to be evaluated to ensure that both parties will receive the same set of documents when accessing the data sources. As stated before, many unchanging data sources can be used for the persistent mode. We empirically evaluated our chosen sources by accessing them multiple times at different dates. We found no changes in our chosen sources.

Wikipedia. Wikipedia keeps a permanent history of all changes made to an article. Thus, it is easy to choose a certain past version of an article in the construction of the tree instead of the newest one available. We conducted an empirical evaluation of Wikipedia sources by downloading the same historical version of three articles 12 times over the span of a month while comparing the resulting hashes. We found no changes in any of the articles that we tested.

Twitter. In September 2022, Twitter added a feature to edit tweets [6]. But this feature only allows editing a tweet up to 30 minutes after initially publishing it, and it still retains a history of all edits. It therefore poses no problem for our use case; however, tweets may still be deleted. Similar to Wikipedia, we also evaluated Twitter by downloading the sources and comparing the resulting hashes. For our Twitter sources, we used a search query to only include tweets from a certain time frame. This procedure should result in the same set of tweets after each execution (except for deleted tweets). During our evaluation, we found however that the results of a search did change without any tweets being deleted. In detail, this means that some tweets, which were once included in our search results, were no longer included. However, if we accessed the missing tweets through their ID, they were still available. This means that Twitter might not be the best choice for the persistent mode but a good choice for the ephemeral mode.

All in all, we believe that it is possible to choose unchanging data sources, or at least sources that can be easily reconstructed (e.g., Wikipedia edit history) and conclude that altered documents are not a significant concern for OPPRESSION.

Nonetheless, an additional robustness measure that can be implemented is to take a hash sum of the chosen document set when it was first retrieved and to compare it to the hash of the newly re-retrieved document set before sending a message. This allows the sender or the receiver to notice potential problems with the dataset and allows them to adjust their document set by exchanging their stored version for the up-to-date version.

5.2.2 Document Availability. To some capacity, we have to rely on the hosting provider of our chosen data source, as there is no way for the sender to influence the uptime of foreign web servers. We therefore chose well-known and reliable data sources such as Wikipedia, Twitter, BBC etc.

Similar to the server uptime, we need to consider the general lifetime of our data sources. In our opinion, it is highly unlikely that any of our chosen data sources will go defunct in the near future, as all of them are large organizations or corporations with significant infrastructure. A censor may be able to block some texts available for encoding. However, it is not feasible to block all texts that may be utilized for OPPRESSION. Our approach includes also pages that are legitimate for a censor, e.g., one could also refer to pages created by a censor. However, still a censor may deny covert communication by modifying or manipulating the *carrier* of a covert channel, which is out of scope of this paper.

5.3 Discussion

5.3.1 Reliance on Third-party Censorship Circumvention Tools. OPPRESSION is not a censorship circumvention tool by itself. Instead, it is tailored to enhance the stealthiness for users of existing tools that take care of embedding a secret message into a carrier, e.g., through means of tunneling. This also provides the option to use OPPRESSION with upcoming tools should current ones become obsolete. OPPRESSION may improve the undetectability of third-party censorship circumvention tools that benefit from the minimization of caused secret traffic, e.g., Stegozoa [14]. Note that some censors are not relying on such mechanics but apply other heuristics [85].

5.3.2 Throughput. As OPPRESSION can be considered more a text encoding and compression strategy than a covert channel, the throughput depends on the covert channel implementation utilized. In general, our approach enables higher throughput rates, thus reducing the number of necessary covert channel packets, leading to a decreased risk of detection. In case of Stegozoa [14], the original throughput has been 8.2 Kbps for a stealthy implementation. The utilization of OPPRESSION would lead to a higher throughput, like theoretically analyzed in Fig. 10 for the transfer of 1 Megabyte (1,048,576 Bytes) of covert information. The calculation of the time reduction was performed for the mean reduction factor of the node tree approach with a depth of 2 that was shown in Fig. 7, which represents the best performing setup of OPPRESSION. Fig. 10(left) visualizes the time in seconds that is necessary to transfer 1 Megabyte of covert information from a sender to a receiver using the stealthy configuration of Stegozoa. The transmission time of GZIP-compressed covert information takes 45.03 seconds, while the transfer of OPPRESSION optimized covert information takes 40.28 seconds. The combination of OPPRESSION and GZIP reduces the theoretical transmission time to 33.93 seconds.

Fig. 10(right) visualizes the reduction of necessary transmission time of covert information in percent. As can be seen, OPPRESSION decreases the necessary transfer time and outperforms GZIP by 10.55%. The combination of OPPRESSION and GZIP reduces the necessary transmission time of covert information by 15.76% in comparison to OPPRESSION and by 24.64% in comparison to GZIP-compressed data, respectively. Please note that for our theoretical evaluation, we utilized the mean reduction factor of scenario 2 with a depth of 2. For highly specialized codebooks, the reduction rate would potentially perform better.

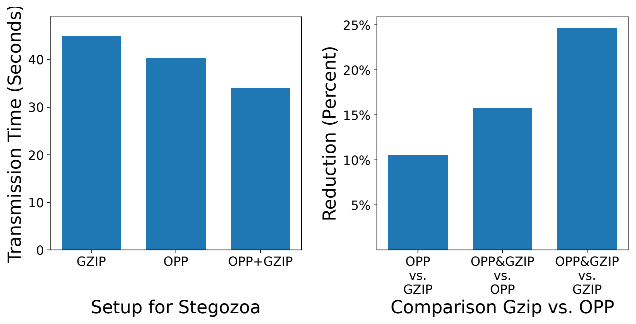

Figure 10: OPPRESSION with Stegozoa

5.3.3 Detectability and Countermeasures.
Detectability: The ultimate detectability of OPPRESSION depends on the censorship circumvention tool that is used to transfer the pointers. However, in general, shorter messages result in fewer anomalous traffic characteristics and less crafted traffic (fewer packets with fewer modifications) overall. Since several methods for the detection of stealthy communications rely on a rather high number of packets and modified packet content to provide quality results, we assume that OPPRESSION generally aids the covertness of all censorship circumvention tools that take advantage of it.

Further, we have assumed that both covert sender and receiver have access to the same online web content and that they send queries to the particular websites to build their tries. Note that a censor cannot correlate the access of covert sender and receiver to the same web resource as the receiver resides outside of the censor's network. However, if the sender cannot access any public web resource, such as Wikipedia, covert sender and receiver could agree to exploit the *censor's own websites* to build their trees. This would indeed allow a censor correlate covert sender's and receiver's access queries.

Countermeasures: Unblockable communication protocols do not exist [83], but certain countermeasures against OPPRESSION can be imagined. A state-level censor might manipulate sources (e.g., available websites) regularly so that it replaces words with synonyms. This would at least influence the communication between sender and receiver if they did not already build their trees, and is especially problematic for the ephemeral mode (the persistent mode can theoretically keep codebooks forever). However, a censor could still force the automated rewriting of texts before delivery, for example under the utilization of machine learning services like ChatGPT. This would result in different dictionaries for sender and receiver, drawing OPPRESSION infeasible.

5.3.4 Applicability. The main purpose of OPPRESSION is to facilitate censorship circumvention for scenarios where sender and receiver can access the same online resource. Due to the fact that sender and receiver build up their codebook using any available online resource, their codebook can be used any time in the future. This means that a censor might block the access to some online resource but sender and receiver can *still* point to the resource. Fewer data transferred thanks to better compression would also positively influence the QoS experience, which is a shortcoming of some tools [40].

5.3.5 Alternative Use Cases. Beside censorship circumvention, OPPRESSION may also be applied to various other scenarios. OPPRESSION can be used as a general encoding approach to reduce the necessary space to store textual information, after adaptions to include punctuation marks and other characters frequently appearing in texts, e.g., by creating additional pseudo-pointers. Highly repetitive texts might be stored more efficient, especially in archive scenarios where information is not supposed to be modified after filing. Also, coding structures may be "oppressed" as the fundamental structure of machine code is also highly repetitive and therefore a possible application scenario for OPPRESSION.

5.3.6 Ethical Considerations. Methods to improve censorship circumvention can be considered as a dual-use good. OPPRESSION is no exception and the improved coding could potentially be applied to enhance the performance and undetectability of malware communications or data exfiltration. While we cannot eliminate such use cases, we believe that keeping OPPRESSION secret would prevent its benefits to people facing censorship, but still not prevent its re-invention and deployment by malware producers, with the additional damage that countermeasures would not have been researched in that case. For this reason, the publication of OPPRESSION and Open-knowledge Compression as a methodology outweighs its concealment.

5.3.7 Document Pointers vs. Tree Nodes. Document pointers and tree nodes have advantages and disadvantages in specific situations to circumvent censorship. Document pointers benefit from the fact, that only the sending communication party needs to craft the dictionary. The dictionary creation both consumes CPU time and disc space and may be noisy, if the system of the receiver is monitored. For this approach, also a trie *for each word* has been created, splitting each dictionary in smaller chunks. This may also allow a better hiding of the dictionary on a local system, if necessary. Beside of these benefits, the receiving party may look up the document of the open-knowledge dictionary directly after receiving the message. This may lead to a timely correlation if more messages are received, unveiling the dictionaries and the communication itself. Further, the encoding process may be more obvious as more than one tree is necessary to compress the message, leading to more potential I/O operations. Opposed to that, for the tree nodes approach, both parties create one tree in advance. Theoretically, no access to a document is needed after this creation, reducing necessary network activities. However, the receiver also needs to create the dictionary, which may lead to detection if the system itself is monitored. Further, the concept relies on one single tree, leading to a potentially large stored file (1GB) on the file system of both communication parties.

5.3.8 Further Enhancements. OPPRESSION could be enhanced by further optimization methods that minimize the size of the secret message by reducing the number of words transferred through raw UTF encoding instead of links to the tree. For instance, one could aim at optimizing the generation and utilization of our codebook by mapping multiple synonyms to only one (short) word, e.g., "researcher" and "scientist" could both be mapped to "scientist" without losing much semantic information. An alternative (or additional) enhancement could be the integration of auto-completion in the sense that the sender is provided with word-completions from the database while typing, thus, increasing the chance for using words that are already included in the database.

Similarly, it would be beneficial to normalize spelling by a) fixing typos b) using consistent spelling (American vs. British English) c) removing accents and other diacritics. This would again increase the similarity between the source and target texts.

Additionally, the length of the pointer can be optimized to decrease potential overheads. Furthermore, the dictionary creation could be improved by specialized concepts like fully-online suffix trees and directed acyclic word graphs (such as described in [51]).

6 CONCLUSION

We present a novel method to enhance the stealthiness of censorship circumvention. To this end, our method, called OPPRESSION (*Open-knowledge Compression*), minimizes the size of textual data transfers based on codebooks crafted by publicly available online data. Our implementation of OPPRESSION can be used with arbitrary (pre-existing) censorship circumvention tools, as only the transferred content is modified.

OPPRESSION offers two different modes, persistent and ephemeral mode, depending on the users' demands. Further, our evaluation demonstrates that Open-knowledge Compression outperforms GZIP-based text compression while providing robustness.

In future work, we plan to analyze additional online platforms and media formats, such as video platforms, as well as the *Internet Archive* and aim at further optimizing achieved compression rates using alternative coding methods. Further, we plan to optimize the generation and utilization of our codebook by mapping multiple synonyms to only one (short) word and by integrating auto-completion on the level of a user interface.

Code and Data Availability Statement

To aid replicability and further research, this submission is accompanied by our Python-based implementation of OPPRESSION. Our code is available on Github:

https://github.com/Stego-Punk-Lab/OPPRESSION

ACKNOWLEDGMENTS

The work of S. Zillien and S. Wendzel was funded by the BMBF project *EMPOWER* (FKZ 03IHS242D).

REFERENCES

[1] Milad Taleby Ahvanooey, Mark Xuefang Zhu, Wojciech Mazurczyk, and Malika Bendechache. 2022. Information Hiding in Digital Textual Contents: Techniques and Current Challenges. *Computer* 55, 6 (2022), 56–65. https://doi.org/10.1109/MC.2021.3113922

[2] Jyrki Alakuijala and Zoltan Szabadka. 2016. Brotli Compressed Data Format. RFC 7932. https://doi.org/10.17487/RFC7932

[3] Anonymous. Sept 10, 2022. Hong Kong: Five jailed for 'seditious' children's books. *BBC* (Sept 10, 2022). https://www.bbc.com/news/world-asia-china-62863622 https://www.bbc.com/news/world-asia-china-62863622.

[4] Jane Austen. 1998. *Pride and Prejudice.* Project Gutenberg. https://www.gutenberg.org/files/1342/1342-0.txt

[5] Diogo Barradas, Nuno Santos, Luís E. T. Rodrigues, and Vítor Nunes. 2020. Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020,* Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM, 35–48. https://doi.org/10.1145/3372297.3417874

[6] Twitter Blog. 2022. Blog entry on tweet edit feature. https://blog.twitter.com/en_us/topics/product/2022/twitter-new-edit-tweet-feature-only-test.

[7] DEF CON. 2022. Twitter. https://twitter.com/defcon.

[8] Sir Arthur Conan Doyle. 1999. *The Adventures of Sherlock Holmes.* Project Gutenberg. https://www.gutenberg.org/ebooks/1661/1661-0.txt

[9] Sean Coughlan and Claire Heald. Sept 11, 2022. Queen to lie in state for four full days before state funeral. *BBC* (Sept 11, 2022). https://www.bbc.com/news/uk-62863486 https://www.bbc.com/news/uk-62863486.

[10] John Davison and Ahmed Rasheed. Aug 23, 2022. A Shi'ite Divine. *Reuters* (Aug 23, 2022). https://www.reuters.com/investigates/special-report/iraq-iran-shiites/ https://www.reuters.com/investigates/special-report/iraq-iran-shiites/.

[11] Rene De La Briandais. 1959. File Searching Using Variable Length Keys. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference* (San Francisco, California) *(IRE-AIEE-ACM '59 (Western)).* Association for Computing Machinery, New York, NY, USA, 295–298. https://doi.org/10.1145/1457838.1457895

[12] L. Peter Deutsch. 1996. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951. https://doi.org/10.17487/RFC1951

[13] Peter Eisler and Nathan Layne. Jul 29, 2022. Machine Politics. *Reuters* (Jul 29, 2022). https://www.reuters.com/investigates/special-report/usa-elections-michigan-investigation/ https://www.reuters.com/investigates/special-report/usa-elections-michigan-investigation/.

[14] Gabriel Figueira, Diogo Barradas, and Nuno Santos. 2022. Stegozoa: Enhancing WebRTC Covert Channels with Video Steganography for Internet Censorship Circumvention. In *Proc. 2022 ACM on Asia Conference on Computer and Communications Security (AsiaCCS).* 1154–1167.

[15] Edward Fredkin. 1960. Trie Memory. *Commun. ACM* 3, 9 (sep 1960), 490–499. https://doi.org/10.1145/367390.367400

[16] Jessica Fridrich. 2009. *Steganography in Digital Media: Principles, Algorithms, and Applications.* Cambridge University Press. https://doi.org/10.1017/CBO978113919290

[17] James Gallagher. Sept 9, 2022. Self-sterilising plastic kills viruses like Covid. *BBC* (Sept 9, 2022). https://www.bbc.com/news/health-62797775 https://www.bbc.com/news/health-62797775.

[18] Bill Gates. 2022. Twitter. https://twitter.com/BillGates.

[19] David Gauthier-Villars, Steve Stecklow, Maurice Tamman, Stephen Grey, and Andrew Macaskill. Aug 8, 2022. Deadly Trade. *Reuters* (Aug 8, 2022). https://www.reuters.com/investigates/special-report/ukraine-crisis-russia-missiles-chips/ https://www.reuters.com/investigates/special-report/ukraine-crisis-russia-missiles-chips/.

[20] John Geddes, Max Schuchard, and Nicholas Hopper. 2013. Cover your ACKs: Pitfalls of covert channel censorship circumvention. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS'13).* ACM, 361–372. https://doi.org/10.1145/2508859.2516742

[21] Phillipa Gill, Masashi Crete-Nishihata, Jakub Dalek, Sharon Goldberg, Adam Senft, and Greg Wiseman. 2015. Characterizing web censorship worldwide: Another look at the opennet initiative data. *ACM Transactions on the Web (TWEB)* 9, 1 (2015), 1–29.

[22] Jacob Grimm and Wilhelm Grimm. 2001. *Grimms' Fairy Tales.* Project Gutenberg. https://www.gutenberg.org/files/2591/2591-0.txt

[23] David Harle and James R. Irvine. 2000. *Data Communication and Networks: An Engineering Approach.* John Wiley & Sons, Inc., USA.

[24] Stephanie Heitman. 2022. What Happens in an Internet Minute on 2022. LOCALiQ. https://localiq.com/blog/what-happens-in-an-internet-minute/.

[25] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy.* IEEE, 65–79.

[26] Amir Houmansadr, Thomas J Riedl, Nikita Borisov, and Andrew C Singer. 2013. I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention. In *2013 Network and Distributed System Security (NDSS) Symposium.*

[27] David A. Huffman. 1952. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898

[28] Ben Jones, Sam Burnett, Nick Feamster, Sean Donovan, Sarthak Grover, Sathya Gunasekaran, and Karim Habak. 2014. Facade: High-Throughput, Deniable Censorship Circumvention Using Web Search. In *4th USENIX Workshop on Free and Open Communications on the Internet (FOCI 14).*

[29] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M Swanson, Steven J Murdoch, and Ian Goldberg. 2016. SoK: Making sense of censorship resistance systems. *Proceedings on Privacy Enhancing Technologies* 2016, 4 (2016), 37–61.

[30] Paul Kirby. Sept 20, 2022. Ukraine war: Occupied areas call urgent vote to join Russia. *BBC* (Sept 20, 2022). https://www.bbc.com/news/world-europe-62965998 https://www.bbc.com/news/world-europe-62965998.

[31] H. Liu, C. Zhang, Z. Wang, et al. 2023. To deliver more information in coverless information hiding. *Multimed Tools and Applications* (2023). https://doi.org/10.1007/s11042-023-15263-7

[32] Anna Harbluk Lorimer, Lindsey Tulloch, Cecylia Bocovich, and Ian Goldberg. 2021. OUStralopithecus: Overt User Simulation for Censorship Circumvention. In *Proc. 20th workshop on Privacy in the Electronic Society.* 137–150.

[33] Jzef Lubacz, Wojciech Mazurczyk, and Krzysztof Szczypiorski. 2008. Hiding data in VoIP. In *Proc. 26th Army Science Conference.*

[34] Andrew R.C. Marshall and Joseph Tanfani. Aug 22, 2022. SkewTube. *Reuters* (Aug 22, 2022). https://www.reuters.com/investigates/special-report/usa-media-misinformation/ https://www.reuters.com/investigates/special-report/usa-media-misinformation/.

[35] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. 2016. *Information hiding in communication networks: fundamentals, mechanisms, applications, and countermeasures.* Wiley-IEEE Press. https://doi.org/10.1002/9781119081715

[36] Poppy McPherson and Wa Lone. Aug 4, 2022. Planned Purge. *Reuters* (Aug 4, 2022). https://www.reuters.com/investigates/special-report/myanmar-rohingya-warcrimes-investigation/ https://www.reuters.com/investigates/special-report/myanmar-rohingya-warcrimes-investigation/.

[37] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. SkypeMorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security.* 97–108.

[38] Donald R. Morrison. 1968. PATRICIA–Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM* 15, 4 (oct 1968), 514–534. https://doi.org/10.1145/321479.321481

[39] Elon Musk. 2022. Twitter. https://twitter.com/elonmusk.

[40] Milad Nasr, Hadi Zolfaghari, Amir Houmansadr, and Amirhossein Ghafari. 2020. MassBrowser: Unblocking the Censored Web for the Masses, by the Masses. In *2020 Network and Distributed System Security (NDSS) Symposium.*

[41] Barack Obama. 2022. Twitter. https://twitter.com/BarackObama.

[42] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. 2020. Decentralized control: A case study of Russia. In *2020 Network and Distributed Systems Security (NDSS) Symposium.*

[43] Condoleezza Rice. 2022. Twitter. https://twitter.com/condoleezzarice.

[44] Justin Rowlatt. Sept 8, 2022. PM will explore energy market reform to cut bills. *BBC* (Sept 8, 2022). https://www.bbc.com/news/science-environment-62832029 https://www.bbc.com/news/science-environment-62832029.

[45] Sarah Huckabee Sanders. 2022. Twitter. https://twitter.com/SarahHuckabee.

[46] Tobias Schmidbauer and Steffen Wendzel. 2022. SoK: A Survey Of Indirect Network-level Covert Channels. In *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022,* Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (Eds.). ACM, 546–560. https://doi.org/10.1145/3488932.3517418

[47] Priyanka Shankar. Sept 9, 2022. Georgia's daring, death-defying pilgrimage. *BBC* (Sept 9, 2022). https://www.bbc.com/travel/article/20220908-georgias-daring-death-defying-pilgrimage https://www.bbc.com/travel/article/20220908-georgias-daring-death-defying-pilgrimage.

[48] The Ellen Show. 2022. Twitter. https://twitter.com/.

[49] Johnathan Spicer. Aug 31, 2022. Muzzled Media. *Reuters* (Aug 31, 2022). https://www.reuters.com/investigates/special-report/turkey-erdogan-media/ https://www.reuters.com/investigates/special-report/turkey-erdogan-media/.

[50] Bram Stoker. 1995. *Dacula by Bram Stoker.* Project Gutenberg. https://www.gutenberg.org/cache/epub/345/pg345.txt

[51] Takuya Takagi, Shunsuke Inenaga, Hiroki Arimura, Dany Breslauer, and Diptarama Hendrian. 2020. Fully-Online Suffix Tree and Directed Acyclic Word Graph Construction for Multiple Texts. *Algorithmica* 82, 5 (2020), 1346–1377. https://doi.org/10.1007/s00453-019-00646-w

[52] The Unicode Consort. (Ed.). 2021. *The Unicode Standard v14.0.* The Unicode Consort., Mountain View, CA. https://www.unicode.org/versions/Unicode14.0.0/.

[53] Twitter. 2022. Twitter. https://twitter.com/Twitter.

[54] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. 2012. Stegotorus: a camouflage

proxy for the tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 109–120.

[55] Steffen Wendzel, Tobias Schmidbauer, Sebastian Zillien, and Jörg Keller. 2022. *DYST (Did You See That?): An Amplified Covert Channel That Points To Previously Seen Data*. Technical Report. arXiv. https://doi.org/10.48550/ARXIV.2212.11850 https://arxiv.org/abs/2212.11850.

[56] Wikipedia. 2022. Ada Lovelace — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Ada%20Lovelace&oldid=1112020624.

[57] Wikipedia. 2022. Alan Turing — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Alan%20Turing&oldid=1109177354.

[58] Wikipedia. 2022. Albert Einstein — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Albert%20Einstein&oldid=1109900732.

[59] Wikipedia. 2022. American football — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=American%20football&oldid=1110666744.

[60] Wikipedia. 2022. Association football — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Association%20football&oldid=1111179380.

[61] Wikipedia. 2022. Baseball — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Baseball&oldid=1109102597.

[62] Wikipedia. 2022. Cape Town — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Cape%20Town&oldid=1109760698.

[63] Wikipedia. 2022. Golf — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Golf&oldid=1109593868.

[64] Wikipedia. 2022. History of China — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=History%20of%20China&oldid=1108716299.

[65] Wikipedia. 2022. History of France — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=History%20of%20France&oldid=1110516646.

[66] Wikipedia. 2022. History of Japan — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=History%20of%20Japan&oldid=1111021728.

[67] Wikipedia. 2022. History of the United States — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=History%20of%20the%20United%20States&oldid=1111221745.

[68] Wikipedia. 2022. Holy Roman Empire — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Holy%20Roman%20Empire&oldid=1111347768.

[69] Wikipedia. 2022. Ice hockey — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Ice%20hockey&oldid=1110086263.

[70] Wikipedia. 2022. Isaac Newton — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Isaac%20Newton&oldid=1110748051.

[71] Wikipedia. 2022. Jennifer Doudna — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Jennifer%20Doudna&oldid=1106083146.

[72] Wikipedia. 2022. John Forbes Nash Jr. — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=John%20Forbes%20Nash%20Jr.&oldid=1111243738.

[73] Wikipedia. 2022. John Snow — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=John%20Snow&oldid=1095496350.

[74] Wikipedia. 2022. London — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=London&oldid=1111184952.

[75] Wikipedia. 2022. Los Angeles — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Los%20Angeles&oldid=1111259679.

[76] Wikipedia. 2022. Marie Curie — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Marie%20Curie&oldid=1110123437.

[77] Wikipedia. 2022. Melbourne — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Melbourne&oldid=1111327847.

[78] Wikipedia. 2022. Nikola Tesla — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Nikola%20Tesla&oldid=1110426323.

[79] Wikipedia. 2022. Rio de Janeiro — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Rio%20de%20Janeiro&oldid=1111063807.

[80] Wikipedia. 2022. Tu Youyou — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Tu%20Youyou&oldid=1100430160.

[81] Wikipedia. 2022. Vienna — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Vienna&oldid=1110876374.

[82] "Wikipedia". 2022. William Arthur Lewis — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=W.%20Arthur%20Lewis&oldid=1093746812.

[83] Philipp Winter, Tobias Pulls, and Juergen Fuss. 2013. ScrambleSuit: A polymorphic network protocol to circumvent censorship. In *Proc. 12th ACM workshop on privacy in the electronic society*. 213–224.

[84] Hans Friedrich Witschel and Chris Biemann. 2006. Rigorous dimensionality reduction through linguistically motivated feature selection for text categorization. In *Proceedings of the 15th Nordic Conference of Computational Linguistics (NODALIDA 2005)*. University of Joensuu, Finland, Joensuu, Finland, 210–217. https://aclanthology.org/W05-1729

[85] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. 2023. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 2653–2670. https://www.usenix.org/conference/usenixsecurity23/presentation/wu-mingshi

[86] Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2019. RNN-Stega: Linguistic Steganography Based on Recurrent Neural Networks. *IEEE Transactions on Information Forensics and Security* 14, 5 (2019), 1280–1295. https://doi.org/10.1109/TIFS.2018.2871746

[87] Zhong-Liang Yang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2021. VAE-Stega: Linguistic Steganography Based on Variational Auto-Encoder. *IEEE Transactions on Information Forensics and Security* 16 (2021), 880–895. https://doi.org/10.1109/TIFS.2020.3023279

[88] Zhili Zhou, Yan Mu, and Q. M. Jonathan Wu. 2019. Coverless image steganography using partial-duplicate image retrieval. *Soft Computing* 23 (2019), 4927–4938.

[89] J. Ziv and A. Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* 23, 3 (1977), 337–343. https://doi.org/10.1109/TIT.1977.1055714

A LIST OF ABBREVIATIONS AND TERMS

The following (partially uncommon) abbreviations were used in this paper:

Combi-(Tree)	Combined encoding Tree, using all available source materials
DYST	*Did You See That* (first history covert channel implementation)
GZ	*see GZIP*
GZIP	GNU Zip
HICCUPS	HIdden Communication system for CorrUPted networkS
LZ77	Lempel-Ziv 77
Opp	OPPRESSION
OPPRESSION	Open-knowledge Compression
OppGz/Opp+Gz	OPPRESSION followed by GZip
Opp+Brt	OPPRESSION followed by Brotli
QoS	Quality of Service
RNN-Stega	Recurrent Neural Networks Steganogr.
UTF	Unicode Transformation Format
WebRTC	Web Real-Time Communication

B DICTIONARY TEXTS

The texts utilized in this paper are listed in the following tables. Tab. 3 lists the open access articles and books, that were utilized to create the dictionaries for our investigation of Open-knowledge Compression. These texts were grouped (G) by their type and split into subgroups (S) like described in Sect. 4.

The texts utilized for the encoding's cross-validation of OPPRESSION are presented in Tab. 4 and are also based on freely accessible resources. The texts were picked from each group and subgroup introduced in Sect. 4.

Table 3: Text Reference Groups and Subgroups

G	S	Title	Ref	Words	
				Entire	Dist.
Books		Grimms Fairy Tales	[22]	61,066	3,881
Books		Pride and Prejudice	[4]	124,749	6,579
Books		The Adventures of Sherlock Holmes	[8]	107,560	8,169
Books		Books		293,370	12,082
Newspapers	BBC	Self-sterilising plastic kills viruses like covid	[17]	573	–
Newspapers	BBC	PM will explore energy market reform to cut bills	[44]	662	–
Newspapers	BBC	Queen to lie in state for four full days before state funeral	[9]	771	–
Newspapers	BBC	Hong Kong: Five jailed in for seditious children's books	[3]	333	–
Newspapers	BBC	Georgia's daring, death-defying pilgrimage	[47]	1,919	–
Newspapers	BBC	BBC		4,258	1,328
Newspapers	Reuters	Muzzled Media	[49]	2,696	–
Newspapers	Reuters	A Shi'ite Divide	[10]	3,610	–
Newspapers	Reuters	SkewTube	[34]	4,048	–
Newspapers	Reuters	Deadly Trade	[19]	3,253	–
Newspapers	Reuters	Planned Purge	[36]	5,855	–
Newspapers	Reuters	Reuters		19,462	3,843
Newspapers		Newspapers		23,720	4,542
Twitter		Barack Obama	[41]	260,326	9,288
Twitter		Bill Gates	[18]	77,875	6,079
Twitter		Condoleezza Rice	[43]	8,444	1,895
Twitter		Elon Musk	[39]	217,447	13,307
Twitter		Sarah Huckabee Sanders	[45]	42,745	4,995
Twitter		DefCon	[7]	118,539	10,152
Twitter		Twitter	[53]	102,293	8,017
Twitter		Twitter		827,669	26,009
Wikipedia	Cities	Cape Town	[62]	15,999	–
Wikipedia	Cities	London	[74]	17,333	–
Wikipedia	Cities	Los Angeles	[75]	11,406	–
Wikipedia	Cities	Melbourne	[77]	11,314	–
Wikipedia	Cities	Vienna	[81]	11,733	–
Wikipedia	Cities	Cities		67,785	8,614
Wikipedia	History	History of France	[65]	31,391	–
Wikipedia	History	History of Japan	[66]	12,177	–
Wikipedia	History	History of the USA	[67]	23,245	–
Wikipedia	History	Holy Roman Empire	[68]	14,079	–
Wikipedia	History	History		80,892	9,452
Wikipedia	Scientists	Alan Turing	[57]	8,690	–
Wikipedia	Scientists	Albert Einstein	[58]	14,575	–
Wikipedia	Scientists	Isaak Newton	[70]	8,888	–
Wikipedia	Scientists	Jennifer Doudna	[71]	3,025	–
Wikipedia	Scientists	John Forbes jr.	[72]	3,229	–
Wikipedia	Scientists	John Snow	[73]	3,014	–
Wikipedia	Scientists	Marie Curie	[76]	7,137	–
Wikipedia	Scientists	Nicola Tesla	[78]	12,747	–
Wikipedia	Scientists	Tu Youyou	[80]	1,598	–
Wikipedia	Scientists	W. Arthur Lewis	[82]	1,709	–
Wikipedia	Scientists	Scientists		58,280	9,070
Wikipedia	Sports	Association Football	[60]	7,375	–
Wikipedia	Sports	Baseball	[61]	10,844	–
Wikipedia	Sports	Golf	[63]	9,730	–
Wikipedia	Sports	Ice Hockey	[69]	13,791	–
Wikipedia	Sports	Sports		41,640	5,990
Wikipedia		Wikipedia		248,597	21,010
Total - Supergroup				1,393,361	42,136

Table 4: Reference sources for cross-validation

Title	Group	Ref	Words	
			Total	Dist.
Dracula by Bram Stoker	Book	[50]	164,382	9,599
Ukraine war: Occupied areas call urgent vote to join Russia	Newspaper BBC	[30]	828	401
Machine Politics	Newspaper Reuters	[13]	4,806	1,237
The Ellen Show	Twitter	[48]	320,080	14,645
Rio de Janeiro	Wikipedia City	[79]	14,577	3,200
History of China	Wikipedia History	[64]	12,858	2,916
Ada Lovelace	Wikipedia Scientist	[56]	5,932	1,750
American Football	Wikipedia Sports	[59]	10,487	1,989

C ADDITIONAL COMPRESSION TESTS

We tested different well-known compression algorithms against a text passage to gauge their relative performance in our scenario. The results of this evaluation are shown in Fig. 11. We can see, that Brotli [2] delivers the best results for this test.

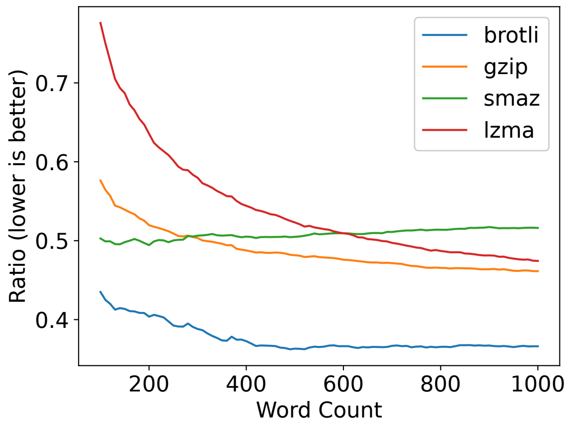

Figure 11: Compression Algorithm Test

Therefore, we also evaluated the Node Tree approach against Brotli and produced the same plot as for GZip (see Fig. 5). For the evaluation, we again used OPPRESSION, Bortli, and OPPRESSION combined with Brotli. Fig. 12 shows the results of these evaluations. We can observe that Brotli performed better than GZip, somewhat closing the gap. But in most cases, there is a configuration of OPPRESSION or OPPRESSION with Brotli that performs better than Brotli alone. If we only consider the combi tree, OPPRESSION performed better than Brotli in all cases. We can therefore conclude that OPPRESSION is a valuable addition, especially for short messages.

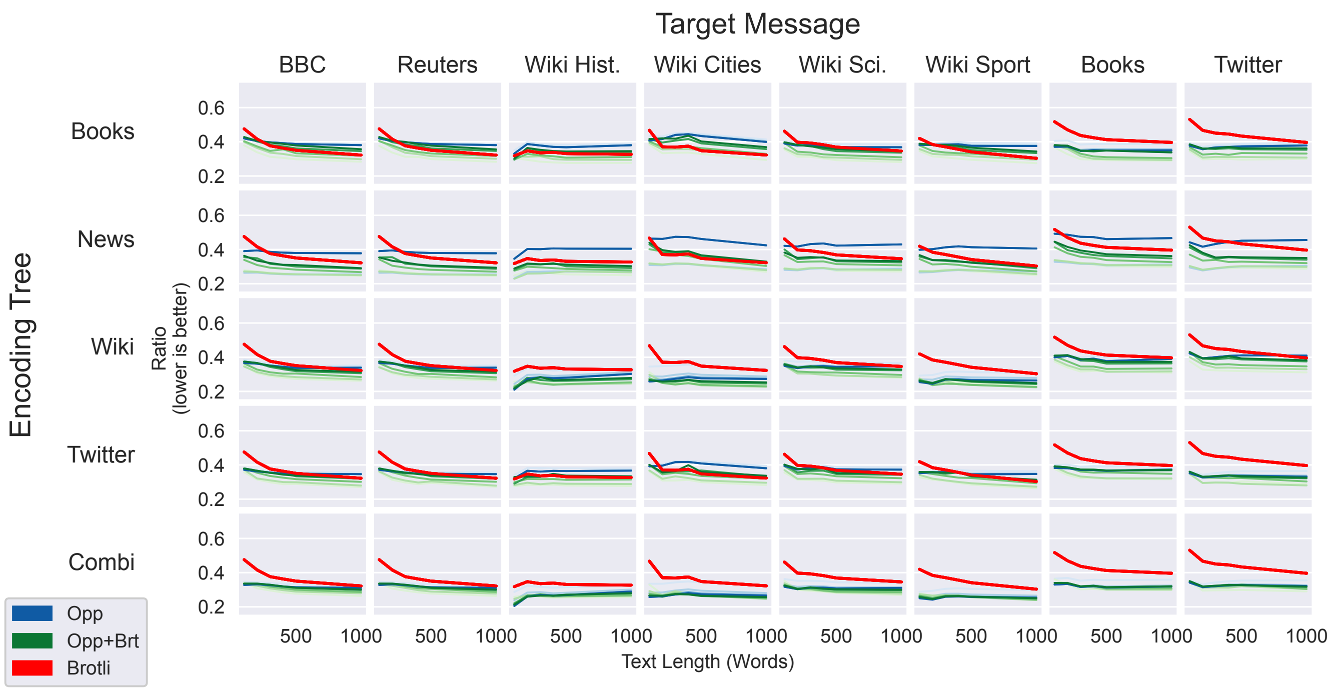

Figure 12: Evaluation Results: Node Tree vs. Brotli. Color shades indicate tree depths (darker color signals a deeper tree)