Elxa: Scalable Privacy-Preserving Plagiarism Detection*

Nik Unger
University of Waterloo
njunger@uwaterloo.ca

Sahithi Thandra
University of Waterloo
sthandra@uwaterloo.ca

Ian Goldberg
University of Waterloo
iang@cs.uwaterloo.ca

ABSTRACT

One of the most challenging issues facing academic conferences and educational institutions today is plagiarism detection. Typically, these entities wish to ensure that the work products submitted to them have not been plagiarized from another source (e.g., authors submitting identical papers to multiple journals). Assembling large centralized databases of documents dramatically improves the effectiveness of plagiarism detection techniques, but introduces a number of privacy and legal issues: all document contents must be completely revealed to the database operator, making it an attractive target for abuse or attack. Moreover, this content aggregation involves the disclosure of potentially sensitive private content, and in some cases this disclosure may be prohibited by law.

In this work, we introduce Elxa, the first scalable centralized plagiarism detection system that protects the privacy of the submissions. Elxa incorporates techniques from the current state of the art in plagiarism detection, as evaluated by the information retrieval community. Our system is designed to be operated on existing cloud computing infrastructure, and to provide incentives for the untrusted database operator to maintain the availability of the network. Elxa can be used to detect plagiarism in student work, duplicate paper submissions (and their associated peer reviews), similarities between confidential reports (e.g., malware summaries), or any approximate text reuse within a network of private documents. We implement a prototype using the Hadoop MapReduce framework, and demonstrate that it is feasible to achieve competitive detection effectiveness in the private setting.

1. INTRODUCTION

Plagiarism—a form of academic dishonesty where one copies the ideas or words of another person without providing a citation—is an increasingly common issue. A pair of Pew Research Center surveys conducted in 2011 found that 55% of college presidents have seen an increase in plagiarism over the past 10 years, while only 2% reported a decrease [42]. Educational institutions have responded to this issue in a number of ways; common techniques include offering information sessions, modifying assignments to resist plagiarism, and employing formative assessment [62]. Addressing the issue is also important for academic venues (e.g., conferences and journals): malicious authors sometimes willfully disregard rules preventing duplicate submissions, or submit slightly modified work created by others.

In order to make use of some preventative techniques, or to apply punishments for violations, plagiarism must first be detected. Many useful tools have been developed to detect plagiarism of software source code and textual documents [1, 33]. While some of these tools are meant to be run locally, it is beneficial to centralize plagiarism detection. State-of-the-art plagiarism detection relies on matching similarities between documents, and thus the effectiveness of a detection tool is limited by the size and characteristics of its corpus. A centralized solution has the advantage of having visibility into documents belonging to every participant in the network, whereas a locally executed tool would not be able to detect plagiarism across venue boundaries. One of the most popular centralized services is developed by Turnitin; this proprietary system serves over 15,000 institutions covering 30 million students [63].

Unfortunately, centralizing plagiarism detection introduces privacy and legal concerns. Documents being checked for plagiarism may contain confidential information (e.g., unpublished product details) that must be revealed to the central authority in these schemes. Moreover, aggregating a large amount of private information at a single source increases its value as a target for security compromise. A centralized service also introduces legal issues; authors may be forced to grant the central service a license to use their intellectual property [16]. In many situations, academic venues may not have the necessary permissions under copyright law to share submissions with central authorities.

These desires lead academic venues to confront seemingly conflicting objectives: they would like to share information for the purpose of detecting plagiarism, but they simultaneously want to preserve the confidentiality of content under their control. In this work, we present Elxa[1], a system that achieves both of these objectives by performing large-scale plagiarism detection on textual content while also preserving the confidentiality of unplagiarized text. When a venue analyzes a suspicious document using Elxa, they are given a list of matching documents, the venues that submitted them, and the matching textual passages. This level of detail in the output is comparable to Turnitin, the most successful non-private system. Academic venues can use Elxa to privately detect duplicate submissions to other venues, or to locate previous submissions and their associated peer reviews. Our scheme includes the use of an untrusted third party that manages the network and controls the admittance of participants, but never learns the original text of any documents. The presence of this entity allows our system to resist attacks by malicious groups, and to provide a monetary incentive for real-world deployment. Specifically, we make the following novel contributions:

- We define a setting and objectives for scalable centralized plagiarism detection with privacy preservation.
- We introduce Elxa, the first protocol to offer privacy protections in the aforementioned setting.

* This is an extended version of our WPES'16 publication [64].

[1] Elxa is named after a fictional wizard, created by Patrick Rothfuss, whose magic relies on the similarity between objects.

- We describe how privacy-enhancing technologies can be applied to a class of state-of-the-art plagiarism detection techniques from the information retrieval field.
- We evaluate a proof-of-concept implementation of our scheme using an industry-standard big data framework to demonstrate that our technique is scalable and deployable on commoditized cloud infrastructure.

The remainder of this paper is organized as follows: we survey related work and its shortcomings in §2; we describe an overview of our design goals and system architecture in §3; we discuss our approach and prototype implementation for finding matches in §4, and for confirming matches in §5; we consider security threats and mitigations in §6; we evaluate Elxa's performance in §7; finally, we offer our concluding thoughts in §8.

2. RELATED WORK

Related work can be classified into two areas: (non-private) plagiarism detection and privacy-preserving information sharing. We begin by surveying the state of the art in plagiarism detection, followed by related cryptographic primitives and protocols.

2.1 Plagiarism Detection

For seven years, an annual international contest of plagiarism detection techniques has been held as part of PAN, a series of evaluation labs at the CLEF conference [40]. Every year, the contest provides training and testing corpora, and solicits submission of software solutions that are compared against all other past and contemporary systems. The original plagiarism detection contest at PAN introduced the first standardized suite of performance metrics [50]. Ever since then, PAN has continued to update its contest as techniques and metrics in the field change. For these reasons, the top-performing systems presented at PAN represent the state of the art of publicly available plagiarism detection methodologies.

Plagiarism detection strategies can be divided into two main categories: intrinsic detection, and external detection [50]. Intrinsic detection involves examining the text of a suspicious document, and attempting to locate sections of text that are written with a significantly different style. Unfortunately, this approach suffers from a number of insurmountable flaws: minor obfuscation is sufficient to defeat it, and there is no possibility of detecting wholly plagiarized documents or duplicate documents submitted to multiple entities. In contrast, external detection involves comparing a suspicious document to a corpus in order to identify text reuse such as copies, translations, paraphrases, and summarization [45].

In recent years, the PAN contest has recognized the importance of centralized corpora by no longer considering intrinsic detection to be a meritorious approach. All submissions to PAN now follow the two-stage strategy pioneered by Stein et al. [60]: a suspicious document is used to search for potential source documents, and then each source document is compared with the suspicious document to identify matching passages. The two stages are called *source retrieval* and *text alignment*, respectively.

PAN submissions are designed to make queries to search engines for source retrieval, which is a reasonable strategy when privacy is not a concern. For the PAN contest, search queries are made against the Clueweb09 data set, which contains a 25 TiB snapshot of approximately 1 billion web pages collected during 2009 [61]. The goal of source retrieval algorithms is to locate as many actual source documents as possible while minimizing the number of search queries issued and documents downloaded. Given potential matches, the text alignment algorithms analyze the texts and attempt to find reuse.

There is an orthogonal field of research examining plagiarism detection for source code [33]. These domain-specific algorithms are much more efficient than simply applying text-based methods to the sources. In this work, we focus on textual plagiarism detection; we leave privacy-preserving source code plagiarism detection as an avenue for future work.

2.2 Private Data Sharing

Many cryptographic protocols are designed to share data with some sort of privacy guarantees, but none fits well with the plagiarism detection model. *Private information retrieval* (PIR) protocols allow clients to query data from a database server without revealing the query [9]. However, the *contents* of a PIR server's database are not hidden. A related technology is *oblivious RAM* (ORAM), which allows a client to store secret information on a server without revealing its access pattern [20, 37]. ORAM protects the privacy of access to a database, but is typically limited to one user.

Searchable encryption allows searches to be performed on ciphertexts [8], but requires that data owners explicitly authorize others to search their ciphertexts. For scalability, we must avoid pairwise access control requirements. *Secure sketches* allow a user to use a key to encrypt data that is unlocked only when a similar key is presented [30]. Unfortunately, secure sketches reveal the plaintext of the document when a match is detected. *Fuzzy Hashing* schemes produce hashes of data that match when the preimages are similar (rather than only when they are identical) [53]. Unlike the primitives used by Elxa, fuzzy hashes operate on opaque data structures, and thus do not take advantage of domain-specific features. *Secure multi-party computation* (SMC) protocols allow multiple parties to interactively compute arbitrary functions without revealing their inputs [17]. We use a domain-specific SMC protocol in Elxa that achieves much better performance than generic constructions. Other related primitives include *garbled circuits*, *fully homomorphic encryption*, and *secure co-processors*. While these techniques are often used to construct SMC protocols, they remain either too resource-intensive or are incompatible with Elxa's threat model.

Privacy-preserving record linkage protocols (PPRLs) are closely related to plagiarism detection [10]. PPRLs attempt to find duplicate entries in confidential distributed databases stored by mutually distrustful parties, such as patient health records from multiple hospitals. Vatsalan et al. [65] performed a comprehensive survey of dozens of PPRLs. While several PPRLs make use of techniques that are similar to our approach [18, 41, 55], none is well suited for our setting. PPRLs are designed to operate on standard relational databases, where each record consists of multiple structured fields that are typically short and noisy. PPRLs also normally include undesirable requirements such as pairwise interaction between parties, the existence of a trusted third party, or small and similarly sized databases [65]. In contrast, Elxa is specifically designed to handle a party with a small database looking for matches in a large distributed database, very long unstructured textual records, an untrusted central server, and no large-scale pairwise interactions between network participants.

3. GOALS AND ARCHITECTURE

Our main objective for this work was to design a usable system that can provide privacy benefits to real users. In this section, we describe our novel design goals and system architecture.

3.1 Participants and Capabilities

An Elxa deployment consists of multiple *clients* (e.g., academic venues) and a central server (the *root*). Each client operates a server called a *node* that is always connected to the root. We assume

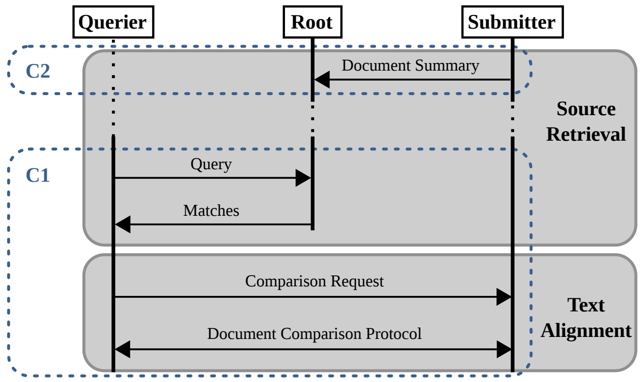

Figure 1: This sequence diagram shows a sample Elxa interaction. A node submits a document to the root (C2). Later, another node performs a search query, finds a potential source, and contacts the submitter to perform a full comparison (C1).

that the root and the nodes all have access to significant computational resources and storage. Collectively, the root and the nodes are called the *network*. The network provides two capabilities:

- **C1: Document Searches.** A node can search the network for *source* document fragments that match fragments of a *suspicious* document. We refer to a node performing this operation as a *querier*.
- **C2: Document Submission.** A node can contribute a document to the network's corpus. We refer to this node as the document's *submitter*.

Both **C1** and **C2** can be performed with some privacy protection, which we discuss in §3.2. Searching for matching documents as part of **C1** consists of two phases: potential source documents are identified using summaries stored by the root (source retrieval), and then the source documents are compared to the suspicious document through communication between the querier and the source documents' submitters (text alignment).

Elxa's high-level operation takes place in three phases:

1. Authors submit their work to a participating venue.
2. The venue's node queries the root using **C1** to find the potential sources of each document. After the root responds, the querier contacts other nodes to confirm plagiarism.
3. The venue's node submits summaries of the documents to the root using **C2**.

Figure 1 shows an example illustrating the relationship between the capabilities and the phases of operation.

3.2 Design Goals

Our design goals for Elxa can be divided into two broad categories: functionality and usability.

Functionality goals:

- **G1: Incrimination.** When two parties identify that their documents contain similar fragments, the text of the matching fragments is revealed to the two parties.
- **G2: Effectiveness.** The system should achieve **G1** with effectiveness similar to state-of-the-art non-private plagiarism detection schemes, when measured using standard metrics.
- **G3: Confidentiality Against the Root.** The root cannot reliably construct any of the original text for any document submitted by an honest node. However, the root is permitted to learn metadata about documents, potentially including their subject matter and information about the author's writing style. We precisely describe the metadata that is allowed to leak in §6.
- **G4: Node Accountability.** Nodes that overtly misbehave without colluding with the root can be ejected from the network. Malicious nodes cannot significantly equivocate the contents of a document to the root and to other nodes without detection.
- **G5: Confidentiality Against Nodes.** Honest-but-curious (HbC) nodes cannot reliably construct any of the original text possessed by other nodes, with the exception of incriminating text covered by **G1**. HbC nodes are permitted to learn the document metadata that the root knows, in addition to extra information about the similarity between documents owned by communication partners. We precisely describe the information that is allowed to leak in §6. Our use of the HbC threat model is consistent with most work in the PPRL field [65], although we also achieve some protection against malicious nodes due to **G4**.
- **G6: Author Accountability.** With the cooperation of the submitter, a document can be traced back to the original author.
- **G7: Sybil Attack Prevention.** Sybil attacks [14] are discouraged by requiring verification before enrollment in the network.

Usability goals:

- **G8: Incentivization.** All participants should be incentivized to deploy the system.
- **G9: Deployability.** The system should be deployable on existing cloud infrastructure without non-collusion assumptions.
- **G10: Scalability.** The resource demands on the root scale linearly with the number of simultaneous searches—not the number of nodes. The number of text alignment sessions performed during a search scales with the number of similar documents in the network—not the total number of documents or nodes.

3.3 Architectural Overview

All communication channels in the network are secured and authenticated using TLS. We use client authentication with the root acting as the sole certificate authority to ensure that every node in the network must be approved by the root before enrolling, achieving **G7**. The root can charge a subscription fee for enrollment to fund network maintenance, incentivizing deployment (**G8**). Since nodes are manually approved, we assume that nodes exchange public keys with the root using an authenticated out-of-band channel.

Each node in the network has a submitter identifier that is associated with their TLS certificate. Each document is assigned a unique document identifier by its submitter. Therefore, each document is uniquely identified by its submitter and document identifiers. Submitters maintain a local database of venue-specific user identifiers (e.g., author usernames) for their submissions. These databases achieve **G6** by allowing documents to be traced to the authors.

Although we focus on the submission of private documents, Elxa can also trivially support the inclusion of public corpora. The root can add public documents (e.g., web pages) to the corpus using the same protocol as submitters and a specialized submitter identifier.

The remainder of this paper describes and evaluates the protocols that implement the capabilities **C1** and **C2** defined in §3.1, and how the other goals defined in §3.2 are met. We also describe details of our prototype implementation of Elxa. §4 describes the process for submitting documents to the root, and how the root locates potential matches for a querier (i.e., private source retrieval). §5 describes the interaction between two nodes attempting to confirm a case of plagiarism (i.e., private text alignment).

4. SOURCE RETRIEVAL

Elxa's source retrieval system incorporates ideas from one of the best non-private schemes, as determined by the information retrieval community. We first describe the non-private scheme, and then show how to privately perform source retrieval.

4.1 Non-Private Approach

4.1.1 *Query Generation*

The most recent PAN evaluation of source retrieval systems [23] identified the scheme proposed by Haggag and El-Beltagy [24], hereafter referred to as HEB13, as the best in terms of precision, total number of downloads, and number of downloads until a true source of plagiarism is identified. Compared to the other schemes, HEB13 generates very few search queries while also maintaining high precision. These attributes are desirable in our setting because minimizing the number of requests sent to the root is important for achieving our scalability goal (**G10**). HEB13 makes use of a multi-stage query generation pipeline consisting of three phases: preprocessing, keyphrase extraction, and query generation.

In the first phase, non-English letters are removed and the document is split into words by tokenizing on whitespace. Next, the document's word frequency distribution is computed.

In the second phase, the document is split into segments using the TextTiling algorithm [25]. Each segment corresponds to a set of topically related paragraphs. The KP-Miner algorithm is used to extract a keyphrase—a short set of one to three words that is "characteristic" of the paragraph—for each segment [4]. The segments are split into sentences using the Punkt sentence tokenizer [28] as implemented in the Natural Language Toolkit [7]. These sentences are combined into chunks of four. For each chunk, the words are sorted by ascending document frequency. The rarest words from each chunk are concatenated with the keyphrase for the segment, up to a maximum of 10 words, forming the query for that chunk. *Stop words*—words that are a necessary part of the language but that have little or no intrinsic semantic value—are ignored when constructing the queries.

In the final phase, the queries are submitted to the search engine, which returns a set of results. Each result is associated with a document identifier, and contains a snippet of the text surrounding the match. Snippets contain approximately 500 non-whitespace characters. If the snippet includes at least 50% of the query's keywords, then it is considered a match and the document is downloaded. Subsequent queries are checked against the previously downloaded documents. If any previously downloaded document contains at least 60% of a query's keywords, then the query is suppressed.

4.1.2 *Search Engine*

While HEB13 produces search queries that are relatively successful at locating plagiarism cases, the actual work of matching documents is performed by the search engine. Submissions to PAN are tested using the ChatNoir search engine [47], which indexes the ClueWeb09 database [61]. ChatNoir uses several weighted metrics to search for query matches [22]: BM25F [66], PageRank [38], SpamRank [5], and Proximity [15]. We focus on BM25F [66], the most important metric, for our design.

BM25F is a modification of BM25 [52], an extremely popular measurement used for information retrieval. BM25 scores the match of a query of n keywords, $Q = (q_1, q_2, \ldots, q_n)$, against a document D using the term frequency (TF) and inverse document frequency (IDF) of each keyword. For this reason, BM25 is known as a TF-IDF statistic. The term frequency $tf(q_i, D)$ of a keyword q_i in a document D denotes the number of times that q_i appears in D. The IDF $idf(q_i)$, which reduces the influence of keywords that appear frequently across many documents, is given by Robertson and Sparck Jones [51]:

$$idf(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \qquad (1)$$

where N is the number of documents in the collection, and $n(q_i)$ is the number of documents containing at least one occurrence of q_i. The BM25 score for a document is given by Robertson et al. [52]:

$$BM25(Q, D) = \sum_{i=1}^{n} idf(q_i) \cdot \frac{tf(q_i, D)}{K(D) + tf(q_i, D)} \qquad (2)$$

The support function $K(D)$ is defined as:

$$K(D) = k_1 \left((1 - b) + b \cdot \frac{dl(D)}{avdl} \right) \qquad (3)$$

where k_1 and b are system parameters, $dl(D)$ is the number of words in D, and $avdl$ is the mean number of words per document across the corpus. BM25F differs from BM25 by assigning weights to different parts of the document (e.g., headings and body text).

4.2 Privacy Preservation Overview

We now introduce our novel privacy-preserving source retrieval system. Unlike ChatNoir in the PAN setting, the root in our setting does not have access to the original text of the documents (**G3**). Instead, the root stores only the information necessary for evaluating a document's BM25 score. This makes it impossible for the root to generate document snippets in the traditional manner (i.e., by simply selecting a substring surrounding the matching keywords in a document). We overcome this problem by having document submitters break documents into non-overlapping snippets during submission (**C2**). §4.3 describes this procedure in detail.

Source retrieval also encompasses the first phase of the private searching process (**C1**). Elxa incorporates ideas from both HEB13 and the ChatNoir search engine in order to implement a scalable BM25-based search of the snippets in the root's database. §4.4 describes the details of our scheme.

4.3 Snippet Generation

When a node is ready to submit a document to the network (**C2**), it begins the process of snippet generation. The submitter stores details about the document source (e.g., the author username), the document identifier, and the raw document text in its local database. The document is then pre-processed in a manner similar to HEB13: non-English letters are removed, and non-stop words are tokenized. Next, the node partitions the document into chunks of word tokens, where the total length of the tokens within each chunk is as close as possible to, but does not exceed, 500 characters. Words are never split to fit within the character limit. The next step is to transform these chunks of tokens into snippets.

Our snippet generation procedure involves the use of a data structure known as a *count-min sketch* (CMS). A CMS is a probabilistic data structure that stores an approximate representation of a multiset (e.g., a frequency table). CMSes were introduced by Muthukrishnan and Cormode [11], and are related to Bloom filters (which operate on sets rather than multisets). A CMS contains a two-dimensional table with a pre-defined depth and width. Each row in the table is associated with a pairwise-independent hash function. When a value is inserted into the structure, the hash for each row determines the associated column. The frequency of the value is then added to these cells. To query for a frequency, the same hashes are performed, and the output is the minimum value in the resulting cells. A CMS will never underestimate, but may overestimate, an event's frequency.

Once the document has been split into chunks, the submitter computes the word frequency distribution of each chunk and stores the distributions in CMSes (one per chunk). Words are stemmed using the Porter word stemmer [44] and then processed by a collision-resistant hash function with a globally shared salt before being added to the CMS. Each CMS represents one document snippet.

The snippets are then shuffled and sent to the root alongside the document identifier. Appendix A includes an example of a snippet.

4.4 Private Search Queries

While HEB13 is designed to use a public general-purpose web search engine, Elxa interacts with a custom server storing snippets encoded as CMSes. The key insight of this approach is that storing snippet frequency tables in a CMS preserves the information necessary to compute BM25 scores, but recovering the original text from such a CMS is believed to be difficult (see §6).

To begin a search, the querier stems and hashes the query keywords with the same method used for snippet generation. The server then scores each snippet in the database against the query using BM25, as defined in Equation 2. Our server does not use the BM25F, PageRank, SpamRank, or Proximity metrics used by ChatNoir, since these approaches are inapplicable to the setting or rely on information that is discarded for privacy reasons.

The term frequency for BM25 is computed on a per-snippet basis: $tf(q_i, S)$ for a keyword q_i and a snippet S is computed by querying the CMS associated with S for the frequency of q_i. Computing K is slightly more difficult. In this context, $dl(S)$ can be found by computing the sum of any row in the CMS, since this indicates the number of events (i.e., words in the frequency table). Likewise, $avgdl$ is the mean value of $dl(s)$ across every s in the database. The idf for each keyword is computed on a per-document basis, and can be found by querying the CMSes in a similar manner.

HEB13 requires that clients only download documents where at least 50% of the query keywords appear in the snippet. Since our server is designed to exclusively support Elxa rather than acting as a general-purpose search engine, we implement this logic in the server. Specifically, any snippets that do not meet this criterion are given a negative BM25 score, causing them to be discarded. Once the server has ranked matching snippets by their score, it sends information about the top five associated documents to the node. Each result includes the document's submitter identifier, document identifier, and constituent snippets. The identifiers allow the querier to contact the submitters to perform the text alignment phase of **C1**, while the document snippets allow it to suppress redundant queries in the same way as HEB13.

Since any big data platform can perform these computations, Elxa achieves our deployability (**G9**) and scalability (**G10**) goals.

4.5 Implementation

To demonstrate the practicality of our approach, we developed a prototype implementation of Elxa's source retrieval system for the Apache Hadoop MapReduce platform [2]. Since Hadoop is widely available in commoditized cloud infrastructure, using it as a platform for our prototype demonstrates Elxa's deployability (**G9**). Hadoop MapReduce jobs include two stages: mappers, which read input key/value pairs and produce intermediate key/value pairs; and reducers, which read all intermediate values with the same key and produce output key/value pairs. We will not describe the details of MapReduce here; we refer the interested reader to the original MapReduce paper for more information [12]. Our root server listens on a TCP port using TLS with mandatory client authentication, and communicates with nodes using custom Google Protocol Buffer structures [21]. This binary encoding is extremely efficient, minimizing the amount of network traffic used for transmitting CMSes. We developed a node implementation based on the original HEB13 Python code.

The client nodes are expected to perform document submission regularly. Each node reads the set of unsubmitted documents, generates the snippets, and submits them to the root. Our prototype uses the MurmurHash3 non-cryptographic hash function [3] to implement the CMSes. All documents submitted in a single session are stored within a single Hadoop sequence file—an efficient binary format storing key/value pairs—in the Hadoop distributed filesystem. Each input key in the sequence file is a randomly selected identifier, and the input values are structures consisting of one or more snippets. Since each snippet is stored with a submitter and document identifier, sequence files can be merged without conflicts. This capability allows the root to optimize performance based on the Hadoop block size.

When a query is submitted to the root, it initiates two MapReduce jobs. The first job is responsible for counting the number of documents in the collection, the number of words in the collection, the number of snippets in the collection, and the number of documents containing each keyword in the query. These values enable the root to compute N and $idf(q_i)$ for each keyword q_i (see Equation 1), as well as $avdl$ (see Equation 3). We set the system parameters to the standard values of $k_1 = 2$ and $b = 0.75$ [31, p.233].

The second MapReduce job uses the counts computed by the first job to produce search results. The mapper computes the BM25 scores for all snippets contained in a sequence file and outputs a set of document identifiers, and their associated snippets, sorted by decreasing score. All output keys are set to the same value, ensuring that all data is sent to a single reducer. The reducer merges the list of matches, keeping them sorted by BM25 score, and outputs the five documents with the highest scores. We use the same reducer implementation for the optional combiner stage in order to achieve negligible bandwidth requirements (i.e., each node in the cluster sends five or fewer documents to the reducer for each job).

The counting job is not strictly necessary in this process, especially as the database scales up—in a real large-scale deployment, approximate IDF values would be sufficient. Instead of performing actual document counts for the IDF, the root could simply refer to a static frequency table. Any words that are not found in the frequency table could be assumed to have some base frequency. By eliminating the counting job, each query can be performed approximately twice as quickly.

5. TEXT ALIGNMENT

After completing the source retrieval process, the querier has a set of potential source documents. The next step of the search procedure (**C1**) is to contact the submitters of the source documents in order to identify and confirm matches with the suspicious document. This is the role of the text alignment protocol.

5.1 Non-Private Approach

Before describing our privacy-preserving text alignment scheme, we first examine one of the best non-private schemes [46]: the system designed by Sanchez-Perez et al. [56], hereafter referred to as SP14. Since Elxa shares many components with SP14, we now describe the approach in detail. SP14 involves four phases:

1. Preprocessing: The texts are split into sentences and words.
2. Seeding: All pairs of sentences from the documents are compared to find similar sentence pairs known as *seeds*.
3. Extension: The seeds are extended into pairs of similar document excerpts called *fragments*. The fragment pairs are candidate plagiarism cases.
4. Filtering: The candidates are filtered to reduce false positives.

SP14 offers one set of system parameters that is tuned to detect summarization, and another set that is better suited to detecting other forms of text reuse. Since it is not possible to know a priori what type of plagiarism has occurred, SP14 dynamically selects the best parameter set during execution.

Preprocessing: To begin, the document is split into sentences using the same Punkt sentence tokenizer used by HEB13 [28]. Each sentence is broken into word tokens using the Treebank word tokenizer [32]. Non-English words and stop words are discarded. All words are converted to lower case and passed through the Porter word stemmer [44]. The last stage of preprocessing repeatedly merges sentences with fewer than four words (after filtering).

Seeding: The purpose of the seeding phase is to identify many candidate text reuse cases. A seed is a pair of sentences—one from each document. To locate seeds, every possible pair of sentences is compared to assess their similarity.

Before comparing the sentences, they must first be expressed in terms of a vector space model [54]. Conceptually, each sentence is transformed into a vector where the frequency of every possible word is given by the value of an associated component. These frequencies are computed using a variant of the TF-IDF statistic [59] known as TF-ISF (term frequency—inverse sentence frequency). The TF-ISF value for a word token t in a sentence s is given by:

$$tf\text{-}isf(t,s) = tf(t,s) \times \log \frac{|S|}{|\{s' \in S : t \in s'\}|} \tag{4}$$

where S is the set of sentences in the two documents, and $tf(t,s)$ denotes the number of times that t appears in s.

Once the sentences from the source and suspicious documents have been converted into vectors, the similarity of the vectors is computed. SP14 uses two different metrics to evaluate similarity: the cosine and Sørensen-Dice [13, 58] metrics. If s_1 (resp., s_2) is the vector corresponding to the sentence from the suspicious (resp., source) document, then the cosine similarity is given by:

$$cos(s_1,s_2) = \frac{s_1 \bullet s_2}{|s_1||s_2|} \tag{5}$$

where $\bullet$ denotes the dot product, and $|s|$ denotes the Euclidean norm of s. If b_1 (resp., b_2) is the set of words appearing at least once in the suspicious (resp., source) sentence, then the Sørensen-Dice similarity is given by:

$$Dice(b_1,b_2) = \frac{2|b_1 \cap b_2|}{|b_1|+|b_2|} \tag{6}$$

If the cosine similarity of a sentence pair exceeds $th1$ and the Sørensen-Dice similarity also exceeds $th2$, where $th1$ and $th2$ are system parameters, then the pair is accepted as a seed.

Fragment extension and filtering: During the next two phases of SP14, the seeds, each denoting a pair of similar sentences, are extended into pairs of similar document fragments. A document fragment is a contiguous sequence of sentences from one of the documents. The output of these phases is a set of pairs of similar document fragments (i.e., plagiarism cases). The similarity of two fragments F_1 and F_2 is computed as the cosine similarity of the sum of the corresponding sentence vectors:

$$sim(F_1,F_2) = cos(\Sigma_{s_1 \in F_1} s_1,\ \Sigma_{s_2 \in F_2} s_2) \tag{7}$$

SP14 reports only non-overlapping plagiarism cases exceeding a minimum length. Fragments in Elxa are formed in precisely the same way as in SP14; we refer the interested reader to the SP14 paper for details [56].

5.2 Privacy Preservation

We now introduce Elxa's mechanism for transforming SP14 into an interactive privacy-preserving protocol. More generally, our approach can derive private variants of text alignment schemes that compare documents with the cosine and Sørensen-Dice metrics.

In the non-private setting, SP14 is given full access to both the suspicious and source document, and the algorithm is executed by a single party. In our setting, the algorithm is executed jointly between two HbC parties, and each party has access to only one of the documents. Since the behavior of SP14 is deterministic for given input documents, the core approach is to have both parties execute the algorithm independently. Wherever the non-private algorithm performs a computation accessing both texts, the private protocol interactively performs an equivalent private joint computation.

We begin by discussing a *non-private* joint computation of SP14, and then describe how to construct a privacy-preserving variant. From the perspective of one party, the following operations require knowledge about the other document:

1. The number of sentences in the other document must be known. This information is needed for several reasons: the seeding phase requires a pairwise measurement of sentence similarities, computing the TF-ISF measure given in Equation 4 uses the number of sentences to compute the inverse sentence frequencies, and the document bounds are used for fragment extension and filtering.
2. The order of sentences in the other document must be known. During the extension phase, the initial seeds are extended to form document fragments. These fragments consist of contiguous sentences, and the formation of fragments is based on the location of the seeds.
3. The number of sentences in the other document that contain a given word must be known in order to compute the word's ISF.
4. The vector space model representation of each sentence in the other document must be known in order to compute the similarity between pairs of sentences and document fragments.

No other information must be shared to complete SP14.

Consider a "straw man" privacy preservation protocol in which the two parties, Alice and Bob, simply reveal the aforementioned information in order to jointly compute the output of SP14. This approach fails to achieve our confidentiality goal (**G5**), because it reveals almost everything about the documents. Alice knows the TF-ISF vector associated with every sentence of Bob's document, and she knows their order. Since she also knows enough information to compute the ISF of every word token t, she can solve Equation 4 for the term frequency $tf(t,s)$. Now Alice knows a raw "bag of words" representation of each sentence in Bob's document, and Bob can learn the same information about Alice's document.
We can provide much better privacy by making a key observation: since SP14 uses the TF-ISF vectors only to compute the cosine and Sørensen-Dice metrics, they can perform a secure multi-party computation of Equations 5 and 6 without revealing their own vectors. Now, HbC parties can no longer derive the bag of words corresponding to sentences in the other document. Consequently, the only information that is revealed is the degree of similarity between sentences and the position of those sentences within the documents—precisely what is needed to detect text reuse.

Elxa's text alignment protocol is interactively performed by a client—the node with the suspicious document—and a server—the node with the source document. The parties first exchange the number of sentences in their documents, and then jointly enumerate the sentence pairs in a specified order. For each pair, they execute a secure multi-party computation protocol that we call *priv-sim*, with each party providing its sentence as input. *priv-sim* outputs the cosine and Sørensen-Dice metrics of the two sentences. These values are compared to the threshold parameters $th1$ and $th2$ as in SP14. After the seeding phase, the parties perform fragment extension and filtering in the same way as in SP14, except that *priv-sim* is used to compute fragment similarities.

5.3 Private Similarity Computation

The core of Elxa's text alignment scheme is the *priv-sim* protocol, which computes the cosine and Sørensen-Dice similarities of two private sentences as follows:

1. Both parties reveal the number of unique words in their sentence.
2. The parties jointly compute the set of words appearing in both sentences—we call these the *shared words*.
3. If they haven't done so already, both parties reveal the sentence frequency of each shared word.
4. The parties compute the TF-ISF of their words.
5. The parties jointly compute the cosine similarity.
6. The parties compute the Sørensen-Dice similarity.

After exchanging their word counts in step 1, the parties compute the shared words in step 2. If we view each sentence as a set of unique word tokens, then the problem of privately computing the shared words is exactly the *private set intersection* (PSI) problem. PSI protocols are common in the cryptographic literature [43]; they allow parties to jointly compute the intersection of their private sets without revealing any information about those sets.

For our application, we make use of the PSI protocol introduced by Huberman, Franklin, and Hogg [26], hereafter called HFH99. The HFH99 protocol is itself based on an older scheme proposed by Meadows [34]. HFH99 operates in the same setting as Diffie-Hellman key exchanges. Each party converts their word tokens into group elements, randomly permutes the order, and raises each element to a secret exponent. Each party transmits the resulting sequences, raises the values they receive to their secret exponent, and transmits the final result. Because exponentiations are commutative, equal group elements from each set will be raised to the same exponent. Each party can identify these shared elements and, since they know their own secret permutation, they can identify the original word tokens.

Once the shared words have been identified using HFH99, the parties compute the TF-ISF values of their words. This presents a challenge: the inverse sentence frequency term in Equation 4 requires knowledge of the number of sentences (across both documents) in which the associated word appears. Each party knows this sentence frequency for their own document, but not for the other party's document. One possible way to handle this problem would be to have the parties reveal all sentence frequencies to each other at the beginning of the protocol. However, this reveals a lot of information, such as the complete list of all words that appear in the document. Instead, we accept a compromise. In step 3, the parties exchange exact sentence frequencies for only the shared words. The values are cached to minimize communication. In step 4, the TF-ISF value for each term t in sentence s is then computed as:

$$tf\text{-}isf(t,s) = tf(t,s) \times \log \frac{|D| + |D'|}{sf(t,D) + sf(t,D')}$$

where D is the document known to the party, D' is the other party's document, each document is a set of sentences, and $sf(t,D)$ denotes the frequency of the term t in the sentences of document D. For shared words, the value of $sf(t,D')$ was explicitly transmitted in step 3. For non-shared words, the value is estimated as:

$$sf(t,D') = sf(t,D) \cdot \left(1 + \frac{|D'|}{|D|}\right) \quad (8)$$

This optimistic estimation assumes that the sentence frequency of non-shared words is equal in both documents.

Next, the parties want to privately compute the cosine similarity metric depicted in Equation 5. In §5.1, we suggested that each input vector for the cosine similarity metric had a component associated with every possible input word. Of course, this does not work well in practice: there is no dictionary of every possible word that will ever appear in documents submitted to the system. Luckily, the metric only requires computation of the dot product and the Euclidean norm of the vectors—both of these measures are unaffected by zero entries. The Euclidean norm can be computed privately by each party on the $tf\text{-}isf$ values for all words in their sentence. To compute the dot product, the parties need to construct vectors with the $tf\text{-}isf$ values for only the shared words. They can sort the words to ensure a common ordering for the vector entries.

Given the input vectors associated with shared words, the two parties perform secure multi-party computation of the cosine similarity metric. We adapt the technique introduced by Jiang et al. [27, 36]. The strategy is to divide each $tf\text{-}isf$ value by the Euclidean norm, and then perform a private dot product computation. We can compute the dot product by using a cryptosystem that supports homomorphic addition of plaintexts and homomorphic multiplication by a scalar.[2] Like Jiang et al., we use the Paillier cryptosystem [39]. Given n-dimensional input vectors v^A and v^B known to only Alice and Bob, respectively, the protocol proceeds as follows:

1. Bob generates a Paillier key pair $(pk, pr) \leftarrow KeyGen()$ and sends public key pk to Alice.
2. Alice sends $Enc_{pk}(v_1^A), \ldots, Enc_{pk}(v_n^A)$ to Bob.
3. Bob homomorphically computes $Enc_{pk}(v_1^B \times v_1^A), \ldots, Enc_{pk}(v_n^B \times v_n^A)$.
4. Bob homomorphically computes $Enc_{pk}(v_1^B \times v_1^A + \cdots + v_n^B \times v_n^A)$ and sends it to Alice.
5. Alice decrypts the dot product $v_1^B \times v_1^A + \cdots + v_n^B \times v_n^A$ and sends it to Bob.

After running this protocol on their input vectors for their common words, the two parties have computed the cosine similarity metric.

The last value to compute as part of *priv-sim* is the Sørensen-Dice similarity metric defined in Equation 6. This is trivial: the formula depends only on the number of unique words in each sentence, and the number of shared words. Since these values are already known to both parties, the metric can be computed directly.

5.4 Implementation

We developed proof-of-concept client and server implementations of Elxa's text alignment protocol in the Go programming language [19]. When a client connects to the server, it begins by sending the document identifier of the source document. The server uses this identifier to look up the document's text in its local database, and responds with a freshly generated Paillier public key. The text alignment protocol is then executed. In a real deployment of Elxa, each node would run this text alignment server. The root would distribute connection information for each node so that a querier could connect to the submitters of potential source documents.

An interesting but orthogonal question is what to do when plagiarism is detected. The output of the text alignment protocol is a set of matching document fragment pairs. The client and server might notify their respective administrators to ask for approval before acting. In our prototype, the parties automatically exchange and display matching plaintexts.

6. SECURITY ANALYSIS

Elxa's source retrieval approach, described in §4, provides significant privacy benefits compared to HEB13, and provides substantial confidentiality against the root (**G3**). However, it necessarily leaks some information; the root is able to learn the number of documents submitted by each node, the approximate size of each document, and the timing and source node of submissions. If the root performs a dictionary attack, constructing a mapping between words and their hashed values, then it can learn the approximate frequency of non-stop words in each snippet. However, the hash protects all words that do not appear in the root's dictionary, such

[2]We encode the rational numbers as integer multiples of a tiny value. The precision loss is inconsequential.

as unique names of products. We stress that the root **cannot** learn the following information:

- the order of the snippets within each document;
- the order of words within each snippet;
- the plaintext of very rare and previously unseen words; and
- the position or frequency of stop words.

This limited information is enough to perform some information retrieval operations, such as BM25, but falls far short of being able to reconstruct the original document (**G3**), even with extrinsic information like language frequency distributions and the topic of documents. Appendix B includes an example of the data that the root can collect by performing a dictionary attack.

Neither HEB13 nor Elxa are explicitly designed to find highly obfuscated plagiarism. While modifications to the protocols could be made to improve performance for obfuscated documents, most plagiarists used little or no obfuscation when constructing the PAN evaluation corpus [24]. For applications where most instances of plagiarism are egregious, more obfuscation-resistant algorithms, which typically have higher false positive rates, are unnecessary.

Of course, the root is able to attack the availability of the network by shutting down the database or by refusing to accept valid connections. We consider such availability attacks to be outside the scope of our threat model, especially since the server has a monetary incentive to continue operating (**G8**).

Our use of TLS with mandatory client authentication serves to prevent abuse of the network by malicious nodes (**G4**). Any unauthorized party, such as a malicious author or an advertiser, cannot use the search (**C1**) or submit (**C2**) capabilities. An authorized client acting maliciously, such as a node attempting a denial-of-service attack on the network, or a node that queries the database without contributing to the corpus, can be easily blocked.

Elxa's text alignment protocol, described in §5, is secure against HbC attackers (**G5**). In addition to plagiarized texts, nodes learn the following non-incriminating data about the other party's document:

- the number of sentences;
- the location of similarity matches;
- the similarity scores of every pair of sentences;
- the shared words for every pair of sentences; and
- the sentence frequency of all shared words.

A malicious node can cause the other party to reveal the entire plaintext of their document during the text alignment protocol. The HFH99 protocol is secure only in the HbC setting, so a malicious party can force all words in the sentence to match. A malicious party can also set their inputs to the private dot product protocol to be extremely large values, causing matches between all sentence pairs. The result of these attacks is that the entire documents will appear to match, and thus the full plaintexts will be revealed. A malicious party can also choose to withhold their matches. One possible defense against overtly malicious nodes is to compare the information received during the text alignment protocol to the associated snippets submitted to the root. If a party manipulates their protocol inputs to maliciously reveal plaintext without the cooperation of the root, then the malicious behavior may be detected by querying the CMSes. Additional accountability can be achieved by requiring digital signatures for text alignment messages; an honest participant could then submit a non-repudiable transcript of misbehavior to the root, which could expel the malicious party (**G4**).

Defending against malicious attackers in the text alignment protocol is very difficult. Simply replacing the Paillier and HFH99 components with sub-protocols that are secure in the malicious setting is insufficient; malicious parties could still potentially cause information leaks by behaving incorrectly in the other phases of SP14 (e.g., fragment extension). A full defense may require the use of extremely expensive approaches such as performing verified computation with SNARKs [6], or performing the entire SP14 protocol with secure multi-party computation. We stress that even a text alignment protocol secure against malicious attackers would not prevent participants from fabricating documents; malicious parties would still be able to perform attacks at this higher protocol layer. For example, a malicious submitter could fabricate documents with sensitive sentences to identify sensitive submissions; without a trusted party to certify document authenticity, this is an extremely difficult challenge to overcome. Instead of making major performance or deployability concessions, Elxa uses accountability (**G4**) to restrict malicious nodes to covert actions.

7. EVALUATION

We performed several experiments to determine if our prototype implementation of Elxa meets our objectives defined in §3.2. In this section, we investigate Elxa's effectiveness (**G1**, **G2**) and scalability (**G10**) by addressing the following questions:

1. What is a good choice for the size of the CMSes?
2. How effective is Elxa's source retrieval scheme?
3. How many queries are needed to search the network?
4. What is the effectiveness and performance of Elxa's text alignment protocol?

7.1 Count-Min Sketch Size

One of the challenging aspects of deploying the source retrieval system described in §4 is choosing an appropriate size for the count-min sketches. Each document snippet consists of at most 500 non-whitespace characters, so each snippet will contain approximately 100 English words.[3] For a given number of events, the depth and width of the CMS determine the false positive rate and the upper bound for overestimations. In our application, we must carefully select the CMS size to balance query accuracy, the amount of information revealed to the root, and the storage and bandwidth costs.

We tested a variety of CMS configurations under simulated conditions. We began by acquiring frequency distributions for 794,767 words in written British English, as sampled from the British National Corpus [29]. We removed non-English words and the 319 English stop words included in the NLTK corpus [7], and used the resulting distribution to generate random words.

We selected four cell counts: 750, 1000, 1500, and 2000. For each cell count, we varied the depth of the CMS between 1 and 40 and computed the corresponding width. We ran 300 experiments for each size configuration. We began each experiment by inserting 100 random words into the CMS. We then queried the CMS for the frequency of all dictionary words. For each word, we recorded the error (i.e., the difference between the reported and true frequency).

Figure 2 shows the mean probability of a non-zero error for a word. There is a clear optimal size configuration minimizing the probability of overestimation for a given cell count. This value, for a word chosen uniformly at random, is significant because a dictionary attack on a document snippet would involve querying the CMS for every possible word. Figure 3 shows the mean overestimation for queries that resulted in non-zero errors. The figure confirms that the mean overestimation is very low for the values tested, except for extreme configurations.

Given these results, we selected a CMS depth of 11 and width of 182—a total of 2002 cells—for our implementation. At this size, we expect that 0.010% of words would yield overestimations with a mean error of 1.2. With a dictionary of 500,000 words, a dictio-

[3]A well-known information retrieval heuristic is that the average English word contains approximately 5 characters [35, 57].

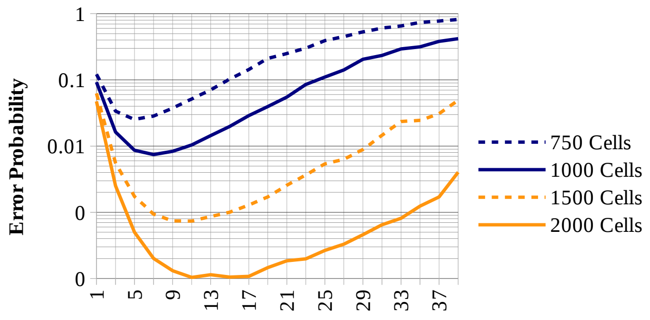

Figure 2: Probability (log scale) of overestimating the frequency of a uniformly random word using the CMS.

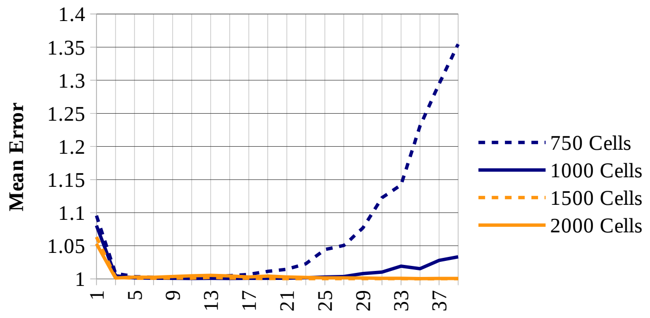

Figure 3: The mean overestimation of the CMS for words with non-zero error.

nary attack launched against a typical CMS would produce 100 true positives and 50 false positives. Nearly all false positives would be for words that were not in the original text. While the attacker can use word frequencies to predict which values are false positives, it cannot do so with certainty. Appendix A contains an example of a CMS and discusses technicalities of selecting a CMS size. Appendix B shows examples of information that the root can learn by performing dictionary attacks.

7.2 Source Retrieval Performance

To evaluate the performance and scalability of our source retrieval implementation, we tested our prototype using artificially constructed document databases of varying sizes. We began by collecting 57 suspicious documents from the PAN plagiarism detection testing corpus [40]. These documents were created by crowd-sourcing plagiarism, and ground truth data is available [48]. For each suspicious document, we used the original HEB13 implementation to query ChatNoir. We downloaded the top 10 matching documents for each query, as well as the ground truth source documents for the testing set. This process produced 6392 documents that we call the *source corpus*.

Next, we deployed our prototype Elxa root server on a Hadoop cluster with six machines. Five machines acted as the HDFS data and YARN nodes, while the remaining machine acted as the resource manager, HDFS name node, and Elxa root. We designed our prototype root server to allow submissions and queries to multiple databases, allowing us to use the same server to simulate different experimental conditions in parallel.

We constructed 11 databases by processing documents with our snippet generator (**C2**, see §4.3). The first database contained only the source corpus. Each of the 10 subsequent databases successively added an additional 10,000 documents selected at random from the ClueWeb09 corpus, which we call *noise documents*.

Finally, we measured the performance of our implementation by searching for plagiarism with each suspicious document in each

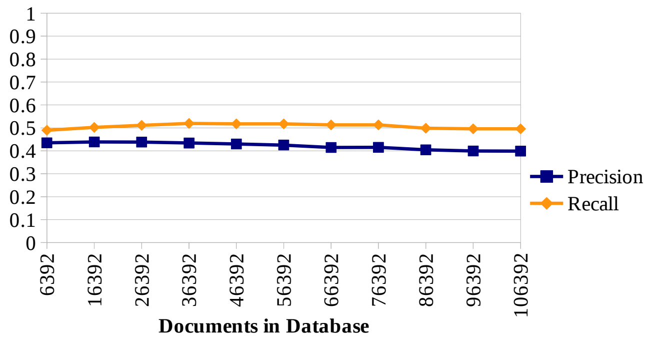

Figure 4: Elxa's effectiveness is stable as the database grows.

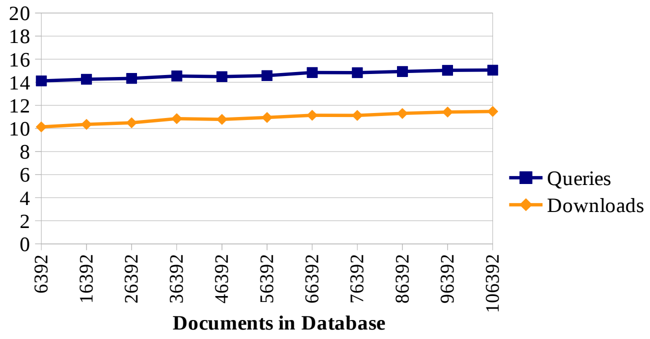

Figure 5: Elxa's workload scales well with database size.

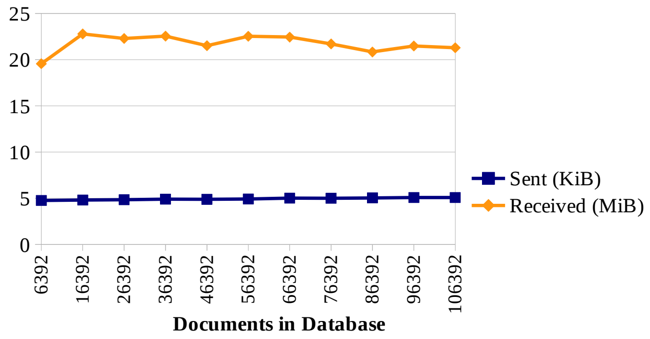

Figure 6: Elxa's per-document bandwidth requirements are stable, but participants require substantial downstream bandwidth (as they do for other PAN algorithms).

database. We ensured that the document identifiers of the source corpus were preserved so that we could compare the results to the ground truth values. We evaluated the results using the official performance metrics provided by PAN [40].

Figure 4 shows the performance of Elxa as the database size increases, in terms of precision and recall—the standard performance metrics used at PAN [23]. Precision is defined as the proportion of downloaded documents that are plagiarism sources or near duplicates of those sources. Recall is defined as the proportion of plagiarism sources that are identified during the search. Both the precision and recall remain stable (approximately 0.4 and 0.5, respectively) as the size of the database grows, even when 94% of the documents in the database are unrelated to the query. The precision and recall of HEB13 interacting with ChatNoir are 0.67 and 0.31, respectively [24]. While not directly comparable[4], these values show that Elxa performs well according to PAN's standards.

[4]Performing a direct comparison is challenging, since the algorithms are designed for use in very different settings and with search engines using different similarity metrics.

Table 1: Elxa's text alignment scheme performs competitively. Values are computed over 4,800 document pairs. PAN publications do not report standard deviations [46].

METRIC	ELXA	SP14
Precision	0.83	0.87
Recall	0.88	0.92
Granularity	1.00	1.00
Plagdet	0.85	0.89

Algorithms submitted to PAN are evaluated based on the number of queries and downloads that they perform. Figure 5 depicts the workload generated by Elxa as the database grows. For comparison, HEB13 generates a mean of 13.9 queries and 5.2 downloads when interacting with ChatNoir. Figure 5 shows that Elxa performs more downloads than HEB13, but only slightly more queries. These results reflect ChatNoir's better matching algorithm; Elxa is restricted to using only BM25 for private documents. Nonetheless, the workload generated by Elxa scales well.

Figure 6 depicts the average traffic generated for each document. The "sent" traffic flows from the submitter to the root (shown in KiB), and "received" traffic flows from the root to the submitter (shown in MiB). PAN publications do not include statistics on network traffic. However, queries incur smaller bandwidth costs for Elxa than for HEB13 since the Elxa root includes only matching documents in its results, whereas HEB13 must download document snippets from ChatNoir before deciding if the full documents should be downloaded. Therefore, "received" values are necessarily larger for HEB13 than for Elxa.

Our prototype server implementation takes a mean of 95 seconds (standard deviation 20 seconds) to search the source set. With the addition of 100,000 noise documents, the query time increases to 130 seconds (standard deviation 40 seconds). Implementing Elxa directly using the Hadoop MapReduce framework incurs a non-trivial minimal cost per query, and leads to this relatively poor query time scalability. Implementing Elxa on a big data computing platform specifically designed to perform real-time queries should dramatically improve the results. Nonetheless, even our unoptimized prototype achieves good performance for large databases. For comparison, HEB13 required over 46 hours to process 99 documents using ChatNoir [23]. We reiterate that Elxa deployments can double query performance by heuristically estimating IDF values instead of performing exact counts (see §4.5).

7.3 Text Alignment Performance

We evaluated the performance of Elxa's text alignment scheme using the PAN 2014 testing corpus #3—the largest and most recent publicly available testing set [40]. The corpus contains 4,800 pairs of suspicious and source documents with 1,600 instances of verbatim text reuse and 1,600 instances involving obfuscation.

We ran our client and server prototypes on a single machine and executed the private text alignment protocol for every document pair. We used the official PAN performance analysis tool to compute the precision, recall, granularity, and plagdet score [49]. Granularity measures whether the detections covered the entire plagiarism case, or only a fraction of it. The plagdet score is a combination of the other three metrics.

Table 1 shows our results compared to SP14. The results demonstrate that Elxa performs comparably to state-of-the-art text alignment schemes. The only effective difference between Elxa and SP14 is that Elxa estimates the inverse sentence frequency of words until they have been observed in the intersection of words from two sentences. If this inaccuracy was removed through one of the techniques described in §5.3, then the performance would be identical (at the cost of reduced privacy).

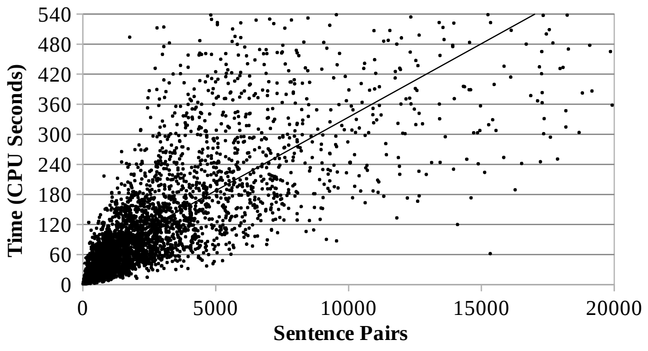

Figure 7: Elxa's text alignment completes within minutes for large documents in the PAN testing corpus.

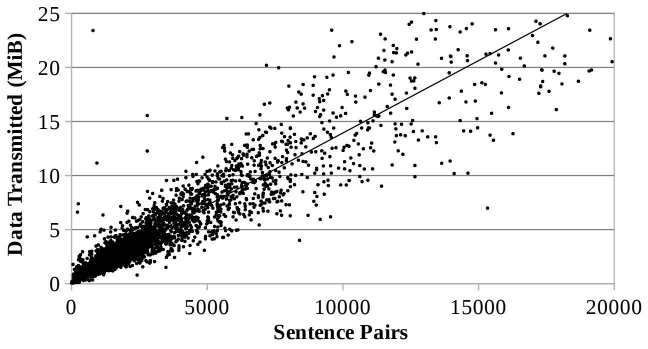

Figure 8: The amount of text alignment data transmitted grows linearly with the number of sentence pairs.

We performed our experiments in parallel using 65 2.4 GHz CPU cores.[5] Since the most expensive operations in the algorithm are performed for each sentence pair, we measured the time and bandwidth requirements in terms of the sentence pair count for each of the 4,800 test cases. Figure 7 plots the time required, and Figure 8 plots the total data transmitted; in both cases, there is a clear linear correlation. The mean number of sentence pairs compared per second was 26 with a standard deviation of 18. The mean number of sentence pairs compared per MiB was 650 with a standard deviation of 230. More operations must be performed when documents are more similar, causing the variance in Figures 7 and 8. Since text alignment is performed rarely, these results show that content matching can be completed relatively quickly and efficiently.

8. CONCLUSION

In this work we introduced Elxa, a privacy-preserving centralized plagiarism detection system incorporating state-of-the-art information retrieval techniques. Elxa is designed to be scalable, and can operate on existing cloud computing infrastructure. Our implementation of Elxa for the Hadoop MapReduce platform demonstrates the feasibility of our approach. As future work, we plan to develop and deploy a full implementation of Elxa based on a real-time cloud computing platform.

We also described a new mechanism for privately computing the cosine and Sørensen-Dice similarity metrics between two HbC parties. Our general approach can be used to construct privacy-preserving variants of many similar schemes. While we discussed

[5]While Elxa's text alignment protocol is highly parallelizable, we used only one core per experiment for evaluation purposes.

Elxa in the context of plagiarism detection, it can also be used to find any type of approximate text reuse, such as similarities between confidential corporate reports.

Plagiarism detection is one of the most important problems facing academia today. The architecture, objectives, and scheme presented in this work are the first step toward joining the best techniques from the privacy-enhancing technology and information retrieval communities. Previously, the best plagiarism detection systems introduced major privacy and legal concerns. Elxa improves this situation dramatically with a reasonable performance cost. It is our hope that this work will encourage wider use of plagiarism detection systems by alleviating these major adoption obstacles.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments and feedback, and NSERC for grant RGPIN-341529. This work benefited from the use of the CrySP RIPPLE Facility at the University of Waterloo.

References

[1] Salha M. Alzahrani, Naomie Salim, and Ajith Abraham. “Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 42.2 (2012), pp. 133–149.

[2] Apache. *Hadoop*. 2011. URL: https://hadoop.apache.org/ (visited on 2016-07-27).

[3] Austin Appleby. *SMHasher*. 2008. URL: https://github.com/aappleby/smhasher (visited on 2016-07-27).

[4] Samhaa R. El-Beltagy and Ahmed Rafea. “KP-Miner: A Keyphrase Extraction System for English and Arabic Documents”. In: *Information Systems* 34.1 (2009), pp. 132–144.

[5] András A Benczúr, Károly Csalogány, Tamás Sarlós, and Máté Uher. “Spamrank—Fully Automatic Link Spam Detection”. In: *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*. 2005.

[6] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. “SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge”. In: *Advances in Cryptology – CRYPTO 2013*. Springer, 2013, pp. 90–108.

[7] Steven Bird. “NLTK: The Natural Language Toolkit”. In: *Proceedings of the COLING/ACL on Interactive Presentation Sessions*. Association for Computational Linguistics. 2006, pp. 69–72.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. “Public Key Encryption with Keyword Search”. In: *Advances in Cryptology – EUROCRYPT 2004*. Springer, 2004, pp. 506–522.

[9] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. “Private Information Retrieval”. In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981.

[10] Chris Clifton, AnHai Doan, Ahmed Elmagarmid, Murat Kantarcioğlu, Gunther Schadow, Dan Suciu, and Jaideep Vaidya. “Privacy-Preserving Data Integration and Sharing”. In: *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. ACM. 2004, pp. 19–26.

[11] Graham Cormode and S. Muthukrishnan. “An improved data stream summary: the count-min sketch and its applications”. In: *Journal of Algorithms* 55.1 (2005), pp. 58–75.

[12] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[13] Lee R. Dice. “Measures of the Amount of Ecologic Association Between Species”. In: *Ecology* 26.3 (1945), pp. 297–302.

[14] John R. Douceur. “The Sybil Attack”. In: *Peer-to-Peer Systems*. Springer, 2002, pp. 251–260.

[15] Tamer Elsayed, Jimmy Lin, and Donald Metzler. “When Close Enough Is Good Enough: Approximate Positional Indexes for Efficient Ranked Retrieval”. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM. 2011, pp. 1993–1996.

[16] Eric Goldman. *Clickthrough Agreement Binding Against Minors—A.V. v. iParadigms*. 2008. URL: http://blog.ericgoldman.org/archives/2008/03/clickthrough_ag.htm (visited on 2016-07-27).

[17] Joan Feigenbaum and Rebecca N Wright. *Systemization of Secure Computation*. Tech. rep. DTIC Document, 2015.

[18] Shahabeddin Geravand and Mahmood Ahmadi. “An efficient and scalable plagiarism checking system using Bloom filters”. In: *Computers and Electrical Engineering* 40.6 (2014), pp. 1789–1800.

[19] Go Project. *The Go Programming Language*. 2009. URL: https://golang.org/ (visited on 2016-07-27).

[20] Oded Goldreich. “Towards a Theory of Software Protection and Simulation by Oblivious RAMs”. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*. ACM. 1987, pp. 182–194.

[21] Google. *Protocol Buffers*. 2008. URL: https://developers.google.com/protocol-buffers (visited on 2016-07-27).

[22] Jan Graßegger, Maximilian Michel, Martin Tippmann, Matthias Hagen, Martin Potthast, and Benno Stein. “The ChatNoir Ranking”. 2012. URL: http://webis15.medien.uni-weimar.de/data/chatnoir-ranking.pdf (visited on 2016-07-27).

[23] Matthias Hagen, Martin Potthast, and Benno Stein. “Source Retrieval for Plagiarism Detection from Large Web Corpora: Recent Approaches”. In: *Working Notes for the CLEF 2015 Conference*. 2015.

[24] Osama Haggag and Samhaa El-Beltagy. “Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring”. In: *Working Notes for the CLEF 2013 Conference*. 2013.

[25] Marti A Hearst. “TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages”. In: *Computational Linguistics* 23.1 (1997), pp. 33–64.

[26] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. “Enhancing Privacy and Trust in Electronic Communities”. In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. ACM. 1999, pp. 78–86.

[27] Wei Jiang, Mummoorthy Murugesan, Chris Clifton, and Luo Si. “Similar Document Detection with Limited Information Disclosure”. In: *Proceedings of the IEEE 24th*

International Conference on Data Engineering. IEEE. 2008, pp. 735–743.

[28] Tibor Kiss and Jan Strunk. “Unsupervised Multilingual Sentence Boundary Detection”. In: *Computational Linguistics* 32.4 (2006), pp. 485–525.

[29] Geoffrey Leech, Paul Rayson, and Andrew Wilson. *Word Frequencies in Written and Spoken English. Based on the British National Corpus*. Routledge, 2014.

[30] Qiming Li, Yagiz Sutcu, and Nasir Memon. “Secure Sketch for Biometric Templates”. In: *Advances in Cryptology – ASIACRYPT 2006*. Springer, 2006, pp. 99–113.

[31] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. 1st ed. Cambridge University Press, 2008.

[32] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. “Building a Large Annotated Corpus of English: The Penn Treebank”. In: *Computational linguistics* 19.2 (1993), pp. 313–330.

[33] Vítor T. Martins, Daniela Fonte, Pedro Rangel Henriques, and Daniela da Cruz. “Plagiarism Detection: A Tool Survey and Comparison”. In: *3rd Symposium on Languages, Applications and Technologies*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 143–158.

[34] Catherine Meadows. “A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party”. In: *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE. 1986, pp. 134–134.

[35] George A. Miller, Edwin B. Newman, and Elizabeth A. Friedman. “Length-Frequency Statistics for Written English”. In: *Information and Control* 1.4 (1958), pp. 370–389.

[36] Mummoorthy Murugesan, Wei Jiang, Chris Clifton, Luo Si, and Jaideep Vaidya. “Efficient privacy-preserving similar document detection”. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 19.4 (2010), pp. 457–475.

[37] Rafail Ostrovsky. “Efficient Computation on Oblivious RAMs”. In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*. ACM. 1990, pp. 514–523.

[38] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. report. Stanford InfoLab, 1999.

[39] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology – EUROCRYPT’99*. Springer, 1999, pp. 223–238.

[40] PAN. *PAN @ CLEF 2016*. URL: http://pan.webis.de/ (visited on 2016-07-27).

[41] Hweehwa Pang, Jialie Shen, and Ramayya Krishnan. “Privacy-Preserving Similarity-Based Text Retrieval”. In: *ACM Transactions on Internet Technology* 10.1 (2010), 4:1–4:39.

[42] Kim Parker, Amanda Lenhart, and Kathleen Moore. “The Digital Revolution and Higher Education. College Presidents, Public Differ on Value of Online Learning”. In: *Pew Internet & American Life Project* (2011).

[43] Benny Pinkas, Thomas Schneider, and Michael Zohner. “Faster Private Set Intersection Based on OT Extension”. In: *Proceedings of the 23rd USENIX Security Symposium*. 2014, pp. 797–812.

[44] Martin F. Porter. “An algorithm for suffix stripping”. In: *Program* 14.3 (1980), pp. 130–137.

[45] Martin Potthast. “Technologies for Reusing Text from the Web”. Dr. rer. nat. thesis. Bauhaus-Universität Weimar, 2012.

[46] Martin Potthast, Steve Göring, Paolo Rosso, and Benno Stein. “Towards Data Submissions for Shared Tasks: First Experiences for the Task of Text Alignment”. In: *Working Notes for the CLEF 2015 Conference*. 2015.

[47] Martin Potthast, Matthias Hagen, Benno Stein, Jan Graßegger, Maximilian Michel, Martin Tippmann, and Clement Welsch. “ChatNoir: A Search Engine for the ClueWeb09 Corpus”. In: *Proceedings of the 35th International ACM Conference on Research and Development in Information Retrieval*. ACM. 2012, pp. 1004–1004.

[48] Martin Potthast, Matthias Hagen, Michael Völske, and Benno Stein. “Crowdsourcing Interaction Logs to Understand Text Reuse from the Web”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2013, pp. 1212–1221.

[49] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. “An Evaluation Framework for Plagiarism Detection”. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics. 2010, pp. 997–1005.

[50] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeno, and Paolo Rosso. “Overview of the 1st International Competition on Plagiarism Detection”. In: *Working Notes for the CLEF 2009 Conference*. 2009.

[51] Stephen E Robertson and K Sparck Jones. “Relevance Weighting of Search Terms”. In: *Journal of the American Society for Information Science* 27.3 (1976), pp. 129–146.

[52] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. “Okapi at TREC-3”. In: *Proceedings of the Third Text REtrieval Conference*. National Institute of Standards and Technology, 1994, p. 109.

[53] Vassil Roussev. “An evaluation of forensic similarity hashes”. In: *Digital Investigation* 8 (2011), S34–S41.

[54] Gerard Salton, Anita Wong, and Chung-Shu Yang. “A Vector Space Model for Automatic Indexing”. In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[55] Bharath K. Samanthula, Gerry Howser, Yousef Elmehdwi, and Sanjay Madria. “An Efficient and Secure Data Sharing Framework using Homomorphic Encryption in the Cloud”. In: *Proceedings of the 1st International Workshop on Cloud Intelligence*. ACM. 2012, 8:1–8:8.

[56] Miguel A. Sanchez-Perez, Grigori Sidorov, and Alexander Gelbukh. “A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014”. In: *Working Notes for the CLEF 2014 Conference*. 2014.

[57] Claude E. Shannon. “Prediction and Entropy of Printed English”. In: *Bell System Technical Journal* 30.1 (1951), pp. 50–64.

[58] Thorvald Sørensen. “A method of establishing groups of equal amplitude in plant sociology based on similarity of

species and its application to analyses of the vegetation on Danish commons”. In: *Biologiske Skrifter* 5 (1948), pp. 1–34.

[59] Karen Spärck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[60] Benno Stein, Sven Meyer zu Eissen, and Martin Potthast. “Strategies for Retrieving Plagiarized Documents”. In: *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval*. ACM. 2007, pp. 825–826.

[61] The Lemur Project. *The Clueweb09 Dataset*. 2009. URL: http://lemurproject.org/clueweb09/ (visited on 2016-07-27).

[62] Turnitin. *Plagiarism Report*. 2013. URL: http://turnitin.com/en_us/resources/blog/421-general/1660-plagiarism-report-infographic (visited on 2016-07-27).

[63] Turnitin. *Technology to Improve Student Writing*. 2016. URL: http://turnitin.com/ (visited on 2016-07-27).

[64] Nik Unger, Sahithi Thandra, and Ian Goldberg. “Elxa: Scalable Privacy-Preserving Plagiarism Detection”. In: *Workshop on Privacy in the Electronic Society (WPES’16)*. ACM. 2016.

[65] Dinusha Vatsalan, Peter Christen, and Vassilios S. Verykios. “A taxonomy of privacy-preserving Record linkage techniques”. In: *Information Systems* 38.6 (2013), pp. 946–969.

[66] Hugo Zaragoza, Nick Craswell, Michael J Taylor, Suchi Saria, and Stephen Robertson. “Microsoft Cambridge at TREC 13: Web and Hard Tracks”. In: *Proceedings of the 13th Text REtrieval Conference*. Vol. 4. 2004.

A. CMS SIZE CALIBRATION

When deploying Elxa, one should choose the CMS size based on the specific corpus being used. In this appendix, we show an example of a CMS stored by the root and discuss a general procedure for configuring CMS sizes.

To demonstrate the structure of snippets, we processed the text of this paper using our document submission client. Elxa breaks this document into 95 snippets, each represented by a CMS. The CMS size normally used by our implementation is too large to include here, but for illustrative purposes we processed the document using a smaller size. The first snippet, which spans the initial sentences of the abstract, is encoded by the following 10×25 CMS:

```
8 3 1 2 3 1 1 4 3 4 1 2 4 7 1 1 2 3 3 1 1 3 1 1 3
2 1 3 2 3 5 2 1 1 1 2 3 5 5 2 3 2 1 0 1 1 5 4 2 7
1 4 2 0 1 0 6 4 3 4 3 3 5 0 1 2 3 3 0 8 0 0 1 2 8
1 5 1 0 1 3 2 2 4 3 0 3 0 3 2 2 2 7 1 4 1 4 4 6 3
2 2 2 1 1 1 3 4 5 4 1 2 4 2 3 7 2 0 0 3 4 4 1 4 2
3 1 2 6 2 0 3 5 0 4 5 4 4 3 1 1 4 0 2 2 2 0 3 5 2
4 0 2 4 3 3 1 2 4 0 2 5 1 5 3 2 3 3 1 6 1 1 4 3 1
2 4 3 1 3 1 3 1 3 3 1 5 1 1 5 2 2 2 1 5 2 5 6 1 1
1 1 4 3 1 2 1 3 7 3 0 3 1 4 5 3 2 2 1 3 2 5 3 3 1
7 1 2 3 2 0 3 1 3 5 3 3 2 1 1 3 5 1 0 1 3 5 5 2 2
```

The best approach to selecting the CMS size for a real deployment is to begin by choosing a desired error rate. Higher error rates will decrease the effectiveness of the source retrieval procedure, but will reveal less information to the root. Given an error rate, the total number of cells in the CMS should be adjusted until the optimal shape configuration yields the desired rate; this procedure minimizes the storage and network overhead for the given performance-privacy tradeoff. The optimal shape for a given cell count depends on the number of words stored in a snippet.

Table 2: The true positives encountered by the root

FREQUENCY	WORD TOKENS
1	abstract, abuse, academic, another, assembling, attractive, authors, centralized, challenging, completely, conferences, contents, database, databases, document, documents, dramatically, e, educational, effectiveness, ensure, entities, facing, g, identical, improves, institutions, introduces, journals, large, legal, making, multiple, must, number, one, operator, papers, plagiarized, preserving, problem, products, revealed, scalable, source, submitted, submitting, target, techniques, today, typically, wish, work
2	issues, privacy

Table 3: The false positives encountered by the root

FREQUENCY	WORD TOKENS
1	$750,000,000, £127,500, 1,195, 18–4, 36532, bipm, birds-parrot, californian-based, datis, earth/shield, ed7/k2, episodically, gas-exchange, gershuny, mizzi, oversubscribe, pictures/worcester, reassembled, s-bus, sela, unmuzzled, withy
2	gas-exchange

B. ROOT ATTACK CAPABILITIES

In this appendix, we show examples of the information that the root can learn by performing dictionary attacks.

Following the approach described in Appendix A, we used our Elxa client to generate 9×167 snippets for the text of this paper as it appeared at WPES’16 [64]; this CMS size is appropriate for conference papers, which often use longer words than typical English documents. To demonstrate the type of information that can be recovered from snippets, we performed a dictionary attack on the first snippet using the method described in §7.1. Table 2 lists the word frequencies correctly recovered from the snippet (i.e., true positives), and Table 3 lists the recovered frequencies that were not in the original text (i.e., false positives). For ease of presentation, the tables group words by frequency, and the words are shown unstemmed (in a true deployment, all words in the snippet would be passed through the Porter word stemmer before being hashed). The attack failed to recover “Elxa” from the snippet, since it was not in the dictionary; this was the only false negative. Even if the attacker could unerringly classify the tokens in Table 3 as false positives, recovering the original text from the data in Table 2 is challenging.

In addition to analyzing the snippets in its database, the root can learn information about documents by performing dictionary attacks on queries. To demonstrate this attack, we used our source retrieval implementation to generate search queries for this paper. The client produced 175 queries. Normally, most of these queries would be suppressed because they would match potential sources that were already downloaded. The following ten queries were excerpted from the complete list:

1. disclosur plagiar aggreg detect sensit prohibit content potenti law involv
2. unstructur usabl user similar interact long look elxa object size
3. sourc reason retriev keyword statist
4. case lower last stemmer sentenc merg fewer four repeat pass
5. cryptosystem keygen dimension scalar word multipl share plaintext support proceed
6. duplic respect proport remain approxim near unrel elxa stabl grow
7. eyal plagiar sudan benni madhu detect chor kushilevitz

8. build marcinkiewicz plagiar annot marcus ann beatric detect santorini penn
9. danish plant sociolog potthast thorvald common establish amplitud martin analys
10. uner group eas snippet attack negat correct fail recov present

Note that the keywords have been stemmed, so some information about word form is unavailable to the root. The root would not learn any keywords that were not contained in its dictionary. Even with a fully comprehensive dictionary, recovering the original text of this paper from these queries would be extremely difficult.

While the words recovered from snippets or queries may reveal information like the topic of the document, this is also the type of information that is needed to perform plagiarism detection.