# Performance Evaluation

This section describes the performance evaluation conducted to compare the performance of the *Inverted index* (leaks too much information and therefore may not be appropriate, but this is the most popular index in information retrieval), a secure index using the *Bloom filter*, and a secure index using the *Perfect filter*.

We compare the performance of the three offline secure indexes by implementing and executing searches in an isolated host computer for various **experiment designs**. *[Describe how we implemented the three search methods, such as the programming language used for implementation, the specs of the host computer used for our performance studies, etc.]*

## Controllable Inputs and Measurable Outputs

For the sake of quantifying the performance of each index, several different measurable outputs and controllable inputs have been identified. See Table 1, “Output Parameters” and “Input Parameters”. The identified *input parameters* will be varied in a controlled manner for the purpose of measuring the effect they have on the outputs. Specifically:

1. For each **experiment design**, execute steps two through eleven.
2. For each *input parameter*, a sequence of different values will be chosen for the given experiment design.
3. An ***artificial* text corpus** will be constructed with respect to the chosen inputs.
4. A set of input queries will be generated for the artificial text corpus.
5. A non-secure, established IR system will be employed to classify the relevancy of each document in the artificial text corpus to each query. That is, a mapping from query to ranked set of documents will be generated.
6. A **secure index** of each type for the artificial text corpus will be generated with respect to the chosen *inputs* and *experiment design*.
7. Each secure index construction will classify the relevancy of each document in the artificial text corpus to each query.
8. The outputs will be measured and recorded. (A trial run.)
9. Go back to step three **N** more times.
10. An average of each of the measured outputs will be recorded for the given inputs.
11. Go back to step two until the sequence of values for the inputs are exhausted.
12. Go back to step one until each experiment design has been ran.

|  |  |
| --- | --- |
| **Output Parameters** | **Definition** |
| Secure index database (SID) construction time | Total time taken to construct SID for artificial text corpus. |
| SID query lag time | Total time taken to respond to all generated queries. |
| Resident SID memory-space | Total system memory of resident SID process or processes (for securely indexing a given artificial text corpus). |
| Non-resident SID file-size | Total file size of SID for artificial text corpus. |
| Precision | {retrieved documents AND relevant documents} / {retrieved documents} |
| Recall | {retrieved documents AND relevant documents} / {relevant documents} |
| Mean average precision (MAP) | Since it is trivial to optimize **recall** by retrieving **all** document and trivial to optimize **precision** by retrieving **no** documents, a more insightful metric is based on the precision-recall curve. Moreover, unlike **precision** and **recall**, **MAP** may take into account the order (how retrieved documents were ranked).  **Note**: To determine which documents are relevant to specific queries, an existing IR library will be used on the artificially generated text corpuses. The output from this library will serve to “classify” the relevancy of certain documents to certain queries. |
| **Input Parameters** | **Definition** |
| Number of documents | Total number of documents in artificial text corpus. |
| Document size | Number of words per document. |
| Block size | Maximum number of words per block. |
| Search class | There are four search classes:   1. **No** proximity scoring, **no** term weighting (*Boolean matching: no degree of relevancy – a document is relevant to a query if all the terms in the query are present, otherwise it is not relevant.*) 2. Proximity scoring, **no** term weighting 3. **No** proximity scoring, term weighting 4. Proximity scoring, term weighting |
|  |  |
| False positive rate | The theoretical false positive rate. This ought to have a strong effect on **precision** output.  **Note**: For the **Perfect filter**-based secure index, the false positive rate is determined by the number of bits per hashed index. For the **Bloom filter**-based secure index, false positive rate is determined by the number of hash functions, *k*, the number of index positions, *m*, and the number of members, *n*. |
| Number of query terms | Number of terms per query. **Note:** A single term is either a *word* (1-gram) or a *phrase* (n-gram). |
| Number of words per query term | The number of words per query term. Number of words per query term can vary from 1 (the term is a single word, or 1-gram) to n (the term is an n-gram).  Note: A *term* is present or not present in a secure index; approximate term matching is possible (e.g., locality-sensitive hashing techniques like stemming), but will be considered separately to simplify analysis. Likewise, wildcard matching will also be considered separately. |
| Text corpus class | (Previously ***document class***) Artificial text corpuses will be generated from a trigram language model. Trigram language models will in turn be generated from typical actual text corpuses. This is done to facilitate precise control over input parameters **Number of documents** and **Document size** while still generating an arbitrary number of plausible **SDI**s for testing purposes.  Classes are defined by actual text corpuses. Actual text corpuses are defined by word distributions (subject—e.g., general, medical, etc.), the number of words per document, and the number of documents. |

**Table 1** - Definitions of the input and the output parameters

## Experiment Designs

First, each **experiment design** has the following in common:

* Text corpus class
  + The trigram language models will be constructed from a real text corpus. Real text corpuses will be sampled from ***general text, web pages, medical text***.
  + The constructed trigram language model will be used to generate the documents in the artificial text corpus.
* Number of documents in the database: ***K***
  + **K** can be a sequence of natural numbers or a random variable (distribution). A plausible distribution can be generated from the chosen text corpus class.
* Number of words per document: ***N***
  + **N** can be a sequence of natural numbers or a random variable (distribution). A plausible distribution can be generated from the chosen text corpus class.
* Number of query terms and number of words per query term
  + A single term is either a *word* (1-gram) or a *phrase* (n-gram). Some plausible distribution will be devised to allow for the following kinds of experiments:
    - Experiments in which only 1 single keyword search is performed.
    - Experiments which consist of multiple keyword searches.
    - Experiments which consist of multiple term searches, where a term is as previously described, e.g., four terms, two of which are single words and two of which are exact phrases (n-grams of some arbitrary size).
* False positive rate: A sequence of theoretical false positive rates, as determined by the secure index structure.

### Experiment #0 – Baseline Experiment

A **Boolean matching** experiment (i.e., a document is relevant or not relevant—*no degree of relevancy scoring will be provided*) will serve as the baseline experiment. Many existing encrypted search solutions can be represented with this approach.

* Block size: ***All words in the document*** (1 Block). Since the base experiment performs no relevancy scoring, there is no reason to segment documents into multiple blocks for proximity scoring or keyword weighting. The most efficient representation is therefore a single block.  
    
  **Note:** This should have the least time and space complexity, but also the worst relevancy score.

### Experiment #1

**Proximity scoring, no term weighting.** Every term will be equally weighted, but proximity will be taken into consideration.  
  
On the one hand, reducing the block size should permit the proximity measure to more effectively rank the relevancy of documents to queries, but on the other hand it will harm the other outputs, e.g., file size, lag time, etc. In particular, depending upon the chosen algorithm for proximity scoring, this could significantly increase the lag time.

* Block size: A sequence of different block sizes will be tried. There will probably be no obviously best block size configuration; it will depend on the importance of the different outputs.

### Experiment #2

**No proximity scoring, term weighting.** Proximity will not be taken into account, but terms will not be equally weighted. A standard heuristic, e.g., some **tf-idf** variant, will be employed.  
  
On the one hand, reducing the block size should provide the **tf-idf** metric with more information to more effectively rank the relevancy of documents to queries, but on the other hand it will harm the other outputs, e.g., file size, lag time, etc. In particular, lag time may be significantly increased due to the fact that the entire secure index collection[[1]](#footnote-1) must be queried for term frequencies.

* Block size: A sequence of different block sizes will be tried. There will probably be no obviously best block size configuration; it will depend on the importance of the different outputs.

### Experiment #3

**Proximity scoring, term weighting.** Proximity scores and non-uniformly weighted keywords will be used.   
  
On the one hand, reducing the block size should provide the tf-idf metric and the proximity metric with more information to more effectively rank the relevancy of documents to queries, but on the other hand it will harm the other outputs, e.g., file size, lag time, etc.

* Block size: A sequence of different block sizes will be tried. There will probably be no obviously best block size configuration; it will depend on the importance of the different outputs.

## Observed Outputs

### Experiment #0

Describe the outcomes from this experiment. Graphs and tables may be used.

### Experiment #1

Describe the outcomes from this experiment. Graphs and tables may be used.

### Experiment #2

Describe the outcomes from this experiment. Graphs and tables may be used.

## Performance Analysis

We analyze the outputs from our experiments. We describe the analysis we made and summarize them. The major findings from our experiments should be emphasized here and we should use the highlights in the section of “conclusions” after this section.

1. Term frequencies should not be given freely to the untrusted CSP; however, once term frequency information has been leaked for certain hidden terms in queries, the CSP may decide to track this information. Indeed, caches may be used by the CSP to quickly map hidden query terms to a set of documents. [↑](#footnote-ref-1)