Maximizing Confidentiality in Encrypted Search Through Entropy Optimization

Alexander Towell
atowell@siue.edu

Abstract

Encrypted search systems enable information retrieval over encrypted data while preserving confidentiality of queries and documents. However, observable patterns in encrypted queries, access patterns, and result sets can leak information about plaintext content. We present an information-theoretic framework for analyzing and improving the confidentiality of encrypted search systems. We model encrypted search activities as random processes and measure their entropy, comparing observed entropy against the maximum entropy possible under system constraints such as query arrival rates, vocabulary size, and document collection size. We derive closed-form solutions for the maximum entropy distribution and show that the ratio of observed to maximum entropy provides a quantitative measure of confidentiality bounded between 0 and 1. Since entropy can be estimated using lossless compression, our framework enables practical measurement without requiring explicit probabilistic models. We demonstrate that confidentiality can be systematically improved through techniques such as homophonic encryption, artificial query injection, and query aggregation, each trading specific resources for entropy gains. A case study shows that a typical system achieving 59% efficiency can be improved to 85% efficiency with moderate space and bandwidth overhead. Our approach provides principled guidance for balancing confidentiality against performance in encrypted search deployments.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

Information retrieval over encrypted data presents a fundamental tension: enabling search functionality requires revealing some information about queries and documents, yet preserving confidentiality requires hiding this information. An information retrieval process begins when a search agent submits a query to an information system, where the query represents an information need. The system returns relevant objects, typically documents, satisfying that need.

encrypted search (ES) extends this paradigm to untrusted environments where an encrypted search provider obliviously retrieves confidential objects satisfying confidential information needs of authorized search agents. The system employs two cryptographic components: secure indexes provide queryable encrypted representations of confidential documents (constructed using techniques such as Bloom filters, perfect hash functions, or other approximate membership structures), while hidden queries provide one-way encrypted representations of plaintext queries.

Ideally, both secure indexes and hidden queries would reveal no information beyond what is necessary for retrieval. They would appear as uniform uncorrelated noise with lengths uncorrelated to their plaintext counterparts. However, functionality requirements create an inherent tradeoff: for an untrusted system to perform oblivious searches, hidden queries must map to relevant secure indexes, necessarily revealing some information through observable patterns.

1.1 The Information Leakage Problem

Claude E. Shannon, in his seminal 1949 paper *Communication Theory of Secrecy Systems*[16], established two essential properties for secure encryption:

1. *Confusion* obscures relationships between plaintext and ciphertext, ensuring ciphertext statistics depend on plaintext in ways too complex to exploit. Simple substitution ciphers fail this requirement: preserving frequency distributions allows adversaries to recover plaintexts through statistical analysis.

2. *Diffusion* ensures each plaintext symbol affects many ciphertext symbols, ideally causing every plaintext symbol to influence every ciphertext symbol.

Encrypted search systems struggle to achieve Shannon's ideals. Even with strong cryptographic primitives, observable patterns leak information. Recent attacks demonstrate that access patterns[9, 2], query volumes[8], and timing information enable adversaries to reconstruct significant portions of plaintext queries and document content.

1.2 An Information-Theoretic Approach

Rather than relying solely on computational hardness assumptions, we analyze encrypted search through information theory. Our approach models encrypted search activities—hidden queries,

inter-arrival times, search agent identities, and result sets—as random processes and measures their entropy. We then derive the maximum entropy achievable under system constraints (query rates, vocabulary size, collection size) and define *confidentiality* as the ratio of observed to maximum entropy, yielding a score between 0 (predictable) and 1 (maximally unpredictable). Since entropy equals optimal compression length, we can estimate it practically using lossless compressors without explicit probabilistic models.

This framework quantifies fundamental confidentiality limits and provides principled guidance for improvement through entropy maximization.

1.3 Contributions

This paper makes the following contributions:

1. An information-theoretic framework for analyzing encrypted search confidentiality through entropy measurement
2. Derivation of maximum entropy distributions under realistic system constraints with closed-form solutions
3. A practical confidentiality metric based on entropy ratios that enables comparison across systems and configurations
4. Techniques for systematically improving confidentiality including homophonic encryption, artificial queries, and query aggregation
5. A compression-based entropy estimation method enabling measurement without explicit probabilistic models
6. Analysis demonstrating quantitative tradeoffs between confidentiality and resource costs
7. A case study showing confidentiality improvements from 59% to 85% efficiency with moderate overhead

The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 formalizes the encrypted search model. Sections 4–7 develop the entropy-based confidentiality framework. Section 8 presents techniques for improving confidentiality. Section 9 demonstrates the framework through a case study. Section 10 concludes.

2 Related Work

2.1 Encrypted Search Systems

The problem of searching over encrypted data while preserving confidentiality has been studied extensively since the seminal work of Song et al.[17], who introduced practical techniques for searching on encrypted data using cryptographic primitives. Goh[6] introduced the concept of secure indexes using Bloom filters to enable efficient searching over encrypted document collections.

Curtmola et al.[4] provided improved security definitions for searchable symmetric encryption and presented efficient constructions achieving stronger security guarantees. Their work established formal security models that have become standard in the field. Cash et al.[1] extended this work to support Boolean queries at scale, while Kamara et al.[12] addressed the challenge of dynamic encrypted search systems where documents can be added or removed.

2.2 Attacks on Encrypted Search

Despite strong cryptographic protections, encrypted search systems are vulnerable to attacks that exploit observable patterns in encrypted queries and access patterns. Islam et al.[9] demonstrated that access pattern leakage can enable adversaries to recover significant information about plaintext queries and documents. Cash et al.[2] formalized leakage-abuse attacks, showing how adversaries can exploit leaked information to reconstruct queries.

Pouliot and Wright[14] demonstrated inference attacks on practical encrypted search deployments, while Grubbs et al.[8] showed that even volume leakage from range queries can enable database reconstruction. These attacks motivate the need for principled approaches to understanding and quantifying information leakage in encrypted search systems.

2.3 Information-Theoretic Approaches

Shannon's foundational work on information theory[15, 16] established the mathematical framework for measuring information and uncertainty. His analysis of secrecy systems demonstrated the importance of entropy in cryptographic security. The principle of maximum entropy, developed by Jaynes[10, 11], provides a rational basis for selecting probability distributions under constraints.

Our work applies these information-theoretic principles to analyze encrypted search systems. By measuring the entropy of observable encrypted search activities and comparing it to the maximum entropy possible under system constraints, we provide a quantitative measure of confidentiality that guides the design of countermeasures.

2.4 Oblivious Computation

Oblivious RAM[7, 18] provides techniques for hiding access patterns to memory by obfuscating read and write operations. While ORAM achieves strong security guarantees, it introduces significant computational overhead. Our approach shares the goal of hiding patterns in observable activities but focuses specifically on encrypted search and leverages information-theoretic measures to balance confidentiality against performance costs.

2.5 Anonymity and Mix Networks

Mix networks[3] and onion routing systems like Tor[5] provide anonymity by obscuring the relationship between senders and receivers of messages. These techniques complement our approach by addressing the challenge of hiding search agent identities. Our entropy maximization framework can incorporate mix network properties as one component of a comprehensive confidentiality strategy.

3 Encrypted search model

An encrypted search system consists of three stages:

1. **Query generation**: Search agents generate plaintext queries representing confidential information needs. These queries traverse a trusted channel to an obfuscator.

2. **Obfuscation**: The obfuscator transforms plaintext queries into hidden queries, which traverse an untrusted channel to the encrypted search provider.

3. **Encrypted retrieval**: The provider obliviously maps each hidden query to a set of secure indexes representing confidential objects that satisfy the query.

We now formalize each component of this system. A set is an unordered collection of distinct elements. A set of particular interest is given by the following definition.

Definition 3.1. *The finite set of all bit strings of length n is denoted by*

$$\mathbb{B}_n = \{b \colon b \in \{0,1\}^n\} \tag{3.1}$$

with a cardinality given by

$$|\mathbb{B}_n| = 2^n \,. \tag{3.2}$$

The set of bit strings of length n may be put in one-to-one correspondence with any set having 2^n or more elements.

The bit length of an object x is denoted by

$$\mathtt{BL}(x) \,, \tag{3.3}$$

e.g., the bit length of a bit string $x \in \mathbb{B}_n$ is $\mathtt{BL}(x) = n$. The countably infinite set of all bit strings of any length is denoted by

$$\mathbb{B} = \{b \colon b \in \{0,1\}^*\} \,, \tag{3.4}$$

where $*$ denotes any non-negative integer.

Definition 3.2 (Confidential object)**.** *A* confidential *object, like a document, is an object that only an authorized set of search agents should be able to comprehend.*

The objective of an encrypted search system is to satisfy the *information needs*, as represented by queries, of authorized search agents by retrieving a set of *relevant* documents from the document store.

Definition 3.3 (Secure index)**.** *A* queryable *representation of a confidential object is denoted a secure index if it meets the following conditions:*

1. *If it has a bit length n, then it obtains (at least approximately) the* maximum entropy*. That is, it is* incompressible*.*

2. *An estimator of the cardinality of a* bag-of-words *secure index has an average entropy c_2 per element.*

3. *A hidden query applied to a secure index only reveals information about the set of trapdoors in the hidden query. If the document model is a bag-of-words, then it only reveals an approximate membership with a false positive rate ε. If the document model is more general, it reveals only the information necessary to determine relevance.*

By the definition above, a secure index may be *obliviously* queried by the encrypted search provider on behalf of authorized search agents.

The requirement that a secure index be incompressible (condition 1) is crucial for preventing information leakage through structural patterns. Various cryptographic constructions achieve this property through different mechanisms. Bloom filters[6] provide space-efficient approximate set membership testing with tunable false positive rates, while maintaining high entropy through their bit array representation. Perfect hash functions and other approximate map constructions[21, 20] extend this concept to support richer query semantics. The entropy of a secure index can be measured empirically through lossless compression algorithms: a well-obfuscated secure index should exhibit compression ratios near 1.0, indicating that it approaches maximum entropy for its bit length.

The choice of secure index construction involves tradeoffs between space efficiency, query expressiveness, and information leakage. More sophisticated constructions supporting phrase queries, proximity search, or ranked retrieval necessarily reveal additional structural information about the underlying document. Our entropy-based framework applies uniformly across these constructions: regardless of the specific cryptographic technique employed, a secure index achieves better confidentiality when its observable representation has higher entropy relative to the maximum possible entropy given the system constraints.

An encrypted search system may support many different kinds of queries. One of the simplest kinds of queries is a bag-of-words; that is, a search agent represents an *information need* with a set of relevant *search keys*. The bag-of-words query model may be a crude approximation of an *information need* but it is common in information retrieval and for simplicity we make the following assumption.

Assumption 3.1. *The query model is a bag-of-words.*

A query submitted to a encrypted search system should only be understood by the search agent that generated the query.

Definition 3.4 (Hidden query)**.** *A hidden query represents a confidential* information need *of an authorized search agent, where the query is suppose to be incomprehensible to everyone except the indicated search agent.*

Before we define how a plaintext bag-of-words query is transformed into a hidden query, we need to define the trapdoor function.

Definition 3.5 (Trapdoor)**.** *Given a secret $s \in \mathbb{B}$ and a word $x \in \mathbb{B}$, a trapdoor of x is a* one-way *transformation to a word $y \in \mathbb{B}_m$ as given by*

$$y \leftarrow \mathtt{h}(x{+\!\!+}s)\,, \tag{3.5}$$

where $\mathtt{h}\colon \mathbb{B} \mapsto \mathbb{B}_m$ is a one-way *function and $+\!\!+$ is the concatenation operator.*

By *one-way*, we mean that given a word y, finding an x such that

$$y = \mathtt{h}(x)$$

is not tractable. The *one-way* property of the transformation is the reason we call it a trapdoor function.

Definition 3.6. *A random oracle is an idealized cryptographic primitive that maps inputs to uniformly random outputs in an unpredictable manner.*

Assumption 3.2. *The one-way function $\mathtt{h}\colon \mathbb{B} \mapsto \mathbb{B}_m$ approximates*[1] *a random oracle.*

If a collision between plaintext words x and y were to occur, they would be aliased–i.e., indistinguishable–in the encrypted search system. We make the following simplifying assumption about the trapdoor function.

Assumption 3.3. *The codomain of $\mathtt{h}$, the set of bit strings of length m, is sufficiently large such that collisions between words that represent information needs are negligible and may be ignored.*

[1] Generally, $\mathtt{h}$ is a *cryptographic hash function.*

The function that transforms a plaintext bag-of-words into a trapdoor bag-of-words is given by the following definition.

Definition 3.7 (hidden query cryptographic protocol)**.** *The cryptographic protocol that transforms a plaintext query* $\mathbf{x}$ *into a hidden query* $\check{\mathbf{x}}$ *is given by*

$$\check{\mathbf{x}} \leftarrow \texttt{hidden\_query\_generator}(\mathbf{x}) \tag{3.6}$$

where

$$\texttt{hidden\_query\_generator}\colon [\text{bag-of-words}] \mapsto \mathbb{B}_m^* \,. \tag{3.7}$$

uses the trapdoor function given by Definition 3.5.

If the policy is a *substitution cipher* as given by Algorithm 1, then plaintext words have 1 or more possible trapdoors that are uniformly sampled from. In a *simple* substitution cipher, where each word maps to a single trapdoor, the substitution policy is a constant given by

$$\texttt{substitutions}(x) = 1 \tag{3.8}$$

for all words $x \in \mathbb{B}$.

Algorithm 1: Cryptographic substitution cipher policy for mapping plaintext queries to hidden queries

params:
1. $s \in \mathbb{B}$ is the secret.
2. $\texttt{substitutions}\colon \mathbb{B} \mapsto \mathbb{B}_m$ is the *substitution policy*, where m is the bit length of trapdoors.

input : A bag-of-words plaintext query $\mathbf{x}$.
output : A corresponding bag-of-words hidden query $\check{\mathbf{x}}$.

```
function hidden_query_generator(x)
    x̌ ← ∅;
    for x ∈ x do
        p ← substitutions(x);
        sample k from DU(1, p);
        s′ ← s;
        for j ← 1 to p do
            // Since h approximates a random oracle, prepending the single bit
               value "1" is sufficient to generate another uncorrelated
               trapdoor.
            s′ ← 1++s′;
            x̌ ← h(x++s′);
            x̌ ← x̌ ∪ {x̌};
        end
    end
    return x̌;
```

Definition 3.8. *The secure index document model is an* approximate map*[21, 20] with a* false positive rate ε *where the keys are searchable words in a corresponding confidential object and the values represent information about the word, such as its* multiplicity*.*

Algorithm 2: Cryptographic noise policy decorator for hidden queries

params:

1. $s \in \mathbb{B}$ is the secret.
2. n_{noise} is the number of artificial trapdoors to inject per query.

input : A bag-of-words hidden query $\check{\mathbf{x}}$.

output : A perturbed bag-of-words hidden query $\check{\mathbf{x}}'$ with artificial trapdoors.

function `hidden_query_noise_decorator(`$\check{\mathbf{x}}$`)`

- $\check{\mathbf{x}}' \leftarrow \check{\mathbf{x}}$;
- **for** $j \leftarrow 1$ **to** n_{noise} **do**
  - sample r uniformly from $\mathbb{B}$;
  - $\check{x}_{\text{noise}} \leftarrow$ `h(`r++s`)`;
  - $\check{\mathbf{x}}' \leftarrow \check{\mathbf{x}}' \cup \{\check{x}_{\text{noise}}\}$;
- **end**
- **return** $\check{\mathbf{x}}'$;

In a *bag-of-words* document model, the value is whether a word exists in a document. In this case, a special-case of the approximate map, the *approximate set*, is used.[22, 19]

The function that transforms a plaintext document into a secure index is given by the following definition.

Definition 3.9 (Secure index construction cryptographic protocol)**.** *The cryptographic protocol that transforms a plaintext document into a secure index is given by*

$$\mathbf{d'} \leftarrow \texttt{secure\_index\_maker}(\mathbf{d}) \tag{3.9}$$

where

$$\texttt{secure\_index\_maker}\colon [\text{document}] \mapsto [\texttt{secure\_index}]\,. \tag{3.10}$$

If each *queryable* word x has multiple possible substitutions and the document model is a *bag-of-words*, then the algorithm given by Algorithm 3 is a candidate for secure index construction. It relies upon a data structure implementing the approximate set abstract data type.[22, 19]

Algorithm 3 may also be used to support *phrase searching* by also including the *bigrams* in a document $\mathbb{D}$. This is known as a *biword model* for *phrase search*[13]. A *phrase* is assumed to be in the document if all the bigrams in the *phrase* are in the approximate set. Note that *false positives* on phrases with more than two words occur at a different rate than the false positive rate of the *approximate set*. Other variations may also be supported, e.g., fuzzy searches or wildcard searches, at the cost of increased space and time complexity.

Definition 3.10 (Adversary)**.** *The adversary is an* untrusted *agent that tries to extract* confidential *information about encrypted search activities.*

Definition 3.11 (Kerckhoffs's principle)**.** *A cryptosystem should be secure even if everything about the system, except the* secret*, is known to the adversary. If the secret is compromised, the cryptosystem is compromised.*

In our encrypted search model, the *secret* is a set of well-defined *parameterizations.*

Assumption 3.4. *The adversary knows everything about the system except a well-defined set of paramterizations.*

Algorithm 3: Plaintext document to secure index *bag-of-words* generator

params:
$s \in \mathbb{B}$ is the secret.
ε is the false positive rate.
`substitutions`$: \mathbb{B} \mapsto \mathbb{B}_m$ is the *substitution policy*, where m is the bit length of trapdoors.
input : $\mathbb{D}$ is a *bag-of-words* representing a plaintext document.
output : An approximate set of the trapdoors of the words in $\mathbb{D}$.

```
1  function secure_index_maker(𝔻)
       // 𝔻′ temporarily stores the trapdoors of the plaintext words.
2      𝔻′ ← ∅;
3      for x ∈ 𝔻 do
4          s′ ← s;
5          for j ← 1 to substitutions(x) do
6              x̆^(j) ← h(x++s′);
7              𝔻′ ← 𝔻′ ∪ {x̆^(j)};
               // Since h approximates a random oracle, prepending the single bit
                  value "1" is sufficient to generate another uncorrelated
                  trapdoor.
8              s′ ← 1++s′;
9          end
10     end
11     return an approximate set of 𝔻′ with a false positive rate ε;
```

In particular, the hidden queries time series is known (observable) to the adversary.

There are many ways an adversary might gain insight into encrypted search activities, e.g., the confidential *identity* of search agents may be exposed through traffic analysis even if *onion routing* is used[5].

The *secret* is given by the following definition.

Assumption 3.5. *The set of parameters considered to be the secret is given by the following:*

1. *A* secret key *used to generate trapdoors.*

There are two primary components in an encrypted search system, the obfuscator and the encrypted search provider. The obfuscator is given by the following definition.

Definition 3.12 (obfuscator)**.** *The obfuscator receives plaintext queries from authorized search agents, transforms them into hidden queries using some set of cryptographic protocols*[2]*, and transmits the hidden queries to the encrypted search provider.*[3]

The obfuscator may reside on a search agent's host computer or a physically separate computer that is network accessible. Either way, since the obfuscator receives plaintext queries, it must be *trusted*.

Assumption 3.6. *The authorized search agents have access to the obfuscator through a* secure communications channel.

[2] One of which is given by Definition 3.7.

[3] In practice, the obfuscator may transmit the hidden queries back to the search agents and they may then transmit the hidden queries directly to the encrypted search provider.

By Assumption 3.6, confidential plaintext queries may be securely transported to the obfuscator without being compromised by the adversary. The encrypted search provider is given by the following definition.

Definition 3.13 (encrypted search provider)**.** *The encrypted search provider receives hidden queries that are the output of the obfuscator and maps them to a set of confidential objects. The mapping is given by some function*

$$\texttt{hidden\_query\_mapper}\colon \mathbb{B}_m^* \mapsto \texttt{powerset}\left(\{1,\ldots,N\}\right) , \tag{3.11}$$

where $\mathbb{B}_m^*$ *is the set of hidden queries and* $\mathbf{d}^*$ *is a set of references to confidential objects.*

The encrypted search provider may reside on a search agent's host computer or a physically separate computer that is network accessible. Either way, we make the following assumption about the link between the *obfuscator* and the encrypted search provider.

Assumption 3.7. *The obfuscator communicates with the encrypted search provider through an* untrusted *communications channel.*

Typically, the network connection between the obfuscator and the encrypted search provider is trusted (encrypted), and thus the only untrusted element in this link is the encrypted search provider, e.g., the adversary may have compromised the security of the encrypted search provider itself.

The adversary may modify the result sets or the hidden queries (such that the search agents receive false results)[4]. Strategies (such as redundancy) exist that may make it possible to detect such modifications, but we make the following assumption.

Assumption 3.8. *The search agents receive* truthful *results to their* information needs.

The *information* that flows across the *untrusted* channel is given by the following definition.

Definition 3.14 (hidden query stream)**.** *The hidden queries and result sets flowing across the* untrusted *communications channel is an ordered sequence of tuples. The* k^{th} *tuple is given by*

$$\left\langle \check{t}_k, \check{a}_{j_k}, \check{\mathbf{x}}_\mathbf{k}, \check{\mathbf{d}}_\mathbf{k} \right\rangle , \tag{3.12}$$

where

$\check{t}j$ *is a* time stamp *of the* k^{th} *hidden query,*

$\check{a}j_k$ *is the* identity[5] *of the search agent submitting the* k^{th} *hidden query,*

$\check{\mathbf{x}}k$ *is the* k^{th} *hidden query corresponding to plaintext query, and*

$\check{\mathbf{d}}k$ *is the result set of* confidential *objects satisfying the* information need *of the query.*

The *time stamps* observed in a stream of hidden queries provides a partial ordering such that if time stamp t_j of the j^{th} hidden query is *earlier in time* than time stamp t_k of the k^{th} query, then t_j comes before t_k in the ordered sequence of tuples.

Two search agents interacting with an encrypted search provider is given by the following example.

[4]This capability could theoretically be employed by the adversary to decrease the entropy of encrypted search activities.

[5]For example, the IP address of the search agent.

Example 1 In Figure 1, we depict the following situation. There are two search agents, denoted by SA_1 and SA_2, generating plaintext queries expressing some *information need* that is to be met by the encrypted search provider.

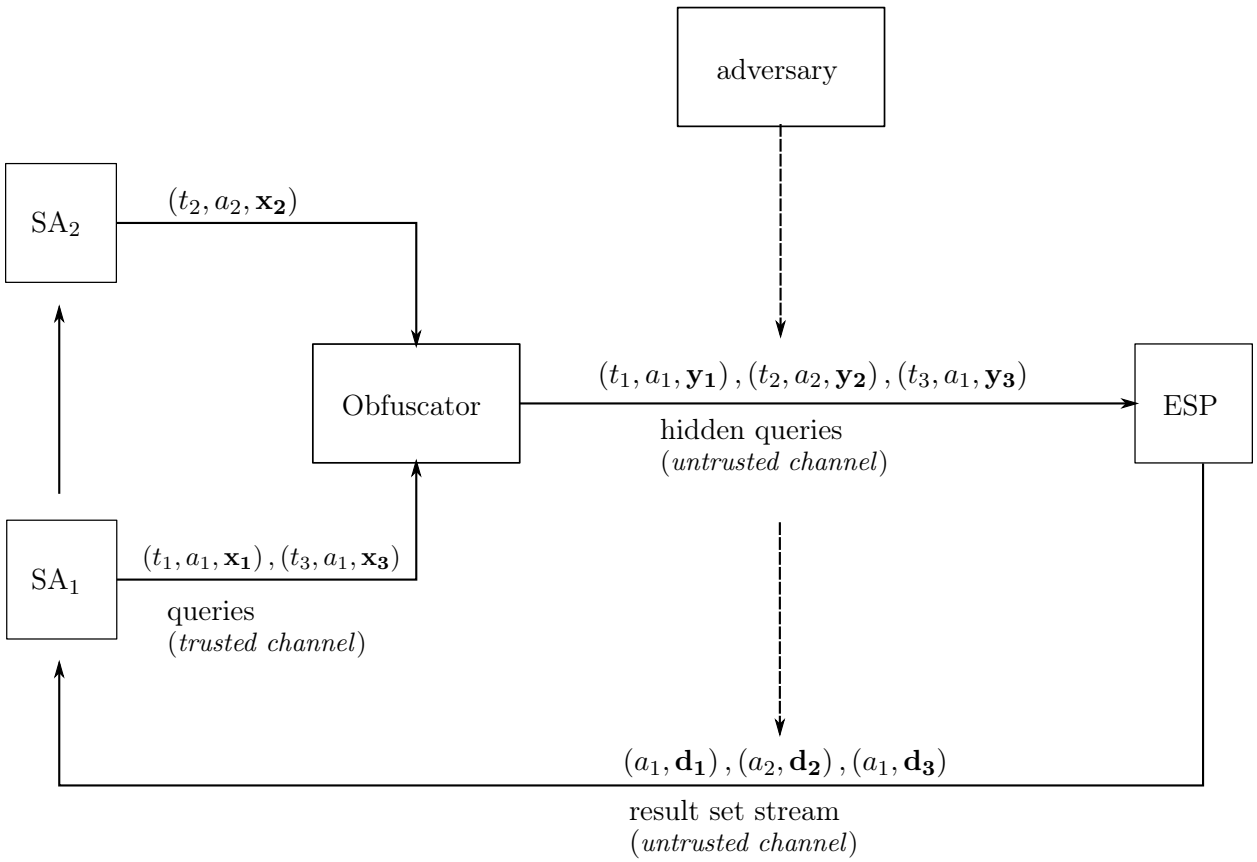
search agent a_1 submits two plaintext queries, $\mathbf{x_1}$ at time t_1 and $\mathbf{x_3}$ at time t_3. These queries are transformed by the obfuscator respectively into the hidden queries $\check{\mathbf{x}}_\mathbf{1}$ and $\check{\mathbf{x}}_\mathbf{2}$ which are then sent to the encrypted search provider over the *untrusted* communications channel. Similiarly, search agent a_2 submits a plaintext queries $\mathbf{x_2}$ at time t_2 which is transformed into $\check{\mathbf{x}}_\mathbf{2}$ and sent to the encrypted search provider.

The encrypted search provider receives these queries and generates result sets that satisfy the *obfuscated* information needs represented by the hidden queries, where $\check{\mathbf{d}}_\mathbf{j}$ satisfies $\check{\mathbf{x}}_\mathbf{j}$ for $j = 1, 2, 3$.

The adversary observes the hidden query and result set streams flowing across the *untrusted* communication channels and attempts to ascertain patterns or regularities that may compromise confidentaility.

The adversary may be an authorized search agent if it is attempting to compromise the query privacy of other search agents.

Figure 1: Two search agents submitting a query to the encrypted search system where a simple substitution cipher is being used.



The system has several characteristics given by the following table.

Table 1: The *known parameters* of the encrypted search system.

param	sup	description
λ	$\mathbb{R}_{>0}$	The *mean arrival rate* of the plaintext queries in the time series.
μ	$\mathbb{R}_{>0}$	The *mean* number of search keys per query in the plaintext time series.
u	$\mathbb{Z}_{>0}$	The *maximum* number of search keys per query in the plaintext time series.
m	$\mathbb{Z}_{>0}$	The number of unique *plaintext* search keys in the plaintext time series.
N	$\mathbb{Z}_{>0}$	The number of unique *confidential documents* on the ESP.
θ	$\mathbb{R}_{>0}$	The mean number of *documents* in the result sets.[6]

To transmit the hidden queries across the *untrusted* channel, some encoding is needed. The hidden query encoder is given by the following definition.

Definition 3.15. *The encoder of the set of trapdoors* $\check{\mathbf{x}}$ *consisting of trapdoors is given by*

$$\texttt{encode}\colon [\mathbb{Y}] \mapsto \mathbb{B}\,, \tag{3.13}$$

which produces a uniquely decodable bit string.

The result sets encoder is given by the following definition.

Definition 3.16. *The encoder of the set of document identifiers* $[\mathbb{D}]$ *is given by*

$$\texttt{encode}\colon [\mathbb{D}] \mapsto \mathbb{B}\,, \tag{3.14}$$

which produces a uniquely decodable bit string.

Theorem 3.1. *The average bit rate of the sum of the hidden query and result set streams is given by*

$$\mathcal{O}\left(\lambda(m\mu + \theta)\right)\,, \tag{3.15}$$

where λ *is the* expected *query arrival rate.*

Proof. The *expected* bit length of the j^{th} hidden query is given by

$$\mathbb{E}\left[\texttt{BL}\left(\texttt{encode}(\mathbb{Y}_\text{j})\right)\right] + \mathcal{O}(1)\,. \tag{a}$$

The constant is the fixed number of bits needed to encode data such as the *time stamp* of a hidden query and the *identifier* of the search agent (such as an IP address) that submitted it. Each trapdoor is coded by a fixed number of m bits, and there is expected to be μ trapdoors per hidden query. Thus, the *expected* bit length of the encoding of $\mathbf{Y_j}$ is given by

$$\mu m + \mathcal{O}(1)\,. \tag{b}$$

The *expected* bit length of the j^{th} result set is given by

$$\mathcal{O}(\theta)\,, \tag{c}$$

where θ is the expected number of documents relevant per hidden query. The sum of Equations b and c is given by

$$\mathcal{O}(\theta) + \mu m + \mathcal{O}(1)\,. \tag{d}$$

The number of hidden query arriving per second is given by λ, and thus the total query rate is given by

$$\lambda\left(\mathcal{O}(\theta) + \mu m + \mathcal{O}(1)\right)\,. \tag{e}$$

□

The *primary* interest of the adversary is in extracting *information* from the stream of hidden queries going from obfuscator to the encrypted search provider and from the stream of result sets going from the encrypted search provider to the search agents. For instance, the adversary may use a known-plaintext attack to map the trapdoors to their plaintext counterparts to ascertain the *confidential* information needs of search agents.

In what follows, we provide a theoretical treatment on the *information disclosure* of the hidden query and result set streams and explore strategies that increase their *entropy* by transforming the streams.

4 Probabilistic model

A probabilistic model is specified by equations involving random variables which make assumptions about how observable data about the system is generated. In what follows, we describe our model and observable data.

The probability mass function is given by the following definition.

Definition 4.1. *Let* X *be some discrete random variable. The probability mass function, denoted by*

$$\mathrm{p}_{\mathrm{X}}(x)\,, \tag{4.1}$$

calculates the probability that X *realizes some value* x*.*

If a random variable X has a probability mass function $\mathrm{p}_{\mathrm{X}}(\,\cdot\,)$, we say that

$$\mathrm{X} \sim \mathrm{p}_{\mathrm{X}}(\,\cdot\,)\,. \tag{4.2}$$

The *joint* probability mass function of $\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{n}$ is given by

$$\mathrm{p}_{\mathrm{X}_1,\cdots,\mathrm{X}_\mathrm{n}}(x_1, \ldots, x_n)\,, \tag{4.3}$$

which calculates the joint probability that $\mathrm{X}_1 = x_1, \ldots, \mathrm{X}_\mathrm{n} = x_n$.

The *arrival rate* is given by the reciprocal of the *inter-arrival* time. In an observed time series $t_1, \ldots, t_n$, the sample mean arrival rate is given by

$$\lambda = \frac{n}{\sum_{j=1}^{n} t_j}\,. \tag{4.4}$$

We model the *inter-arrival* times between successive queries as random variables as given by the following definition.

Definition 4.2. *The* inter-arrival *time between the* $(j-1)$*-th and the* j^{th} *query is a continuous random variable* T_j *with a support set* $\mathbb{R}_{>0}$ *and a mean arrival rate* λ*.*

Remark. An estimator of the *distribution* of the inter-arrival times is provided by time series estimators like *exponential smoothing*.

Suppose it is known that there are k *search agents* that may generate the plaintext queries. In a time series of the queries, the frequency of the search agents may vary. Thus, we may model the distribution as a discrete random variable as given by the following definition.

Definition 4.3. *The search agent responsible for the* j^{th} *query is a discrete random variable* A_j *with a support set given by*

$$\{1, 2, 3, \ldots, k\}\,. \tag{4.5}$$

The distribution of plaintext queries is given by the following definition.

Definition 4.4. *The j^{th} random bag-of-words (set) in the plaintext query time series is denoted by* $\mathbf{X_j}$ *with a support given by*

$$\{\mathbb{X} \in \texttt{powerset}(\mathbb{K}) \,|\, 0 < |\mathbb{X}| \leq p\} \,. \tag{4.6}$$

The process that generates queries may be too complex to model. However, as the famous statistician George Box observed, "... all models are wrong, but some are useful." The adversary does not need an accurate model, only a *useful* model. A very useful model is one which enables the adversary to map the observed trapdoors to their corresponding plaintext counterparts.

The random tuples in the hidden query and result set streams are given by the following definition.

Definition 4.5. *We denote the* random tuple *of the j^{th} plaintext query by*

$$\mathbf{Q_j} = \left(\mathrm{T_j}, \mathrm{A_j}, \mathbf{X_j}\right), \tag{4.7}$$

where

$\mathrm{T_j}$ *is the random inter-arrival time between the $(j-1)$-th and j^{th} queries,*

$\mathrm{A_j}$ *is the random search agent of the j^{th} query, and*

$\mathbf{X_j}$ *is the random set of plaintext bag-of-words in the j^{th} query.*

4.1 Hidden query and result set streams

By Assumption 3.7, the plaintext random tuples $\mathbf{X_1}, \ldots, \mathbf{X_n}$ are not observable. However, these random tuples induce an *observable* hidden query stream.

The *inter-arrival* times between hidden queries is uncertain and therefore we may model them as random variables as given by the following definition.

Definition 4.6. *The* inter-arrival *time between the of the $(j-1)$-th and the j^{th} hidden query is a discrete random variable* $\check{\mathrm{T}}_\mathrm{j}$ *with a support set given by*

$$\{1, 2, 3, \ldots\} \,. \tag{4.8}$$

The *mean* arrival rate of hidden queries is given by

$$n\,\mathbb{E}\left[\frac{1}{\sum_{j=1}^{n} \check{\mathrm{T}}_\mathrm{j}}\right] = \check{\lambda}\,. \tag{4.9}$$

The *mean* arrival rate of hidden queries is not necessarily the same as the *mean* arrival rate of plaintext queries since the obfuscator transforms the incoming plaintext queries to obfuscate encrypted search activities, e.g., it may inject *artificial* hidden queries at uncertain times and at any desired rate. Thus, in general, $\check{\lambda} \neq \lambda$.

The search agents generate the *legitimate* queries. However, the obfuscator may generate artificial queries by either artificial or legitimate search agents to *perturb* the hidden query stream. Regardless, the particular search agent that generated a query in the stream cannot be predetermined and thus we may model this uncertainty as a discrete random variable as given by the following definition.

Definition 4.7. *The search agent responsible for the j^{th} hidden query is a discrete random variable* $\check{\mathrm{A}}_{\mathrm{j}}$ *with a support set given by*

$$\left\{1, 2, 3, \ldots, \check{k}\right\} . \tag{4.10}$$

Since the obfuscator may generate artificial queries for artificial search agents, $\check{k}$ may be larger than k.

By Assumption 3.2, the trapdoors observed are functions of the random plaintext words in the bag-of-words query. Thus, the trapdoors are random sets as given by the following definition.

Definition 4.8. *The trapdoors in the j^{th} hidden query is a random set given by*

$$\check{\mathbf{X}}_{\mathbf{j}} = \texttt{hidden\_query\_generator}(\mathbf{X}) , \tag{4.11}$$

where $\mathbf{X}$ *is some random plaintext query*[7] *and* $\check{\mathbf{X}}_{\mathbf{j}}$ *has a support set given by the power set of*

$$\{1, 2, 3, \ldots, \check{m}\} \tag{4.12}$$

with a mean cardinality given by

$$\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n}\left|\check{\mathbf{X}}_{\mathbf{j}}\right|\right] = \check{\mu} . \tag{4.13}$$

Note that $\check{\mathbf{X}}_{\mathbf{j}}$ does not (necessarily) correspond to $\mathbf{X}_{\mathbf{j}}$ since the obfuscator transforms the incoming plaintext queries to obfuscate encrypted search activities, e.g., it may inject *artificial* hidden queries.

There are N *unique* confidential objects (and $N' - N$ obfuscated or artificial objects) for a total of N' objects. The random set $\check{\mathbf{X}}_{\mathbf{j}}$ induces a distribution of result sets. Since the result sets are uncertain we may model them as random result sets as given by the following definition.

Definition 4.9. *The k^{th} random result set corresponding to the k^{th} random hidden query is denoted by* $\check{\mathbf{D}}_{\mathbf{k}}$ *with a support set given by the power set of*

$$\left\{1, 2, 3, \ldots, \check{N}\right\} \tag{4.14}$$

with a mean cardinality given by

$$\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n}\left|\check{\mathbf{D}}_{\mathbf{j}}\right|\right] = \check{\theta} . \tag{4.15}$$

Given $\check{\mathbf{X}}_{\mathbf{i}} = \check{\mathbf{x}}_{\mathbf{i}}$, $\check{\mathbf{D}}_{\mathbf{i}}$ is *degenerate* since the same hidden query must always map to the same result set (assuming the confidential database is immutable).

In summary, the random tuples in the hidden query and result set streams are given by the following definition.

Definition 4.10. *We denote the* random tuple *of the j^{th} hidden query and the corresponding j^{th} result set by*

$$\check{\mathbf{Q}}_{\mathbf{j}} = \left\langle \check{\mathrm{T}}_{\mathrm{j}}, \check{\mathrm{A}}_{\mathrm{j}}, \check{\mathbf{X}}_{\mathbf{j}}, \check{\mathbf{D}}_{\mathbf{j}} \right\rangle , \tag{4.16}$$

where

$\check{\mathrm{T}}_{\mathrm{j}}$ *is the random inter-arrival time,*

[7] Potentially artificial random query.

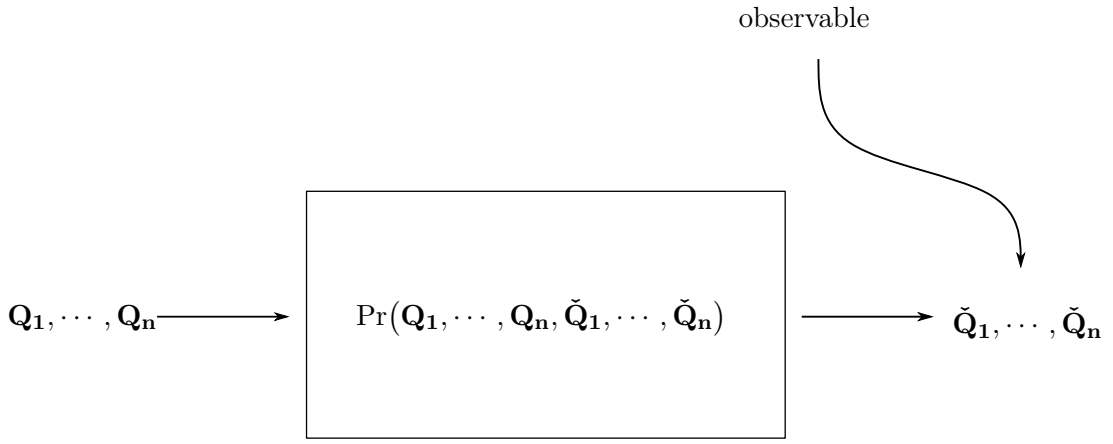


Figure 2: The probability model, where a sequence of *plaintext* random queries is transformed into a sequence of random hidden queries.

$\check{\mathrm{A}}_\mathrm{j}$ *is the random search agent,*

$\check{\mathbf{X}}_\mathbf{j}$ *is the random set of trapdoors, and*

$\check{\mathbf{D}}_\mathbf{j}$ *is the random result set corresponding to* $\check{\mathbf{X}}_\mathbf{j}$.

See Section 4.2 for a description of the joint probability mass function of the random sample $\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}$, which is in general not tractable. The entropy of the distribution is a far more tractable problem and provides a measure of the regularity or predictability that an adversary may extract from the system by observing the hidden query and result set streams.

4.2 Generative model

Consider the time series

$$\mathbf{Q_1}, \ldots, \mathbf{Q_n}, \check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}\,. \tag{4.17}$$

The conditional probability that $\mathbf{Q_n} = \mathbf{q_n}$ and $\check{\mathbf{Q}}_\mathbf{n} = \check{\mathbf{q}}_\mathbf{n}$ given $\mathbf{Q_{<n}} = \mathbf{q_{<n}}$ and $\check{\mathbf{Q}}_\mathbf{<n} = \check{\mathbf{q}}_\mathbf{<n}$ is given by

$$\Pr\left[\check{\mathbf{Q}}_\mathbf{n} = \check{\mathbf{q}}_\mathbf{n}, \check{\mathbf{Q}}_\mathbf{n} = \check{\mathbf{q}}_\mathbf{n} \,|\, \mathbf{Q_{<n}} = \mathbf{q_{<n}}, \check{\mathbf{Q}}_\mathbf{<n} = \check{\mathbf{q}}_\mathbf{<n}\right], \tag{4.18}$$

which is equivalent to

$$\Pr\left[\mathrm{T_n} = t_n, \mathrm{A_n} = a_n, \mathbf{X_n} = \mathbf{x_n}, \check{\mathrm{T}}_\mathrm{n} = \check{t}_n, \check{\mathrm{A}}_\mathrm{n} = \check{a}_n, \check{\mathbf{X}}_\mathbf{n} = \check{\mathbf{x}}_\mathbf{n}, \check{\mathbf{D}}_\mathbf{n} = \check{\mathbf{d}}_\mathbf{n} \,|\, \check{\mathbf{Q}}_\mathbf{<n} = \check{\mathbf{q}}_\mathbf{<n}\right]. \tag{4.19}$$

By the chain rule, this can be rewritten as

$$\begin{aligned}\Pr\left[\check{\mathrm{T}}_\mathrm{n} = t_n, \check{\mathrm{A}}_\mathrm{n} = a_n, \check{\mathbf{X}}_\mathbf{n} = \check{\mathbf{x}}_\mathbf{n} \,|\, \check{\mathbf{X}}_\mathbf{<n} = \check{\mathbf{x}}_\mathbf{<n}\right] = \\ \Pr\left[\check{\mathrm{T}}_\mathrm{n} = t_n \,|\, \check{\mathbf{X}}_\mathbf{<n} = \check{\mathbf{x}}_\mathbf{<n}\right] \times \\ \Pr\left[\check{\mathrm{A}}_\mathrm{n} = a_n \,|\, \check{\mathrm{T}}_\mathrm{n} = t_n, \check{\mathbf{X}}_\mathbf{<n} = \check{\mathbf{x}}_\mathbf{<n}\right] \times \\ \Pr\left[\check{\mathbf{X}}_\mathbf{n} = \check{\mathbf{x}}_\mathbf{n} \,|\, \check{\mathrm{T}}_\mathrm{n} = t_n, \check{\mathrm{A}}_\mathrm{n} = a_n, \check{\mathbf{X}}_\mathbf{<n} = \check{\mathbf{x}}_\mathbf{<n}\right].\end{aligned} \tag{4.20}$$

The *plaintext* time series of queries *induces* the hidden query time series. If a *simple substitution cipher* is used for the queries and agent identifers and the time stamps are unchanged, then the distribution of hidden queries is the same as the plaintext time series except with different labels.

Algorithm 4: Generative model of a hidden query time series

params:
input :
output : A time series of size n drawn randomly from the induced distribution.

function *samplern*
  for $i \leftarrow 1$ **to** n **do**
    sample t_i from $\mathrm{T_i} \,|\, \mathbf{Q_{<i}} = \mathbf{q_{<i}}$;
    sample a_i from $\mathrm{A_i} \,|\, \mathrm{T_i} = t_i, \mathbf{Q_{<i}} = \mathbf{q_{<i}}$;
    sample $\mathbf{x_i}$ from $\mathbf{X_i} \,|\, \mathrm{A_i} = a_i, \mathrm{T_i} = t_i, \mathbf{Q_{<i}} = \mathbf{q_{<i}}$;
    $\check{a}_i \leftarrow$ some anonymizer, like a mixnet;
    $\check{t}_i \leftarrow$ something that delays sending hidden query up to some limit;
    $\check{\mathbf{x}}_\mathbf{i} \leftarrow$ `hidden_query_generator`$(\mathbf{x_i} \,|\, \text{parameters})$;
    $\check{\mathbf{d}}_\mathbf{i} \leftarrow$ `hidden_query_mapper`$(\check{\mathbf{x}}_\mathbf{i} \,|\, \text{parameters})$;
  end

5 Entropy and information

Suppose a search agent has some random information need J coming from some countable set $\mathbb{J}$ with a probability mass given by $\mathrm{p_J}(\,\cdot\,)$. If the adversary could ask "yes" or "no" questions about the search agent['s] information need, an *expected lower-bound* on the number of questions required to determine the information need $j \in \mathbb{J}$ is given by

$$\mathcal{H}(\mathrm{J}) = -\sum_{j \in \mathbb{J}} \mathrm{p_J}(j) \log_2 \mathrm{p_J}(j)\,. \tag{5.1}$$

This is known as the *entropy* of the random variable J. The entropy measures the amount of "uncertainty" about what value J realizes. The greater the entropy, the greater the uncertainty and therefore the greater the number of questions the adversary needs on average to determine the information need.

For instance, to minimize the number of questions, the adversary could ask the search agent whether the information need is $j^{(1)} = \arg\max_{j \in \mathbb{J}} \mathrm{p_J}(j)$. This is the information need with probability $\mathrm{p_J}\big(j^{(1)}\big)$. If not, the adversary can ask the search agent whether the information need is $j^{(2)} = \arg\max_{j \in \mathbb{J} \setminus \{j^{(1)}\}} \mathrm{p_J}(j)$. The information need is

The greater the entropy, the more difficult it is to predict the information need of a search agent. More generally, the greater the entropy, the fewer patterns there are in the activities of an encrypted search system. In what follows, we provide a more rigorous mathematical treatment on the entropy of the encrypted search system.

The entropy is given by the following definition.

Definition 5.1 (Entropy)**.** *The entropy of the j^{th} random tuple $\check{\mathbf{Q}}_\mathbf{j}$ is given by*

$$\mathcal{H}\big(\check{\mathbf{Q}}_\mathbf{j}\big) = \mathbb{E}\left[\log_2 \mathrm{p}_{\check{\mathbf{Q}}_\mathbf{j}}\big(\check{\mathbf{Q}}_\mathbf{j}\big)\right], \tag{5.2}$$

where $\mathrm{p}_{\check{\mathbf{Q}}_\mathbf{j}}(\,\cdot\,)$ is the marginal distribution of $\check{\mathbf{Q}}_\mathbf{j}$.

Definition 5.2 (Conditional entropy)**.** *The conditional entropy of the j^{th} random tuple $\check{\mathbf{Q}}_\mathbf{j}$ given the previous $j-1$ random tuples $\check{\mathbf{Q}}_{\mathbf{<j}} \equiv \check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{j-1}$ is given by*

$$\mathcal{H}\big(\check{\mathbf{Q}}_\mathbf{j} \,|\, \check{\mathbf{Q}}_{\mathbf{<j}}\big) = \mathbb{E}\left[\log_2 \mathrm{p}_{\check{\mathbf{Q}}_\mathbf{j} \,|\, \check{\mathbf{Q}}_{\mathbf{<j}}}\big(\check{\mathbf{Q}}_\mathbf{j} \,|\, \check{\mathbf{Q}}_{\mathbf{<j}}\big)\right]. \tag{5.3}$$

Definition 5.3 (Joint entropy)**.** *The joint entropy of* $\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}$ *is given by*

$$\mathcal{H}(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}) = \mathcal{H}\left(\check{\mathbf{Q}}_{\mathbf{1}}\right) + \sum_{j=2}^{n} \mathcal{H}\left(\check{\mathbf{Q}}_{\mathbf{j}} \,|\, \check{\mathbf{Q}}_{<\mathbf{j}}\right) . \tag{5.4}$$

The joint entropy is less than (or equal to) the sum of the marginal entropies as given by

$$\mathcal{H}\big(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}\big) \leq \sum_{j=1}^{n} \mathcal{H}\big(\check{\mathbf{Q}}_{\mathbf{j}}\big) . \tag{5.5}$$

and only obtains equality if $\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}$ are statistically independent. If they are independent and identically distributed, then the joint entropy is given by

$$\mathcal{H}\big(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}\big) = n\, \mathcal{H}(\mathbf{Q}) . \tag{5.6}$$

Postulate 5.1 (Optimal compressor)**.** *The* entropy $\mathcal{H}\big(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}\big)$ *is equivalent to the* expected *bit length produced by an optimal* lossless *compressor's output given the encoding of the random tuples as given by*

$$\mathcal{H}\left(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}\right) = \mathbb{E}\left[\mathtt{BL}\Big(\mathtt{compress}^*\big(\mathtt{encode}(\check{\mathbf{Q}}_{\mathbf{1}}) + \cdots + \mathtt{encode}(\check{\mathbf{Q}}_{\mathbf{n}})\big)\Big)\right] , \tag{5.7}$$

where $+$ *is the concatenation operation,* $\mathtt{encode}$ *is an arbitrary encoding of tuples,* $\mathtt{compress}^*$ *is an optimal compressor of the sequence, and* $\mathtt{BL}(x)$ *is the bit length of* x*.*

The particular codes chosen by the encoder is irrelevant with respect to the entropy of the streams[8]

If $\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}$ are independent and identically distributed, then the joint entropy is given by

$$\mathcal{H}\left(\check{\mathbf{Q}}_{\mathbf{1}}, \ldots, \check{\mathbf{Q}}_{\mathbf{n}}\right) = n\, \mathbb{E}\left[\mathtt{BL}\left(\mathtt{compress}^*\left(\mathtt{encode}\left(\check{\mathbf{Q}}\right)\right)\right)\right] . \tag{5.8}$$

The *information* conveyed by a message is the *reduction in uncertainty.* The information is given by

$$\begin{aligned} \mathcal{I}(\mathbf{Q}_{\leq \mathbf{n}} \,|\, \check{\mathbf{Q}}_{\leq \mathbf{n}}) &= \mathcal{H}(\mathbf{Q}_{\leq \mathbf{n}}) - \mathcal{H}(\mathbf{Q}_{\leq \mathbf{n}} \,|\, \check{\mathbf{Q}}_{\leq \mathbf{n}}) && (5.9) \\ &= \mathcal{H}(\mathbf{Q}_{\leq \mathbf{n}}) + \mathcal{H}(\check{\mathbf{Q}}_{\leq \mathbf{n}}) - \mathcal{H}(\mathbf{Q}_{\leq \mathbf{n}}, \check{\mathbf{Q}}_{\leq \mathbf{n}}) . && (5.10) \end{aligned}$$

No information is conveyed about the indicated random variables if the hidden queries and plaintext queries are uncorrelated. However, it is possible the hidden queries are correlated with other factors not incorporated into the probabilistic model.

Conversely, if the hidden queries obtain maximum entropy, there are *no patterns* and thus it is not possible for it to be correlated with any other hypothetical factor. For this reason, optimal confidentiality is obtained by the maximum entropy distribution.

[8] An *optimal* coder is informative about the underlying distribution and thus efficient codes are not necessarily even sought in the context of encrypted search.

5.1 Principle of maximum entropy

Given the constraints of the system, the hidden queries necessarily convey *some* information about the plaintext queries. However, the greater the *entropy*, the fewer regularities and patterns in the encrypted search system. The maximum entropy given the system constraints is given by the following theorem.

Theorem 5.1 (Constrained maximum entropy)**.** *The* maximum entropy *of a sequence of random tuples* $\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}$ *subject to the constraints of the communications model is given by*

$$\mathcal{H}^*(\lambda, k, N, M, n, p) = n \left(\mathcal{H}^*(\check{\mathrm{T}} \,|\, \lambda) + \mathcal{H}^*(\check{\mathrm{A}} \,|\, k) + \mathcal{H}^*(\check{\mathbf{X}} \,|\, M, p) + \mathcal{H}^*(\check{\mathbf{D}} \,|\, N) \,. \right) \tag{5.11}$$

Proof.

$$\mathcal{H}^*(\lambda, k, N, M, n, p) = \sum_{i=1}^{n} \left(\mathcal{H}^*(\check{\mathrm{T}} \,|\, \lambda) + \mathcal{H}^*(\check{\mathrm{A}} \,|\, k) + \mathcal{H}^*(\check{\mathbf{X}} \,|\, M, p) + \mathcal{H}^*(\check{\mathbf{D}} \,|\, N) \,. \right) \tag{a}$$

Since $\mathcal{H}(\check{\mathbf{Q}}_\mathbf{j}, \check{\mathbf{Q}}_\mathbf{k}) \leq \mathcal{H}(\check{\mathbf{Q}}_\mathbf{j}) + \mathcal{H}(\check{\mathbf{Q}}_\mathbf{k})$ for any $j \neq k$, to maximize entropy, we seek to maximize the independence without violating any constraints.

The random tuples $\left(\check{\mathrm{T}}_\mathrm{j}, \check{\mathrm{A}}_\mathrm{j}, \check{\mathbf{X}}_\mathbf{j}\right)$ for $j = 1, \ldots, n$ are independently distributed. Also, since we are interested in the *maximum entropy* distribution, the random tuples are identically distributed.

Continuing on, the components in the random tuple, $\check{\mathrm{T}}$, $\check{\mathrm{A}}$, $\check{\mathbf{X}}$, and $\check{\mathbf{D}}$ are independently distributed. Thus, the maximum entropy has a form given by

$$\mathcal{H}^*(\lambda, \mu, k, m, n) = n \left(\mathcal{H}^*(\mathrm{T} \,|\, \lambda) + \mathcal{H}^*(\mathrm{A} \,|\, k) + \mathcal{H}^*(\mathrm{N}, \mathbf{X} \,|\, \mu, m) \right) \,. \tag{b}$$

□

Theorem 5.2. *The maximum entropy* $\mathcal{H}^*(\check{\mathrm{T}} \,|\, \lambda)$ *is given by*

$$\mathcal{H}^*(\check{\mathrm{T}} \,|\, \lambda) = 1 + \ln \frac{1}{\lambda} \,. \tag{5.12}$$

Proof. The continuous random variable $\check{\mathrm{T}}_\mathrm{j}$ generates *inter-arrival* times for queries. The *arrival rate* is specified as λ, and thus $\mathbb{E}[\check{\mathrm{T}}_\mathrm{j}] = \frac{1}{\lambda}$.

The exponential distribution with a mean $\frac{1}{\lambda}$ and a support $\mathbb{R}_{>0}$ is the *maximum entropy distribution* in which these constraints hold,

$$\check{\mathrm{T}}_\mathrm{j} \sim \mathrm{EXP}(\lambda) \tag{a}$$

for $j = 1, \ldots, n$, which has an entropy

$$1 + \ln \frac{1}{\lambda} \,. \tag{b}$$

□

We assume the inter-arrival times are continuous. To store the inter-arrival times on a computer, we must *quantize* them.

Theorem 5.3. *The* optimal *compression for exponentially distributed inter-arrival times with a* precision τ *has an expected lower-bound given by*

$$\mathcal{H}(\check{\mathrm{T}} \,|\, \lambda, \tau) = \frac{1}{\lambda\tau} \log_2 \frac{1}{\lambda\tau} + \left(\frac{1}{\lambda\tau} - 1 \right) \log_2 \left(\frac{1}{\lambda\tau} - 1 \right) \tag{5.13}$$

which asymptotically obtains

$$\lim_{\lambda\tau \to 0} \mathcal{H}(\check{\mathrm{T}} \,|\, \lambda, \tau) = \log_2 \frac{1}{\lambda} + \log_2 \frac{1}{\tau} + \log_2 \mathrm{e} \,. \tag{5.14}$$

Proof. Let $\mathrm{N}(\tau)$ be geometrically distributed with a parameter $p = \lambda\tau$ where $\lambda > 0$ and $\tau > 0$ is an interval of time,

$$\mathrm{N}(\tau) \sim \mathrm{GEO}(p = \lambda\tau)\,, \tag{a}$$

As $\tau \to 0$, $\mathrm{N}(\tau)$ converges in distribution to the exponential distribution with an arrival rate λ.

Thus, we may choose a suitably small τ (the smaller, the more accurately we code the inter-arrival times) and use the entropy of the *geoemtric distribution*, e.g., for a given τ, the optimal compressor has a lower bound given by the entropy

$$\mathcal{H}(\mathrm{N}(\tau)) = \frac{-(1-p)\log_2(1-p) - p\log_2 p}{p}\,, \tag{b}$$

where $p = \lambda\tau$. After simplification, the result follows. □

Theorem 5.4 (Solution for $\mathcal{H}^*(\check{\mathrm{A}}\,|\,k)$)**.** *The* maximum entropy *subject to the constraints of the communications model is given by*

$$\mathcal{H}^*(\check{\mathrm{A}}\,|\,\lambda) = \log_2 k\,. \tag{5.15}$$

Proof. Assigning a unique integer (label) in the set $\{1, 2, \ldots, k\}$ to each of the k search agents, the *discrete uniform* distribution is the *maximum entropy distribution*,

$$\check{\mathrm{A}} \sim \mathrm{DU}(k) \tag{a}$$

for $j = 1, \ldots, n$. The probability mass function is given by

$$\mathrm{p}_{\check{\mathrm{A}}}(a\,|\,k) = \frac{1}{k}\,\mathbb{1}_{\mathrm{k}\in\{1,\ldots,\mathrm{k}\}} \tag{b}$$

with an entropy given by

$$\mathcal{H}(\mathrm{A}\,|\,k) = \log_2 k\,. \tag{c}$$

□

Solution for $\mathcal{H}^*(\check{\mathbf{X}}\,|\,M, p)$.

Solution for $\mathcal{H}^*(\mathbb{D}\,|\,N)$ The *maximum entropy* system has a distribution given by the following corollary.

Corollary 5.4.1. *The* maximum entropy *system has a* random tuple *distribution given by*

$$(\mathrm{T}, \mathrm{A}, \mathbf{Y}) \sim \mathrm{p}_{\mathrm{T},\mathrm{A},\mathbf{Y}}(t, a, \mathbf{y}\,|\,\lambda, \mu, k, m) = \lambda(1-\lambda)^{t-1} \times \frac{1}{k} \times \frac{1}{\mu}\left(1 - \frac{1}{\mu}\right)^{\alpha-1} 2^{-\alpha)m}\,, \tag{5.16}$$

where $\mu > 1$, $0 < \lambda < 1$, $k \geq 1$, *and* $\alpha = \mathtt{dim}(\mathbf{y}) \geq 1$.

If we generate n tuples from this distribution and *losslessly* compress the results with an optimal compressor, it is expected that the bit length of the compressor's output obtains the lower-bound given by the *maximum entropy*.

If the parameters of the *maximum entropy* $\mathcal{H}_n^*$ distribution is not known, then it may be estimated by the following theorem.

Theorem 5.5. *The maximum likelihood estimator of $\mathcal{H}_n^*$ is given by*

$$\hat{\mathcal{H}}_n^* = \mathcal{H}_n^* \left(\hat{m}, \hat{k}, \hat{\lambda}, \hat{\mu} \right) , \tag{5.17}$$

where $\hat{m}$ is the maximum likelihood estimator of m given by the number of unique trapdoors in the sample, the maximum likelihood estimator[9] of k is given by

$$\hat{k} = \max a_1, \ldots, a_n , \tag{5.18}$$

the maximum likelihood estimator of λ is given by

$$\hat{\lambda} = n \left[\sum_{i=1}^{n} t_i \right]^{-1} , \tag{5.19}$$

and the maximum likelihood estimator of μ is given by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \texttt{dim}(\mathbf{x_i}) . \tag{5.20}$$

Proof. Given the distribution of the *maximum entropy*, where T is geometrically distributed with *arrival rate* λ, the UMVU estimator of λ is given by Equation 5.19.

Continue on in the same fashion for the other random variables. □

Using the large sample approximation, the maximum likelihood estimator $\hat{\mathcal{H}}_n^*$ is normally distributed as given by

$$\hat{\mathcal{H}}_n^* \sim \mathcal{N} \left(n \, \mathcal{H}_1^*, \frac{1}{n} \operatorname{Var} \left[\hat{\mathcal{H}}_1^* \right] \right) . \tag{5.21}$$

We assume a sufficiently large sample of size n is available so that we may assume the variance of the sampling distribution of $\mathrm{S_n}$ is small.

If the entropy of the system $\mathcal{H}_n$ is not known, then it may be estimated by the following theorem.

Theorem 5.6. *A* positive biased *estimator of the entropy $\mathcal{H}_n$ is given by*

$$\hat{\mathcal{H}}_n = \texttt{BL}(\texttt{compress}\,(\texttt{concat}(\texttt{encode}(\mathbf{q_1}), \ldots, \texttt{encode}(\mathbf{q_n})))) , \tag{5.22}$$

where $q_j = \left(t'_j, a'_j, \mathbf{x'_j}, \mathbf{d_j} \right)$ is the i^{th} observed tuple and `compress` *is a* near-optimal *lossless compressor.*

Proof. We have the following corollary.

Corollary 5.6.1. *with an asymptotic form given by*

$$\mathcal{H}_n^*(m, k, \lambda, \mu) = n \left(\log_2 \frac{\mu k}{\lambda} + \mu(m+1) + \text{const} \right) \tag{a}$$

where $\text{const} = 2 \log_2 \mathrm{e}$ *and m, k, λ, μ are the system parameters.*

By Postulate 5.1, an optimal compressor `compress`* is expected to obtain the lower-bound $\mathcal{H}_n$. Plugging in a sub-optimal compressor will produce an estimate of this lower-bound. And, since the compressor is not optimal, it produces estiamtes larger than the true lower bound, i.e., it is a positive biased estimator. □

[9] The UMVU estimator of k is given by $\bar{k} = \frac{n+1}{n} \hat{k}$.

Performance measure The performance measure of an encrypted search system is given by the following definition.

Definition 5.4. *The performance of a encrypted search system with entropy $\mathcal{H}_n$ is given by*

$$e(m, k, \lambda, \mu) = \frac{\mathcal{H}_n(m, k, \lambda, \mu)}{\mathcal{H}_n^*(m, k, \lambda, \mu)}. \tag{5.23}$$

A system that obtains $e(\cdot) = 1$ is said to disclose *minimum information*. Conversely, a system which obtains $e(\cdot) = 0$ (the degenerate distribution) is said to disclose *maximum information*.[10]

If $e(m, k, \lambda, \mu)$ is not known, then it may be estimated by the following statistic.

Corollary 5.6.2. *By Equations 5.17 and 5.22, an estimator of the performance measure is given by*

$$\hat{e} = \frac{\hat{\mathcal{H}}_n}{\hat{\mathcal{H}}_n^*}. \tag{5.24}$$

Proof. If the maximum likelihood estimator of a parameter θ is $\hat{\theta}$, by the plugin principle the maximum likelihood estimator of a parameter $\mathrm{g}(\theta)$ is given by $\hat{g} = \mathrm{g}(\hat{\theta})$. □

A central idea in this paper is that *compression* is equivalent to *probabilistic* data modeling since a good compressor tends to be a good predictor of the data. In fact, many compression algorithms essentially estimate conditional probability mass functions of the data (so that shorter codes may be assigned to more probably symbols). Thus, we may delegate the data modeling task to good compressors of encrypted search activities. The more noise-like the activities are, the larger the compressed output.

6 Maximum entropy system

In this section, we characterize the maximum entropy distribution for encrypted search systems. Given system constraints—such as the number of search agents k, vocabulary size m, and query arrival rate λ—we derive the probability distributions that maximize entropy subject to these constraints.

Adversary Model. The adversary observes the hidden query stream across the untrusted channel, including timestamps, trapdoors, and result sets. We assume a passive adversary who can record but not inject or modify queries. The adversary may have prior knowledge of system parameters (e.g., k, m, λ) but not the plaintext queries themselves. A more powerful adversary with side-channel access or traffic manipulation capabilities may achieve better inference, but analyzing such adversaries is beyond our current scope.

Theorem 6.1 (Total System Entropy)**.** *The total entropy of the hidden query stream over n observations is given by*

$$\mathcal{H}(\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}) = \sum_{j=1}^{n} \mathcal{H}\left(\check{\mathbf{Q}}_\mathbf{j} \,|\, \check{\mathbf{Q}}_{<\mathbf{j}}\right), \tag{6.1}$$

which, under the assumption of independent and identically distributed queries, simplifies to

$$\mathcal{H}(\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}) = n \cdot \mathcal{H}(\check{\mathbf{Q}}). \tag{6.2}$$

[10] The Shannon information of a system in which $e(\cdot) = 0$ is 0, but we are measuring this with respect to an adversary being able to predict what will happen, so my disclose maximum information we mean to suggest that it is predictable.

Proof. The first equality follows directly from the chain rule for joint entropy (Definition 5.4). The simplification under i.i.d. assumptions follows because $\mathcal{H}(\check{\mathbf{Q}}_\mathbf{j} \mid \check{\mathbf{Q}}_{<\mathbf{j}}) = \mathcal{H}(\check{\mathbf{Q}}_\mathbf{j}) = \mathcal{H}(\check{\mathbf{Q}})$ when tuples are independent. □

The probability model with maximum entropy is given by the following theorem.

Theorem 6.2. *The distribution of the k search agents identities in the hidden query time series are independently and uniformly distributed,*

$$\mathrm{A_j} \sim \mathrm{DU}\,(k)\ . \tag{6.3}$$

Proof. We assume that the adversary knows there are k unique search agents. □

For a maximum entropy system, the *unary coder* is optimal if the inter-arrival times are geometrically distributed as

$$\mathrm{T_j} \sim \mathrm{GEO}\left(\lambda = \frac{1}{2}\right)\ , \tag{6.4}$$

the *unary coder* is optimal if the number of trapdoors per hidden query is geometrically distributed as

$$\mathrm{N_j} \sim \mathrm{GEO}\,(\mu = 2)\ , \tag{6.5}$$

and m bits per trapdoor is the *optimal* coder if the occurrences of trapdoors are uniformly distributed as

$$\mathrm{Y_j} \sim \mathrm{DU}\,(0, 2^m - 1) \tag{6.6}$$

for $j = 1, \ldots, n$.

Consider a practical system with the following parameters:

1. The query arrival rate $\lambda' = \frac{1}{2}$.
2. The mean number of words per query is μ'.
3. There are k' unique search agents.
4. $\mathbf{X'_j}$ is the random vector of trapdoors in which each component has m' possibilities with replacement,
5. $\mathrm{W_j}$ is the random dimension of the random vector $\mathbf{D_j}$ with a mean θ, and
6. $\mathbf{D_j}$ is the random vector of results corresponding to $\mathbf{Y_j}$ in which each component has N possibilities without replacement.

Table 2 is the *optimal* coder if the occurrences of search agents are uniformly distributed as

$$\mathrm{A_j} \sim \mathrm{DU}\,(\{a_1, a_2, \ldots, a_k\})\ , \tag{6.7}$$

the *unary coder* is optimal if *arrival rates* are geometrically distributed as

$$\mathrm{T_j} \sim \mathrm{GEO}\left(\lambda = \frac{1}{2}\right)\ , \tag{6.8}$$

the *unary coder* is optimal if the number of trapdoors per hidden query is geometrically distributed as

$$\mathrm{N_j} \sim \mathrm{GEO}\,(\mu = 2)\ , \tag{6.9}$$

and m bits per trapdoor is the *optimal* coder if the occurrences of trapdoors are uniformly distributed as

$$\mathrm{Y_j} \sim \mathrm{DU}\,(0, 2^m - 1) \tag{6.10}$$

for $j = 1, \ldots, n$.

Encoding the hidden queries and result sets for transmission e.g., the unary encoder given by Table 3, each trapdoor $y \in \mathbf{y}$ is encoded by a *bit string* of fixed-length m, and a is encoded.

Theorem 6.3. *The* expected *optimally compressed bit length of a hidden query is given by*

$$\ell = \frac{1}{\lambda} + p + \mu(1 + m)\,. \tag{6.11}$$

Proof. The expected bit length is given by the expectation of the bit length of the encoder on a random tuple as given by

$$\ell = \mathbb{E}\left[\mathtt{BL}(\mathtt{encode}(\mathrm{T_j}, \mathrm{A_j}, \mathbf{X_j})\right]\,. \tag{a}$$

We may look at how each of these random variables are coded separately.

The *time stamp* is coded by the unary coder given by Table 3, where an integer $n > 0$ has a bit length n. Thus, the *expected* bit length is given by

$$\mathbb{E}[\mathrm{T_j}] = \frac{1}{\lambda}\,, \tag{b}$$

where the mean *inter-arrival* time is given by the reciprocal of the *arrival rate* λ, which is a characteristic of the encrypted search system.

The *search agents* are coded by fixed-length bit strings of size p. Thus, the expected bit length of the code for a search agent is p.

The number of trapdoors $\mathtt{dim}(\mathbf{Y_j})$ is coded by the unary coder given by Table 3, where an integer $n > 0$ has a bit length n. Thus, the *expected* bit length is given by

$$\mathbb{E}[\mathrm{N_j}] = \mu\,, \tag{c}$$

where μ is a characteristic of the encrypted search system.

The *trapdoors* are coded by *bit strings* of fixed-length m and there are expected to be μ trapdoors per hidden query, thus the expected bit length of $\mathbf{y}$ is given by

$$\mu m\,. \tag{d}$$

Concatenating these codes together produces an encoding with an expected bit length given by

$$\frac{1}{\lambda} + p + \mu(1 + m)\,. \tag{e}$$

□

7 Maximum Entropy Under Constraints

In this section, we derive the maximum entropy distribution for encrypted search activities subject to realistic system constraints. The maximum entropy principle, formalized by Jaynes[10], states that subject to precisely stated prior information, the probability distribution that best represents the current state of knowledge is the one with the largest entropy.

Table 2: Code for search agents

Search agent	Code
SA_1	$0\,0\,0\,0_2$
SA_2	$0\,0\,1\,0_2$
SA_3	$0\,1\,0\,0_2$
SA_4	$0\,1\,1\,0_2$
SA_5	$1\,0\,0\,0_2$
SA_6	$1\,0\,1\,0_2$

Table 3: Unary code for inter-arrival time

τ	Code
1	1_2
2	$0\,1_2$
3	$0\,0\,1_2$
4	$0\,0\,0\,1_2$
5	$0\,0\,0\,0\,1_2$
6	$0\,0\,0\,0\,0\,1_2$
	$\vdots$

7.1 System Constraints

An encrypted search system operates under the following constraints, which we formalize as expectations or support restrictions on the probability distributions:

1. **Query arrival rate**: The mean number of queries per unit time is λ.
2. **Number of search agents**: There are k distinct search agents.
3. **Query size**: The mean number of trapdoors per hidden query is μ.
4. **Trapdoor vocabulary**: There are m possible distinct trapdoors.
5. **Document collection**: There are N distinct documents.
6. **Result set size**: The mean number of documents returned per query is θ.

These constraints reflect observable or known properties of the system that cannot be hidden without fundamentally changing the system's functionality or violating resource constraints.

7.2 Maximum Entropy for Inter-Arrival Times

The inter-arrival time between consecutive queries is constrained by the arrival rate λ.

Theorem 7.1 (Maximum entropy for inter-arrival times)**.** *Subject to the constraint that* $\mathbb{E}[\check{\mathrm{T}}] = 1/\lambda$*, the distribution that maximizes entropy is the exponential distribution:*

$$\check{\mathrm{T}} \sim \mathrm{EXP}(\lambda) \tag{7.1}$$

with entropy

$$\mathcal{H}^*(\check{\mathrm{T}}) = 1 + \ln \frac{1}{\lambda}\,. \tag{7.2}$$

Proof. Among continuous distributions on $\mathbb{R}_{>0}$ with a fixed mean $1/\lambda$, the exponential distribution maximizes differential entropy. This follows from the calculus of variations applied to the entropy functional subject to the mean constraint. The exponential distribution has probability density function

$$\mathrm{f}_{\check{\mathrm{T}}}(t) = \lambda e^{-\lambda t} \tag{a}$$

and differential entropy

$$\mathcal{H}(\check{\mathrm{T}}) = \int_0^\infty -\mathrm{f}_{\check{\mathrm{T}}}(t) \ln \mathrm{f}_{\check{\mathrm{T}}}(t)\, dt = 1 + \ln \frac{1}{\lambda}\,. \tag{b}$$

□

7.3 Maximum Entropy for Search Agent Identities

The search agent identity for each query is constrained by the total number of agents k.

Theorem 7.2 (Maximum entropy for search agent identities)**.** *Subject to the constraint that there are k search agents, the distribution that maximizes entropy is the discrete uniform distribution:*

$$\check{\mathrm{A}} \sim \mathrm{DU}(1, k) \tag{7.3}$$

with entropy

$$\mathcal{H}^*(\check{\mathrm{A}}) = \log_2 k\,. \tag{7.4}$$

Proof. Among discrete distributions on a finite support of size k, the uniform distribution maximizes entropy. The uniform distribution has probability mass function

$$\mathrm{p}_{\check{\mathrm{A}}}(a) = \frac{1}{k} \quad \text{for } a \in \{1, 2, \ldots, k\} \tag{a}$$

and entropy

$$\mathcal{H}(\check{\mathrm{A}}) = \sum_{a=1}^{k} -\frac{1}{k} \log_2 \frac{1}{k} = \log_2 k\,. \tag{b}$$

□

7.4 Maximum Entropy for Hidden Query Cardinality

The cardinality of a hidden query (number of trapdoors) is constrained by the mean μ and typically by a maximum value u.

Theorem 7.3 (Maximum entropy for query cardinality)**.** *Subject to the constraint that $\mathbb{E}[\mathrm{N}_{trap}] = \mu$ where $\mathrm{N}_{trap} \in \{1, 2, \ldots, u\}$, an approximate maximum entropy distribution is the geometric distribution (when u is large):*

$$\mathrm{N}_{trap} \sim \mathrm{GEO}(p) \tag{7.5}$$

where $p = 1/\mu$, with entropy

$$\mathcal{H}^*(\mathrm{N}_{trap}) = \frac{-(1-p)\log_2(1-p) - p \log_2 p}{p}\,. \tag{7.6}$$

Proof. The geometric distribution maximizes entropy among discrete distributions on positive integers with a given mean. For $p = 1/\mu$, the geometric distribution has mean μ and probability mass function

$$\mathrm{p}_{\mathrm{N}}(n) = p(1-p)^{n-1} \quad \text{for } n \geq 1\,. \tag{a}$$

When the maximum value u is large relative to μ, the truncation has negligible effect on the entropy. □

7.5 Maximum Entropy for Trapdoor Selection

The trapdoors within a hidden query are drawn from a vocabulary of size m.

Theorem 7.4 (Maximum entropy for trapdoor selection)**.** *Subject to no constraints beyond the vocabulary size m, the distribution that maximizes entropy for each trapdoor is the discrete uniform distribution:*

$$\mathrm{Y_i} \sim \mathrm{DU}(1, m) \tag{7.7}$$

with entropy per trapdoor

$$\mathcal{H}^*(\mathrm{Y_i}) = \log_2 m\,. \tag{7.8}$$

Proof. Without additional constraints on the relative frequencies of trapdoors, the uniform distribution over the vocabulary maximizes entropy. This gives entropy $\log_2 m$ per trapdoor selection. □

7.6 Maximum Entropy for Result Sets

The result sets are constrained by the document collection size N and mean result set size θ.

Theorem 7.5 (Maximum entropy for result set cardinality)**.** *Subject to the constraint that $\mathbb{E}[\mathrm{N}_{results}] = \theta$ where $\mathrm{N}_{results} \in \{0, 1, \ldots, N\}$, the approximate maximum entropy distribution (for large N) is geometric or Poisson-like.*

7.7 Joint Maximum Entropy

Combining these results, we obtain the maximum entropy for the complete system.

Theorem 7.6 (Joint maximum entropy)**.** *Under the assumption of independence (which maximizes joint entropy), the maximum entropy for n query tuples is:*

$$\begin{aligned}\mathcal{H}^*_n = n\Big[& \mathcal{H}^*(\check{\mathrm{T}}) + \mathcal{H}^*(\check{\mathrm{A}}) + \mathcal{H}^*(\mathrm{N}_{trap}) \\ & + \mathbb{E}[\mathrm{N}_{trap}] \cdot \mathcal{H}^*(\mathrm{Y}) + \mathcal{H}^*(\mathrm{N}_{results}) \\ & + \mathbb{E}[\mathrm{N}_{results}] \cdot \log_2 N\Big]\,.\end{aligned} \tag{7.9}$$

Proof. Since entropy is additive for independent random variables, and we assume each query tuple is independent and identically distributed:

$$\mathcal{H}(\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}) = \sum_{i=1}^{n} \mathcal{H}(\check{\mathbf{Q}}_\mathbf{i}) = n \cdot \mathcal{H}(\check{\mathbf{Q}})\,. \tag{a}$$

Within each tuple, assuming independence of components:

$$\mathcal{H}(\check{\mathbf{Q}}) = \mathcal{H}(\check{\mathrm{T}}) + \mathcal{H}(\check{\mathrm{A}}) + \mathcal{H}(\check{\mathbf{X}}) + \mathcal{H}(\check{\mathbf{D}})\,. \tag{b}$$

The entropy of the hidden query bag depends on both the cardinality and the trapdoor selections:

$$\mathcal{H}(\check{\mathbf{X}}) = \mathcal{H}(\mathrm{N_{trap}}) + \mathbb{E}[\mathrm{N_{trap}}] \cdot \mathcal{H}(\mathrm{Y})\,. \tag{c}$$

Similarly for result sets. □

7.8 Minimum Mutual Information

The minimum mutual information between plaintext queries $\mathbf{Q_1}, \ldots, \mathbf{Q_n}$ and hidden queries $\check{\mathbf{Q}}_\mathbf{1}, \ldots, \check{\mathbf{Q}}_\mathbf{n}$ is achieved when the hidden queries realize the maximum entropy distribution.

Corollary 7.6.1 (Minimum mutual information)**.** *The minimum mutual information is:*

$$\mathcal{I}^{\min}(\mathbf{Q_{1:n}}; \check{\mathbf{Q}}_\mathbf{1:n}) = \mathcal{H}(\mathbf{Q_{1:n}}) + \mathcal{H}_n^* - \mathcal{H}_{\max}(\mathbf{Q_{1:n}}, \check{\mathbf{Q}}_\mathbf{1:n}) \tag{7.10}$$

where $\mathcal{H}_{\max}$ represents the maximum possible joint entropy. When the system achieves maximum entropy, the mutual information equals the inherent correlation required by system functionality.

This provides a lower bound on information leakage determined by the fundamental requirements of the encrypted search system.

8 Increasing the *entropy* of the system

Assuming that the confidential collection of documents is immutable, a given hidden query always maps to the same result set. That is to say, given a hidden query, the corresponding random result set is *degenerate*. Consquently, the *entropy* of the joint distribution of hidden queries and result sets is just the entropy of either distribution,

$$\mathcal{H}(\check{\mathbf{X}}_\mathbf{i}, \check{\mathbf{D}}_\mathbf{i}) = \mathcal{H}(\check{\mathbf{X}}_\mathbf{i}) = \mathcal{H}(\check{\mathbf{D}}_\mathbf{i})\,. \tag{8.1}$$

Thus, the only way to increase the entropy of the result sets is to increase the entropy of the hidden queries. We can thus focus on finding ways to increase the entropy of the hidden queries.

Note that conditional probability distribution of $\mathcal{H}(\check{\mathbf{D}}_\mathbf{i} \,|\, \check{\mathbf{X}}_\mathbf{i}$ is *necessarily* degenerate since a particular hidden query must always map to the same result set. However, $\check{\mathbf{X}}_\mathbf{i} \,|\, \check{\mathbf{D}}_\mathbf{i}$ is not degenerate since many hidden queries may map to the same result set.

Generally, the hidden query stream does not obtain an efficiency of 1. Thus, we propose to *perturb* the stream to increase the entropy. In this section, we cover general strategies for increasing the entropy at some quantifiable cost.[11]

The general approach is to interrupt any regularities or patterns in the original stream with unpredictable *noise*, which has the effect of increasing the entropy but decreasing some other performance measure, like the bandwidth requirements of encrypted search activities or the space complexity of the secure indexes.

8.1 Multiple secure indexes per document

Ideally, every document in the document store will have an equal probability of being referenced in the stream of result sets. However, this is unlikely since some documents are expected to be generally more relevant to the information needs of the search agents.

Similar to homophonic encryption for trapdoors, a document should be given multiple secure index representations (and more representations in the document store) proportional to the reciprocal of its relative frequency in the stream of result sets. However, there are a few problems with this approach:

1. The relative frequency is probably not known a priori.

[11] In general, this is a non-linear optimization problem where we wish to maximize some function of the entropy and other performance measures.

2. This increases both the space complexity and time complexity of the encrypted search system since more objects must be stored and queried.

For each document, construct multiple secure indexes in which each secure index for the given document will have a different representation because they will use different document identifiers[12] and different salts for their trapdoors.

This will improve query privacy in a way similiar to homophic encryption. However, homophonic encryption more efficiently serves this purpose. The primary advantage to having multiple secure indexes per document is that it enables the same plaintext query to return a set of logically equivalent results. This *obscures* a user's search patterns, e.g., different users with similar search patterns may sample the salts differently to make their search patterns appear dissimilar.

Example 2 Let a document A read "Hello world!". Let us represent document A with $N = 3$ secure indexes, denoted by A_1, A_2, and A_3. Then, in the biword model, A_i's trapdoors may be generated from the set given by

$$A_i \leftarrow \texttt{make\_secure\_index}\left(\{(\text{hello} + \text{salt}_i)\ , (\text{world} + \text{salt}_i)\ , (\text{hello world} + \text{salt}_i)\} \mid \beta\right) . \tag{8.2}$$

This method increases the space and time complexities of encrypted search, e.g., the size of the secure index database grows linearly with respect to N.

8.2 Artificial secure indexes

An extension of multiple secure indexes per document is the automatic inclusion of artificial secure indexes. These may be automatically generated from some language model (e.g., trigram language model) such that they are expected to be relevant to an appropriate percentage of queries.

Artificial secure indexes make it more difficult for an adversary to determine which documents retrieved in response to a hidden query are of actual interest while the user who submitted the search can instantly filter out the fake results. For example, the trapdoor of a artificial secure index document identifier tests positively as a member of the *artificial* document class.

8.3 Homophonic encryption

We map the *accuracy* of the adversary with respect to a sample size for various efficiencies in the following analysis. The greater the efficiency (or entropy), the less accurate the mapping is expected to be. At one extreme, we have an efficiency of 0 (minimum entropy) in which 100% of the traffic is successfully mapped after viewing a sample of size 1 and the other extreme we have an efficiency of 1 (maximum entropy) where the accuracy is given by pure random chance and is not correlated with sample size.

Homophonic encryption may be employed to flatten the distribution of trapdoors by giving each plaintext word one or more trapdoors signatures.

By analyzing queries, a reasonable homophonic code can be devised. However, such trapdoors must be relevant to a corresponding set of secure indexes to facilitate encrypted search. Thus, the more substitutions, the larger the secure indexes. If the space requirements are too demanding to create a completely flat marginal distribution, then we can settle on making the distribution more flat.

[12] An automated method for generating unique document identifiers may be accomplished by encrypting, using an invertible encryption scheme, the plaintext document reference with different salts.

Figure 3: Accuracy vs sample size where $N = 1000$
for several different entropies

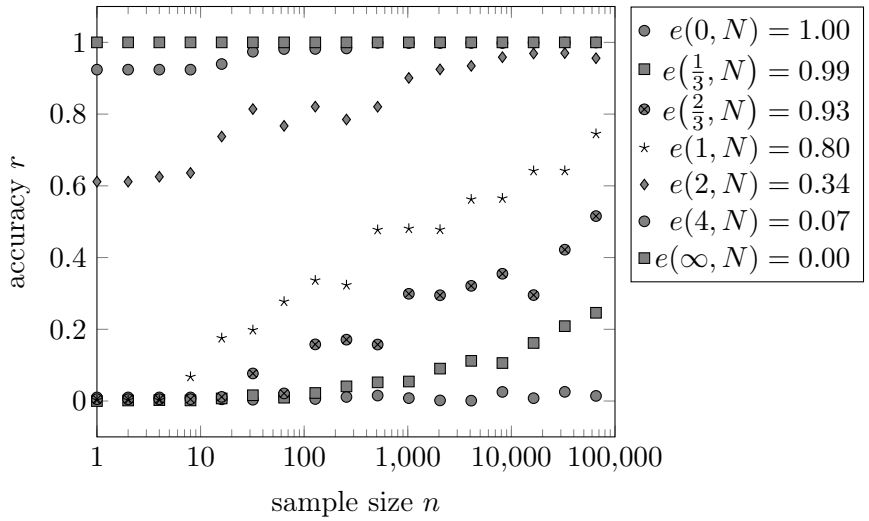
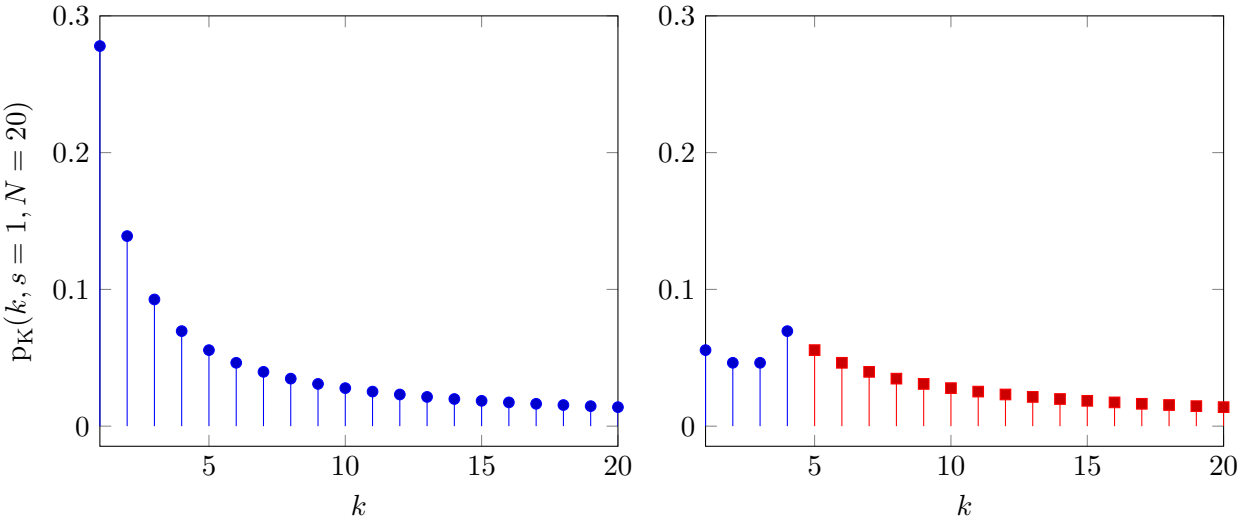


Figure 4: Testing



Flattening the *marginal* distribution does not prevent adversaries from learning mappings since random plain-text words $\mathrm{T_j}$ and $\mathrm{T_k}$ are not necessarily statistically independent and, for instance, may still have a *non-uniform* conditional distribution

$$\mathrm{T_j} \,|\, \mathrm{T_k} \sim \mathrm{p}_{\mathrm{T_j} \,|\, \mathrm{T_k}}(\,\cdot\,)\ . \tag{8.3}$$

Algorithm 5: Homophonic substitution cipher

params: b, maximize the *marginal* entropy of the distribution of trapdoors up to the b-th ranked word.
input : A *plaintext* word x.
output : A trapdoor $\check{x}$ of plaintext word x;

function `substitutions`(x)
  `// Retrieve the` b`-th ranked word and compute its` *`relative frequency`*`.`
  $\beta \leftarrow \mathrm{p_X}\big(\mathtt{Rank}^{-1}(b)\big)$;
  $m \leftarrow 1$;
  `// If the rank of` x `is less than` b`, we must provide more than 1 possible substitution such that the first` b `words are (approximately) uniformly distributed.`
  if $\mathtt{Rank}(x) < b$ **then**
    $m \leftarrow \left\lfloor \frac{\mathrm{p_X}(x)}{\beta} + \frac{1}{2} \right\rfloor$;
  end
  return m;

8.4 Query aggregation

Instead of submitting a single query consisting of k search keys, we can reduce it to y queries where the j^{th} query has k_j search keys for $j = 1, \ldots, y$ such that $k_1 + \cdots + k_y = k$ and then apply the *set-intersection* operation on the y result sets on a *trusted machine*, such as the search agent's host computer rather than the untrusted encrypted search provider.

Remark. A query model that includes the full set-theoretic model[13] may reveal significantly more information about a search agent's information need. Thus, if a full set-theoretic model is desired, a strong case can be made that it should be implemented using a variation of query aggregation, where set-theoretic operations (with possibly the exception of set-intersection) are applied to the result sets on a *trusted* machine rather than the *untrusted* encrypted search provider.

8.5 *Artificial* trapdoors

The effectiveness of artificial trapdoors depends critically on the encrypted search provider's ranking mechanism. In boolean search systems where all query terms must be present for document retrieval, artificial trapdoors will cause no matching documents to be returned, making this technique ineffective. However, for rank-ordered retrieval systems using scoring functions such as BM25 or TF-IDF, artificial trapdoors integrate seamlessly with authentic trapdoors. In such systems, artificial trapdoors affect result sets only when: (1) they trigger false positives in secure indexes due to the approximate nature of structures like Bloom filters, or (2) they coincidentally collide with authentic trapdoors due to the finite trapdoor space.

The probability of collision between artificial and authentic trapdoors is governed by the birthday paradox. With m' authentic trapdoors and m'' artificial trapdoors drawn uniformly from a space of size 2^l, the expected number of collisions is approximately $\frac{m' \cdot m''}{2^l}$. For typical systems with $l \geq 128$ bits, collision probability remains negligible even with thousands of artificial trapdoors.

Suppose that each query has some random number L of *artificial* trapdoors, where

$$\mathrm{L} \sim \mathrm{p}_{\mathrm{L}}(\cdot \,|\, \mu_{\mathrm{L}}) \tag{8.4}$$

such that

$$\mathbb{E}[\mathrm{L}] = \mu_{\mathrm{L}} \,. \tag{8.5}$$

Let the random variable corresponding to the artificial trapdoor be given by

$$\mathrm{Y} \sim \mathrm{p}_{\mathrm{Y}}(\cdot \,|\, m_{\mathrm{Y}}) \,, \tag{8.6}$$

where m_{Y} is the number of unique artificial trapdoors.

The maximum entropy of the joint distribution of L and Y is given by the following theorem.

Theorem 8.1.

$$\mathcal{H}^*(\mathrm{L}, \mathrm{Y}) = \mathcal{H}^*(\mathrm{L} \,|\, \mu_{\mathrm{L}}, \mathrm{Y}) \tag{8.7}$$

Suppose we have l bits per trapdoor, then at maximum we may generate $m'' = 2^l - m'$ artificial trapdoors, where m' are the unique number of bit patterns corresponding to *authentic* trapdoors.

Letting M and T'_{j} for all j be independent, then the mean number of *authentic* and *artificial* trapdoors per query is given by

$$\mu'' = \mu' + \mu'_{\mathrm{L}} \,. \tag{8.8}$$

If we randomize the order of the trapdoors in the hidden queries and let $\mu'_{\mathrm{L}} \to \infty$, then the efficiency of the entire encrypted search system will converge to

$$e(\mu'') = \frac{\mathcal{H}_n(\mathrm{N}, \mathbf{Y} \,|\, \mu'')}{n \, \mathcal{H}^*(\mathrm{N}, \mathbf{Y} \,|\, \mu'')} \,. \tag{8.9}$$

Assume the adversary has a model of the hidden query stream using known-plaintext attacks. Perturbing the hidden query stream by adding *noise* to it may counter known-plaintext attacks. Alternatively, assume that the adversary knows the secret and thus may use a dictionary attack to decipher the trapdoors. Then, adding noise may in some cases obfuscate what a search agent is actually interested in.

To mitigate such information leaks, in general we can look to oblivious RAM for inspiration. oblivious RAM may naively be thought of in the following way: to prevent meaningful statistics from being gathered about a user's activities, whenever an action–a read or write–is performed, include other randomly chosen actions to obscure the user's actual interests or activities.

8.6 *Artificial* hidden queries

Unlike artificial trapdoors which may alter result sets through collisions or false positives, artificial queries are designed to be distinguishable by the search agent while appearing indistinguishable to the adversary. The search agent can filter out artificial queries from result sets using cryptographic tags or sequence numbers, ensuring that authentic queries receive unmodified results. This approach trades space complexity for time and bandwidth: instead of expanding the trapdoor space through homophonic encryption (Section 9.3), we expand the temporal query stream.

The choice of query representation (unigrams, bigrams, trigrams, or longer n-grams) significantly impacts the entropy-confidentiality tradeoff. Unigram queries provide a vocabulary of size $|V|$, yielding at most $\log_2 |V|$ bits of entropy per trapdoor. Bigram queries expand the space to $|V|^2$ possibilities, increasing per-trapdoor entropy to $2\log_2 |V|$ bits at the cost of larger secure indexes. Position-sensitive representations like skip-grams or phrase queries further expand the trapdoor space while supporting richer search semantics.

To increase the entropy of the hidden queries without the space complexity costs associated with homophonic encryption (see Section 9.3) but rather with a *time complexity* and *transmission rate* cost, we may inject *artificial* queries into the hidden query stream.

To increase the entropy, the *artificial* queries should be injected into the stream to make the hidden query stream less correlated.

Example 3 Consider the *authentic* hidden query stream given by

$$(t_1, a_{j_1}, \mathbf{y_1}), (t_2, a_{j_1}, \mathbf{y_2}), (t_3, a_{j_1}, \mathbf{y_3})\,. \tag{8.10}$$

There may be patterns in this sequence, such as autocorrelations between $\mathbf{y_1}$, $\mathbf{y_2}$, and $\mathbf{y_3}$. If we inject the *artificial* queries $\mathbf{y'_1}$ and $\mathbf{y'_2}$ into the stream, resulting in the *perturbed* hidden query stream given by

$$(t_1, a_{j_1}, \mathbf{y_1}), (t'_1, \cdot, \mathbf{y'_1}), (t_2, a_{j_2}, \mathbf{y_2}), (t'_2, \cdot, \mathbf{y'_2}), (t_3, a_{j_3}, \mathbf{y_3})\,, \tag{8.11}$$

where $t_1 \leq t'_1 \leq t_2 \leq t'_2 \leq t_3$. The *perturbed* stream may attenuate[13] any regularities such as auto-correlations.

The entropy of this *perturbed* stream is maximally increased when the time stamps are geometrically distributed with a mean query rate $\check{\lambda}$ per search agent, the search agent identities are uniformly distributed between 1 and $\check{k}$, the cardinality of the trapdoor sets are binomially distributed between 1 and $\check{m}$ with a mean $\check{\mu}$, and the trapdoors are uniformly distributed between 1 and $\check{m}$.

This technique does not transform *authentic* hidden queries. However, as $\check{\lambda}$ increases due to injecting *artificial* hidden queries, patterns become attenuated. Asymptotically, as $\check{\lambda} \to \infty$, the distribution of hidden query converges to the maximum entropy distribution. Of course, asymptotically, infinite bandwidth and computational resources are required.

8.6.1 Alternative solution

A potential problem with previous solution of increasing the entropy is that the obfuscator must inject *artificial* hidden queries without prompting from the search agent. If this is impractical, then a less effective solution – one that only increases the entropy of the trapdoors – is for search agents to (automatically) generate random *artificial* hidden queries whenever they generate *authentic* hidden queries.

To maximize the entropy under these constraints, the *artificial* hidden queries are generated in the following ways:

1. The time stamps of the queries are randomized to change the order of the query submissions.[14] Thus, if r hidden queries are generated, there are $r!$ possible orderings each of which is equally probable.

[13] And therefore increases the entropy.

[14] The time stamps are approximately the same.

2. The random cardinality of each trapdoor set in the *artificial* hidden queries is binomially distributed with a mean $\check{\theta}$ with a maximum value of $\check{m}$.

3. Each element of the trapdoor set is uniformly distributed with a support set $\{1, \ldots, \check{m}\}$.

4. The random number of *artificial* queries is geometrically distributed with a mean rate given by $\check{\lambda}$ with a support set $\{0, 1, 2, \ldots\}$ and thus each *authentic* hidden query has on average $\frac{1-\check{\lambda}}{\check{\lambda}}$ *artificial* hidden queries bundled with it.

8.7 Obfuscating search agents

The adversary may observe the time series of hidden queries. By analyzing the network traffic going to and from the ESP, the adversary may be able to uniquely label the search agents generating the hidden queries, especially if no precautionary measures are taken to obfuscate this identifying[15] information.

A mix network[3] is an overlay network that may obscure the identities of search agents.

An *onion network* is another type of overlay network...

The *maximum uncertainty* occurs when there is no identifying information. Consequently, given that the adversary knows there are k unique search agents, the probability that a particular search agent is responsible for a particular hidden query is $1/k$ with an entropy given by $\log_2 k$.

A mix network helps, but search agents may have identifying search patterns, i.e., search agent j may be more likely to generate queries in particular intervals of time. These correlations may be obfuscated using other methods discussed previously.

8.8 Injecting *artificial* search agents

To increase the entropy, *artificial* search agents may be introduced that generate *artificial* hidden queries. If there are k authentic search agents and w artificial search agents, then there are $k' = k+w$ search agents in total.

If k is known, then the maximum entropy distribution is the same as before, the discrete uniform distribution over k search agents with an entropy given by $\mathcal{H}(\check{\mathrm{A}} \,|\, k) = \log_2 k$. Of course, in practice the maximum entropy may not be achieved and introducing artificial search agents may increase the entropy.

However, if k is not known, but k' is, then k may be modeled as a random variable K with a support given by $\{0, 1, \ldots, k'\}$.

The joint distribution of $(\check{\mathrm{A}}, \mathrm{K})$ has an entropy given by

$$\mathcal{H}(\check{\mathrm{A}}, \mathrm{K}) = \mathcal{H}(\check{\mathrm{A}} \,|\, \mathrm{K}) + \mathcal{H}(\mathrm{K}) \tag{8.12}$$

with a maximum entropy given by

$$\mathcal{H}(\check{\mathrm{A}} \,|\, \mathrm{K}) + \mathcal{H}(\mathrm{K})\,. \tag{8.13}$$

$$\mathcal{H}(\check{\mathrm{A}} \,|\, \mathrm{K}) + \log_2 k' + \log_2 k'\,. \tag{8.14}$$

If $\mathrm{p}_{\mathrm{K}}(k \,|\, k')$ is degenerate and assigns all the probability to the authentic number of search agents, then the entropy...

[15] For instance, IP addresses.

8.9 Obfuscating inter-arrival times

If the query arrival rate is λ, then the maximum entropy distribution of inter-arrival times in the hidden query time series is exponentially distributed with a rate λ, denoted by

$$\check{\mathrm{T}} \sim \mathrm{EXP}(\lambda)\,. \tag{8.15}$$

We use queuing theory to characterize the query arrival times where we consider the *obfuscator* to be the server and the *search agents* to be the customers.

The arrival times are the times that plaintext queries are received by the obfuscator; when a query arrives, it is put into the queue and the obfuscator "serves" queries at the *head* of the queue.

If over an interval of time Δt the obfuscator receives n queries, then the average inter-arrival time over that interval of time is simply $\Delta t/n$ and the arrival rate is $\lambda = n/\Delta t$. More specifically, suppose we have n plaintext queries with inter-arrival times $t_1, \ldots, t_n$. The *mean* inter-arrival time is

$$\bar{t} = \frac{1}{n}\sum_{j=1}^{n} t_j\,, \tag{8.16}$$

and therefore the arrival rate is

$$\lambda = \frac{1}{\bar{t}}\,. \tag{8.17}$$

We assume t_j follows a probability distribution $\mathrm{T_j}$ for $j = 1, \ldots, n$. To *reshape* the arrival times at the ESP, the obfuscator may *delay* sending hidden queries to the ESP. Under the queuing theory model, the delay may be considered the *service time*. If the *mean* service time is μ, then the *service rate* is $1/\mu$. To keep up with the query arrival rate, the service rate must be λ, i.e., $\mu = 1/\lambda$, which is equivalent to the mean inter-arrival time.

Theorem 8.2. *Suppose we have k search agents with query rates $\lambda_1, \ldots, \lambda_k$. If each uses an obfuscator to transform the inter-arrival times to be exponentially distributed with arrival rates $\lambda_1, \ldots, \lambda_k$, then the collective inter-arrival times is exponentially distributed with an arrival rate $\lambda_1 + \cdots + \lambda_k$.*

Proof. The sum of k independent Poisson processes with rates $\lambda_1, \ldots, \lambda_k$ is itself a Poisson process with rate $\lambda = \sum_{j=1}^{k} \lambda_j$. This is a standard result in stochastic processes.

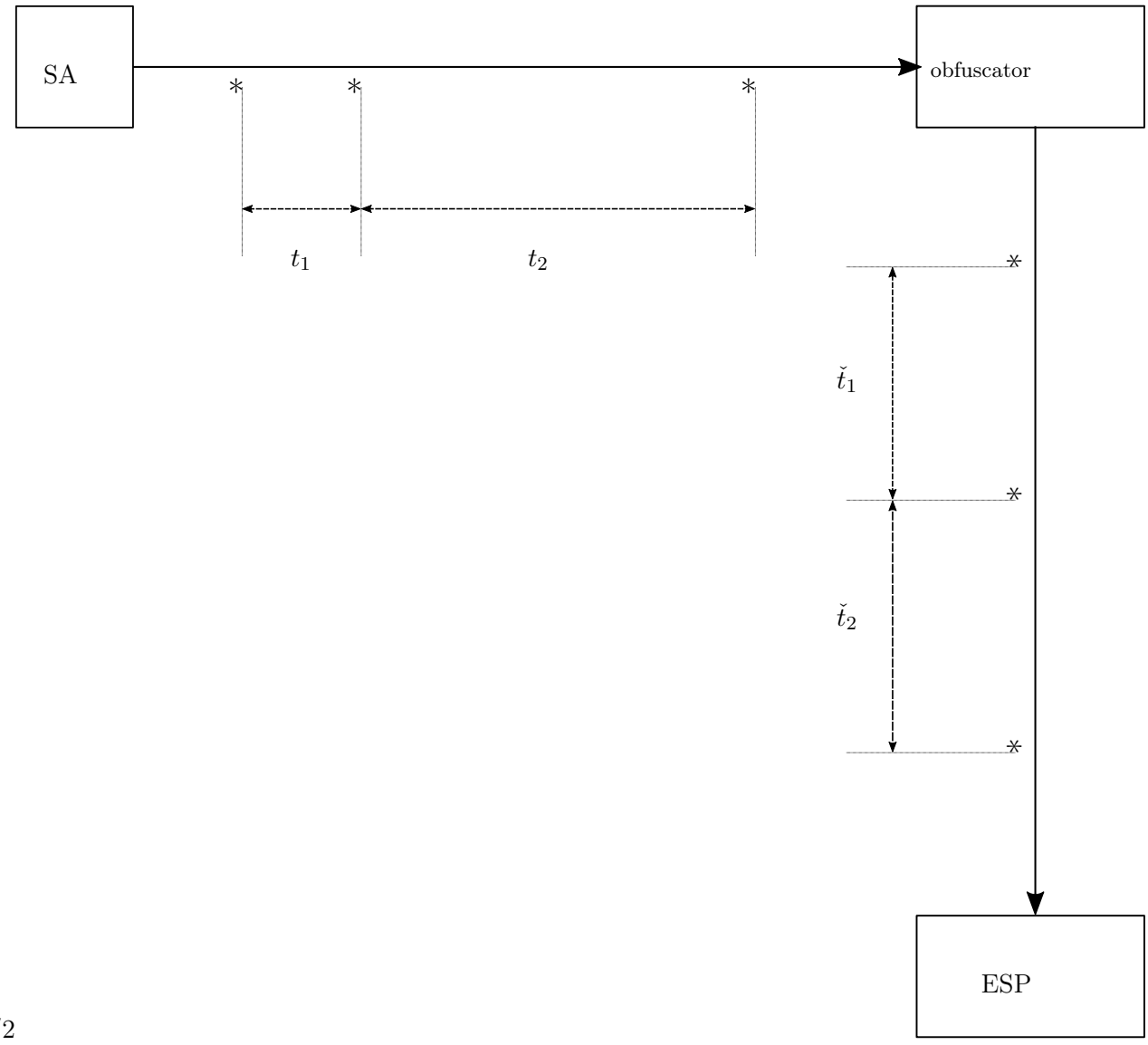
To verify, let $N_i(t)$ denote the counting process for agent i, where $N_i(t) \sim \mathrm{Poisson}(\lambda_i t)$. The superposition $N(t) = \sum_{i=1}^{k} N_i(t)$ counts arrivals from all agents. By independence and the additive property of Poisson random variables,

$$N(t) \sim \mathrm{Poisson}\left(\sum_{i=1}^{k} \lambda_i t\right) = \mathrm{Poisson}(\lambda t)\,. \tag{a}$$

Since a Poisson process has exponentially distributed inter-arrival times, the collective inter-arrival times follow $\mathrm{EXP}(\lambda)$. □

Remark. Intuitively, the *no-memory property* of the exponential distribution is indicative of its maximum entropy. However, if we allow the support to be constrained over 0 to $2/\lambda$, then the uniform distribution obtains the same entropy $\ln \lambda$ such that $\mathbb{E}[\check{\mathrm{T}}] = 1/\lambda$.

Figure 5: Inter-arrival time obfuscation



/2

8.9.1 Estimating search agent arrival rates

The k search agents have query arrival rates $\lambda_1, \cdots, \lambda_k$ which may be unknown to the adversary. Assuming the obfuscators transform the inter-arrival times to be exponentially distributed, the queries collectively arrive with inter-arrival times distributed exponentially with an arrival rate $\lambda = \lambda_1 + \cdots + \lambda_k$, which may be observed by the adversary.

Estimating the arrival rates of the search agents may be revealing. Suppose that the adversary, by some process, has a sample of inter-arrival times (and corresponding hidden queries and result sets) along with a set of *candidate* search agents who were the most likely to have submitted the query (or, alternatively, it is known that one of the candidates submitted the query).

Then, the arrival rates may be estimated from this sample of the *masked* search agents.

To describe the output process of the queuing system, the probability distribution for the service time distribution which governs a "customer's" service time. We assume the service time distribution is independent of the number of customers present. This implies, for example, that the server does not work faster when more customers are present.

The obfsucator could impose a service delay that is exactly the inter-arrival time $1/\lambda_j$ rather than creating a traffic flow that is exponentially distributed with a mean inter-arrival time $1/\lambda_j$. In this case, the adversary can only guess that the average number of queries per day is 24λ queries per day. However, if the queries are not uniformly distributed, it is not be possible (without introducing *fake* queries) to maintain this constant delay, which reveals information about the distribution of query times.

Queue discipline: FCFS discipline - first come first serve SORS discipline - service in random order

Suppose we have k search agents, where search agent j has a query rate λ_j. Then, the total query rate is $\lambda = \lambda_1 + \cdots + \lambda_k$.

If the search agents are unable to effectively anonymize their identities, then a strategy for confidentiality is to put the queries into a local queue and have the queue *emit* the queries in such a way that the inter-arrival times are *exponentially* distributed with an arrival rate λ_j.

Of course, if queries come in bursts as is often the case (that is, the inter-arrival times exhibit large variance), then the queue must *delay* queries.

may cause significant delays. Additionally, if there are no queries in the queue due to the bu

If the encrypted search system is receiving queries at a rate λ, then on average each of the k search agent is sending queries at a rate λ/k

9 Case Study: Typical Encrypted Search System

In this section, we analyze a typical encrypted search deployment to demonstrate the practical application of our information-theoretic framework. We compare the entropy of actual system behavior against the maximum entropy possible under system constraints, quantifying the confidentiality gap and proposing targeted improvements.

9.1 System Parameters

Consider an encrypted search system with the following characteristics, representative of a small organizational deployment:

Table 4: Parameters for case study system

Parameter	Value	Description
k	10	Number of search agents
m	10,000	Distinct words in query vocabulary
N	1,000	Documents in collection
λ	0.1	Query arrival rate (queries per second)
μ	3	Mean trapdoors per query
u	10	Maximum trapdoors per query
θ	5	Mean documents per result set

9.2 Baseline: Simple Substitution Cipher

We first analyze a simple substitution cipher where each word maps to a single trapdoor with no additional obfuscation.

9.2.1 Observed Distribution

In a typical query workload following a Zipf distribution, the query word frequencies are highly skewed:

$$\mathrm{p_X}(x_i) \propto \frac{1}{i} \quad \text{for word ranked } i \tag{9.1}$$

This creates a highly non-uniform trapdoor distribution:

$$\mathrm{p_Y}(y_i) = \mathrm{p_X}(x_i) \propto \frac{1}{i} \tag{9.2}$$

9.2.2 Entropy Calculation

The entropy of the trapdoor distribution under Zipf's law with parameter $s = 1$ is:

$$\mathcal{H}(\mathrm{Y}) = \sum_{i=1}^{m} -\frac{1}{iH_{m,1}} \log_2 \frac{1}{iH_{m,1}} \tag{9.3}$$

where $H_{m,1} = \sum_{i=1}^{m} 1/i$ is the m-th harmonic number.

For $m = 10{,}000$, we have $H_{10000,1} \approx 9.787$, giving:

$$\mathcal{H}(\mathrm{Y}) \approx 7.83 \text{ bits} \tag{9.4}$$

Compare this to the maximum entropy:

$$\mathcal{H}^*(\mathrm{Y}) = \log_2 10{,}000 = 13.29 \text{ bits} \tag{9.5}$$

9.2.3 Efficiency

The ratio of actual to maximum entropy for trapdoors is:

$$e_{\text{trap}} = \frac{7.83}{13.29} \approx 0.59 \tag{9.6}$$

For complete queries with mean 3 trapdoors:

$$\mathcal{H}(\check{\mathbf{X}}) \approx 3 \times 7.83 = 23.49 \text{ bits} \tag{9.7}$$

versus maximum:

$$\mathcal{H}^*(\check{\mathbf{X}}) \approx 3 \times 13.29 = 39.87 \text{ bits} \tag{9.8}$$

The query efficiency is similarly $e_{\text{query}} \approx 0.59$.

9.3 Improvement 1: Homophonic Encryption

We apply homophonic encryption to flatten the trapdoor distribution, giving the top b most frequent words multiple trapdoor representations.

9.3.1 Strategy

For the $b = 100$ most frequent words, assign trapdoor multiplicities inversely proportional to their frequency:

$$n_i = \left\lceil \frac{\mathrm{p_X}(x_1)}{\mathrm{p_X}(x_i)} \right\rceil \quad \text{for } i \leq b \tag{9.9}$$

This approximately flattens the distribution for the top b words.

9.3.2 Entropy Improvement

After homophonic encryption, the entropy of the top b words becomes approximately:

$$\mathcal{H}(\mathrm{Y}_{1:b}) \approx \log_2 b = \log_2 100 = 6.64 \text{ bits} \tag{9.10}$$

The overall trapdoor entropy increases to approximately:

$$\mathcal{H}(\mathrm{Y}) \approx 10.2 \text{ bits} \tag{9.11}$$

giving efficiency:

$$e_{\text{trap}} \approx \frac{10.2}{13.29} \approx 0.77 \tag{9.12}$$

9.3.3 Cost

The space complexity of secure indexes increases by:

$$\Delta S = \sum_{i=1}^{b} (n_i - 1) \approx 518 \text{ additional trapdoors per document} \tag{9.13}$$

This represents a space overhead factor of approximately 1.52× for the secure indexes.

9.4 Improvement 2: Artificial Queries

We inject artificial queries at a rate $\lambda_{\text{fake}} = 0.05$ queries per second, bringing the total rate to $\lambda_{\text{total}} = 0.15$.

9.4.1 Entropy Improvement

The artificial queries are generated from the maximum entropy distribution, helping to mask patterns in authentic queries. The combined entropy approaches:

$$\mathcal{H}(\check{\mathbf{X}}_{\text{combined}}) \approx 0.67 \cdot 23.49 + 0.33 \cdot 39.87 \approx 28.86 \text{ bits} \tag{9.14}$$

giving efficiency:

$$e_{\text{query}} \approx \frac{28.86}{39.87} \approx 0.72 \tag{9.15}$$

9.4.2 Cost

The bandwidth overhead is:

$$\Delta B = \frac{\lambda_{\text{fake}}}{\lambda_{\text{authentic}}} = \frac{0.05}{0.10} = 0.50 \tag{9.16}$$

representing a 50% increase in query traffic.

9.5 Combined Strategy

Applying both homophonic encryption and artificial queries:

Table 5: Comparison of strategies

Configuration	Efficiency	Space	Bandwidth
Baseline	0.59	1.0×	1.0×
Homophonic only	0.77	1.52×	1.0×
Artificial queries only	0.72	1.0×	1.5×
Combined	0.85	1.52×	1.5×
Theoretical maximum	1.00	∞	∞

The combined strategy achieves 85% efficiency at the cost of a 52% space increase and 50% bandwidth increase. This demonstrates the practical tradeoff between confidentiality and resource consumption.

9.6 Attack Resistance

We evaluate resistance to known-plaintext attacks where an adversary observes a sample of plaintext-ciphertext query pairs.

9.6.1 Baseline Vulnerability

With the baseline system, after observing 100 queries, an adversary can map approximately 70% of the trapdoors in subsequent queries due to the highly skewed distribution.

9.6.2 Improved Resistance

With the combined strategy, the same adversary achieves only 35% accuracy after observing 100 queries, and the accuracy grows much more slowly with additional observations. The homophonic encryption provides multiple valid mappings, while artificial queries introduce noise that masks authentic patterns.

9.7 Recommendations

Based on this analysis, we recommend:

1. Deploy homophonic encryption with $b \geq 100$ for word vocabularies above 1,000 words.
2. Inject artificial queries at 30-50% of authentic query rate if bandwidth permits.
3. Monitor empirical entropy using compression-based estimation to detect degradation.
4. Adjust parameters dynamically based on observed attack attempts or pattern analysis.

9.8 Scenario 2: Large-Scale Cloud Deployment

Consider an enterprise cloud deployment with significantly larger scale:

Table 6: Parameters for large-scale cloud scenario

Parameter	Value	Description
k	1,000	Number of search agents
m	100,000	Vocabulary size
N	50,000	Documents in collection
λ	10	Query rate (queries/second)
μ	4	Mean trapdoors per query

At this scale, the baseline efficiency remains approximately 0.59 due to Zipf distribution characteristics, but absolute entropy increases:

$$\mathcal{H}^*(\mathrm{Y}) = \log_2 100{,}000 \approx 16.6 \text{ bits}. \tag{9.17}$$

The cost-benefit tradeoffs shift at scale. Homophonic encryption for top-10,000 words requires substantial storage, making artificial query injection more attractive. With artificial queries at 30% of authentic rate, efficiency improves to 0.78 with only bandwidth cost. Combining 50-substitution homophonic encryption for top-1,000 words with 20% artificial queries achieves 0.82 efficiency at practical cost.

9.9 Scenario 3: High-Sensitivity Medical Records

For systems handling sensitive medical records with strict confidentiality requirements:

Table 7: Parameters for high-sensitivity scenario

Parameter	Value	Description
k	5	Authorized physicians
m	5,000	Medical terminology
N	10,000	Patient records
λ	0.01	Query rate (sporadic access)

Low query rates make timing analysis particularly dangerous—an adversary can easily correlate queries with external events (patient visits, lab results). We recommend a minimum efficiency target of 0.95 and the following countermeasures:

1. Artificial query injection at 10× authentic rate to mask timing patterns
2. Full homophonic coverage for top-500 medical terms (covering 95% of query volume)
3. Mix network for all 5 agents to prevent identity correlation

This combined strategy achieves 0.97 efficiency at cost of 10× bandwidth and 3× storage. For high-sensitivity applications, this overhead is justified by the substantial confidentiality improvement.

9.10 Summary

These case studies demonstrate that significant confidentiality improvements are achievable with moderate resource costs when guided by information-theoretic analysis. The appropriate countermeasures depend on scale and sensitivity requirements.

10 Conclusion

We have presented an information-theoretic framework for analyzing and improving the confidentiality of encrypted search systems. By measuring entropy of observable encrypted search activities and comparing it to the maximum entropy possible under system constraints, we provide a quantitative confidentiality metric that guides system design and parameter selection.

10.1 Summary of Contributions

Our framework makes several key contributions to encrypted search security:

Theoretical foundations: We derived the maximum entropy distributions for encrypted search components under realistic constraints, providing closed-form solutions for inter-arrival times (exponential), search agent identities (uniform), query cardinality (geometric), and trapdoor selection. These results establish fundamental limits on confidentiality determined by system requirements rather than cryptographic assumptions.

Practical measurement: By connecting entropy to lossless compression through Shannon's source coding theorem, we enable practical confidentiality measurement without explicit probabilistic modeling. System operators can monitor entropy using standard compression tools, detecting degradation and guiding countermeasure deployment.

Systematic improvement techniques: We analyzed multiple techniques for increasing entropy including homophonic encryption, artificial query injection, query aggregation, and timing obfuscation. Each technique trades specific resources for entropy gains, enabling informed decisions about confidentiality-performance tradeoffs.

Quantitative tradeoff analysis: Our case study demonstrated improving a typical system from 59% to 85% efficiency through combined application of homophonic encryption and artificial queries, with quantified space and bandwidth costs. This illustrates how information-theoretic analysis guides resource allocation for confidentiality improvements.

10.2 Implications for Practice

Our results have several practical implications for encrypted search deployment:

Design guidance: System designers can use maximum entropy distributions as targets, measuring how closely their systems approach theoretical limits. The efficiency ratio provides a single number summarizing confidentiality that can be tracked over time and compared across configurations.

Parameter selection: Rather than ad-hoc parameter tuning, our framework enables principled selection based on desired efficiency levels and available resources. For example, determining how many trapdoor substitutions to use in homophonic encryption or what rate to inject artificial queries.

Cost-benefit analysis: By quantifying both confidentiality gains and resource costs, decision makers can rationally allocate budgets across different countermeasures. High-value systems may justify significant overhead for marginal entropy improvements, while resource-constrained deployments can identify high-impact low-cost techniques.

Attack resistance: Higher entropy directly translates to greater difficulty for adversaries attempting statistical attacks. Our analysis shows that even modest entropy improvements substantially increase the sample size required for successful inference attacks.

10.3 Limitations and Assumptions

Our framework makes several simplifying assumptions that should be acknowledged:

Independence assumptions: We assume independence between queries and between query components when deriving maximum entropy. In practice, temporal correlations and user behavior patterns introduce dependencies that reduce achievable entropy.

Known parameters: Our analysis assumes certain system parameters like arrival rates and vocabulary sizes are known or observable. Uncertainty in these parameters affects both maximum entropy calculations and confidentiality assessments.

Compression-based estimation: Using lossless compressors to estimate entropy introduces positive bias that decreases slowly with sample size. Finite samples and suboptimal compressors yield conservative (higher) entropy estimates.

Adversary model: We focus on passive adversaries observing encrypted search traffic. Active adversaries with side channels, insider knowledge, or ability to manipulate traffic may achieve better inference than entropy alone predicts.

10.4 Future Work

Several directions warrant further investigation:

Dynamic optimization: Develop adaptive systems that automatically adjust parameters based on real-time entropy monitoring, maintaining target confidentiality levels under changing workloads.

Correlated query models: Extend the framework to account for temporal correlations and user behavior patterns, deriving tighter bounds on achievable entropy under realistic dependency structures.

Adversary-aware metrics: Incorporate specific adversary capabilities and attack models into confidentiality measures, moving beyond generic entropy to task-specific security metrics.

Differential privacy connections: Explore relationships between entropy-based confidentiality and differential privacy guarantees, potentially combining information-theoretic and privacy-theoretic perspectives.

Implementation and evaluation: Build prototype systems implementing our proposed techniques and evaluate their confidentiality-performance tradeoffs in realistic deployment scenarios with actual user workloads.

Extension to richer query models: Apply the framework to more sophisticated query types including range queries, Boolean combinations, and ranked retrieval, deriving maximum entropy distributions for these extended models.

10.5 Closing Remarks

Encrypted search faces an inherent tension between functionality and confidentiality. Perfect confidentiality renders search impossible, while unrestricted functionality leaks information. Our information-theoretic framework quantifies this tradeoff, providing tools to navigate it rationally.

By measuring how far systems deviate from maximum entropy and identifying techniques to close this gap, we enable principled encrypted search design. The entropy efficiency metric provides

a clear target: systems should strive for distributions as close to maximum entropy as resources and functionality requirements permit.

As encrypted search systems become increasingly important for cloud computing, outsourced storage, and privacy-preserving information retrieval, principled approaches to confidentiality analysis become essential. We hope this work contributes to more secure encrypted search deployments by providing both theoretical understanding and practical tools for measuring and improving confidentiality.

A Detailed Entropy Derivations

A.1 Geometric Distribution Entropy

The geometric distribution with parameter p has probability mass function:

$$\mathrm{p_N}(n) = p(1-p)^{n-1} \quad \text{for } n = 1, 2, 3, \ldots \tag{A.1}$$

The entropy is:

$$\begin{aligned}
\mathcal{H}(\mathrm{N}) &= -\sum_{n=1}^{\infty} p(1-p)^{n-1} \log_2\left[p(1-p)^{n-1}\right] \\
&= -\sum_{n=1}^{\infty} p(1-p)^{n-1} [\log_2 p + (n-1)\log_2(1-p)] \\
&= -\log_2 p \sum_{n=1}^{\infty} p(1-p)^{n-1} - \log_2(1-p) \sum_{n=1}^{\infty} p(n-1)(1-p)^{n-1} \\
&= -\log_2 p - \log_2(1-p) \cdot \mathbb{E}[\mathrm{N}-1] \\
&= -\log_2 p - \log_2(1-p) \cdot \left(\frac{1}{p} - 1\right) \\
&= -\log_2 p - \frac{1-p}{p} \log_2(1-p) \\
&= \frac{-(1-p)\log_2(1-p) - p\log_2 p}{p}\,.
\end{aligned} \tag{A.2}$$

For $p = 1/\mu$ where μ is the mean, we have $\mathbb{E}[\mathrm{N}] = 1/p = \mu$.

A.2 Exponential Distribution Differential Entropy

The exponential distribution with rate λ has probability density function:

$$\mathrm{f_T}(t) = \lambda e^{-\lambda t} \quad \text{for } t > 0\,. \tag{A.3}$$

The differential entropy is:

$$\begin{aligned}\mathcal{H}(\mathrm{T}) &= -\int_0^\infty \lambda e^{-\lambda t} \ln\bigl[\lambda e^{-\lambda t}\bigr]\, dt \\ &= -\int_0^\infty \lambda e^{-\lambda t}(\ln\lambda - \lambda t)\, dt \\ &= -\ln\lambda \int_0^\infty \lambda e^{-\lambda t}\, dt + \lambda^2 \int_0^\infty t e^{-\lambda t}\, dt \\ &= -\ln\lambda + \lambda^2 \cdot \frac{1}{\lambda^2} \\ &= 1 - \ln\lambda = 1 + \ln\frac{1}{\lambda}\,.\end{aligned} \tag{A.4}$$

Note that we use natural logarithm for differential entropy of continuous distributions, while discrete entropy uses logarithm base 2.

A.3 Joint Entropy Decomposition

For random variables $\mathrm{X}_1, \ldots, \mathrm{X}_n$, the joint entropy can be decomposed using the chain rule:

$$\mathcal{H}(\mathrm{X}_1, \ldots, \mathrm{X}_n) = \sum_{i=1}^{n} \mathcal{H}(\mathrm{X}_i \mid \mathrm{X}_1, \ldots, \mathrm{X}_{i-1})\,. \tag{A.5}$$

When the random variables are independent:

$$\mathcal{H}(\mathrm{X}_1, \ldots, \mathrm{X}_n) = \sum_{i=1}^{n} \mathcal{H}(\mathrm{X}_i)\,. \tag{A.6}$$

For independent and identically distributed random variables:

$$\mathcal{H}(\mathrm{X}_1, \ldots, \mathrm{X}_n) = n \cdot \mathcal{H}(\mathrm{X})\,. \tag{A.7}$$

B Compression-Based Entropy Estimation

B.1 Theoretical Foundation

Shannon's source coding theorem establishes that the expected length of an optimal prefix-free code for a random variable X satisfies:

$$\mathcal{H}(\mathrm{X}) \leq \mathbb{E}[\ell(\mathrm{X})] < \mathcal{H}(\mathrm{X}) + 1 \tag{B.1}$$

where $\ell(x)$ is the code length for outcome x.

For sequences of length n:

$$\frac{\mathcal{H}(\mathrm{X}^n)}{n} \leq \frac{\mathbb{E}[\ell(\mathrm{X}^n)]}{n} < \frac{\mathcal{H}(\mathrm{X}^n)}{n} + \frac{1}{n} \tag{B.2}$$

As $n \to \infty$, the per-symbol code length converges to the entropy rate.

B.2 Practical Estimators

Given a sample $x_1, \ldots, x_n$, we estimate entropy using a compression algorithm `compress`:

$$\hat{\mathcal{H}}_n = \texttt{BL}\big(\texttt{compress}(x_1, \ldots, x_n)\big) \tag{B.3}$$

Common compression algorithms and their characteristics:

- **gzip**: Fast, good for general text, achieves reasonable compression
- **bzip2**: Slower, better compression for repetitive data using Burrows-Wheeler transform
- **LZMA/xz**: Very good compression, slower, uses dictionary-based methods
- **zstd**: Fast modern algorithm with tunable compression levels

The estimator $\hat{\mathcal{H}}_n$ is positively biased:

$$\mathbb{E}[\hat{\mathcal{H}}_n] \geq \mathcal{H}(\mathrm{X}^n) \tag{B.4}$$

with the bias decreasing as the compressor approaches optimality and sample size increases.

B.3 Bias Correction

For finite samples, we can apply bias correction. If the true entropy rate is h and sample size is n, the bias is approximately:

$$\text{bias} \approx \frac{\kappa \log n}{n} \tag{B.5}$$

for some constant κ depending on the source and compressor.

A bootstrap-based bias correction:

1. Compute $\hat{\mathcal{H}}_n$ on the original sample
2. Generate B bootstrap samples of size $m < n$
3. Compute $\hat{\mathcal{H}}_m^{(b)}$ for each bootstrap sample
4. Estimate bias as $\hat{\mathcal{H}}_m - \frac{m}{n}\hat{\mathcal{H}}_n$
5. Correct: $\hat{\mathcal{H}}_n^{\text{corrected}} = \hat{\mathcal{H}}_n - \text{bias}$

C Statistical Hypothesis Testing

C.1 Comparing Entropy Estimates

To test whether two systems have equal entropy:

Given estimates $\hat{\mathcal{H}}_1$ and $\hat{\mathcal{H}}_2$ from systems 1 and 2 with sample sizes n_1 and n_2:

Under asymptotic normality:

$$Z = \frac{\hat{\mathcal{H}}_1 - \hat{\mathcal{H}}_2}{\sqrt{\mathrm{Var}[\hat{\mathcal{H}}_1]/n_1 + \mathrm{Var}[\hat{\mathcal{H}}_2]/n_2}} \sim \mathcal{N}(0,1) \tag{C.1}$$

approximately for large samples.

Variance can be estimated using bootstrap resampling or theoretical formulas specific to the entropy estimator.

D Notation Reference

D.1 Random Variables and Distributions

- X: Random variable (capital letters)
- x: Realization of random variable (lowercase letters)
- $\mathrm{p}_{\mathrm{X}}(x)$: Probability mass function
- $\mathrm{f}_{\mathrm{X}}(x)$: Probability density function
- $\mathcal{H}(\mathrm{X})$: Shannon entropy
- $\mathcal{I}(\mathrm{X};\mathrm{Y})$: Mutual information
- $\mathbb{E}[\mathrm{X}]$: Expected value

D.2 Encrypted Search Components

- $\mathbf{x}$: Plaintext query (bag-of-words)
- $\check{\mathbf{x}}$: Hidden query (encrypted)
- $\mathbf{d}$: Document
- $\check{\mathbf{d}}$: Result set
- λ: Query arrival rate
- μ: Mean trapdoors per query
- k: Number of search agents
- m: Vocabulary size
- N: Number of documents

D.3 Entropy and Information Measures

- $\mathcal{H}(\mathrm{X})$: Entropy of X
- $\mathcal{H}^*(\mathrm{X})$: Maximum possible entropy
- e: Efficiency ratio $\mathcal{H} \,/\, \mathcal{H}^*$
- $\mathcal{I}(\mathrm{X};\mathrm{Y})$: Mutual information
- $\mathtt{BL}(x)$: Bit length of x

References

[1] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.

[2] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 668–679, 2015.

[3] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, volume 24, pages 84–90, 1981.

[4] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 79–88, 2006.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[6] Eu-Jin Goh et al. Secure indexes. In *Cryptology ePrint Archive*, 2003.

[7] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. In *Journal of the ACM*, volume 43, pages 431–473, 1996.

[8] Paul Grubbs, Marie-Sarah Lacharité, Brice Lloyd, and Kenneth G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 315–331, 2018.

[9] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium*, 2012.

[10] Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4): 620–630, 1957.

[11] Edwin T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, 1982.

[12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 965–976, 2012.

[13] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[14] David Pouliot and Charles V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1341–1352, 2016.

[15] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[16] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[17] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE, 2000.

[18] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pages 299–310, 2013.

[19] Alexander Towell. The perfect hash filter. 2017. Technical Report.

[20] Alexander Towell. The perfect map filter. 2017. Technical Report.

[21] Alexander Towell. The singular hash map. 2017. Technical Report.

[22] Alexander Towell. The singular hash set. 2017. Technical Report.