University of Nevada, Reno

Efficient Private Analytics in a Cybersecurity Threat Intelligence Exchange

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

by

Raghav Kaul

Dr. Shamik Sengupta - Thesis Co-advisor
Dr. Shahariar Badsha - Thesis Co-advisor
August, 2020

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

RAGHAV KAUL

entitled

Efficient Private Analytics in a Cybersecurity Threat Intelligence Exchange

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Shamik Sengupta
Advisor

Shahriar Badsha
Co-advisor

Hanif Livani
Graduate School Representative

David W. Zeh, Ph.D., Dean
Graduate School

August 2020

Abstract

Cyber threat intelligence (CTI) sharing has become increasingly important as a technique to mitigate cyber threats and attacks. Data breaches often share common indicators and CTI sharing allows organizations to learn from shortcomings and improve the overall state of cyber defense. However, data sharing can be seriously hindered by perceived privacy and security risk: threat data often reveals sensitive information about an organization's IT assets, implying that organizations may need guarantees of anonymity to participate in an exchange. This research proposes a privacy-preserving system for fast and practical search over encrypted CTI data. Our design goals were to create an open and extensible system that safely utilized existing searchable encryption primitives. Our work is similar to, but improves upon, existing encrypted databases such as CryptDB. We show several novel security safeguards on encrypted CTI sharing, such as access-control by default , min-entropy measurement, and smart index selection and joining. We also show several performance enhancements that make our system efficient with multiple CTI contributors, particularly enabling data fusion across a variety of heterogeneous and unstructured data sources.

Dedication

Dedicated to Ragya Kaul.

Acknowledgments

I wish to thank my co-advisors, Dr. Shamik Sengupta and Dr. Shahriar Badsha, for their support throughout this process, inspiring and driving this research as well as many other germane research projects. Our collaboration was a great highlight of this research effort, and through clear and consistent communication, we've had a great working relationship. I also wish to thank Dr. Hanif Livani for his guidance and support in reviewing this thesis. Finally, I'd like to thank my family and friends for their support, in ways large and small.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cyber threat intelligence (CTI) sharing has become increasingly important as a technique to mitigate cyber threats and attacks. Though data breaches occur across a myriad of types of IT infrastructure, organizations find that many breaches share common indicators; CTI sharing allows organizations to learn from shortcomings and improve the overall state of cyber defense. A variety of threat information architectures have been introduced [2–4], allowing aggregation of potential attacks and vulnerability information, as well as coordination on security incident documentation and incident response.

Today, CTI gives organizations insight into attack tactics [5], which can improve their security information and event management (SIEM) systems and therefore reduce incidents. CTI data encompasses system logs, configuration changes, database and network events, logins, and intrusion detection system (IDS) events, among other data types. Despite the supposed utility of such exchanges, they are nowhere near universally adopted in organizational workflows [5].

1.1 The Privacy Problem with CTI Sharing

Data sharing can be seriously hindered by perceived *privacy* and *security* risk: threat data often reveals sensitive information about an organization's IT assets, implying that organizations may need guarantees of anonymity to participate in an exchange. This motivates us to approach CTI sharing from a privacy-preservation perspective: how can organizations effectively share CTI while minimizing disclosure risk? Of course, an organization may choose to redact its CTI data [6]. However, the more organizations redact and choose not to share, the fewer valuable signals other organizations gain from participation in such a system [7].

CTI data may inadvertently contain customer personally-identifiable information. For example, a web server access log may reveal unauthorized resource access attempts [8] as well as legitimate accesses; the network addresses of such legitimate accesses may be considered PII under privacy legislation [9]. As another example, a user may want to contribute email server logs as CTI to disclose possible phishing events and identifiers. However, these logs may contain customer e-mail or contact information.

CTI data may also disclose information about IT assets [6]. For example, a typical web server access log or shell access log contains both "source" and "destination" IP addresses. "Sources" are external, but "destinations" could be public, static IPs that are easy targets for sniffing and reconnaissance. Even if the "destination" is a network-translated subnet IP, an attacker with enough "destinations" can piece together an organization's network topology. This can be used to horizontally escalate once a network perimeter is breached, or estimate the quantity of IT assets and reveal business metrics like load, usage, and costs.

CTI data can also reveal running applications, e.g. which specific version of a library or daemon is in use. An attacker with access to this data could adjust their tactics with knowledge of what attacks have already been seen or what defenses and

mitigations are employed.

1.2 Incentives to Participate

Contribution of threat data requires, from an organization, dedicated analyst time for labeling, technical investments, and business justification. Still, participation in a threat exchange is a great value for defenders [10], aiding design of proactive defense such as firewalls and malware scanners , and reactive security investigations [5]. Industry investment in billion-dollar threat exchange platforms such as CrowdStrike and Cisco [11,12] is further proof of the value of threat data exchange; however, these platforms are closed and proprietary, and rely on participants to trust the vendor to ensure security. So, the industry has validated the need for a such a product.

In fact, given the acceptance in industry of threat data, privacy is one of the primary reasons for non-participation in open threat exchanges [5]. In other words, participants are well aware of the rewards, but are still concerned about the risks of exchanges. Further, when the data exchange is modeled game theoretically, it is shown that participants *can* be incentivized to share high-quality data with an equal division of labor [7]. So, this work does not focus on encouraging participation (which is already highly incentivized), but rather *risk minimization*: how to provide high-quality insights while keeping data secure.

1.3 Contribution

For this work, we developed a privacy-preserving system for fast and practical search over encrypted CTI data. Our design goals were to create an open and extensible system that safely utilized existing searchable encryption primitives. Our work is

```
{   "event-id": "5d659c2d34ecea6c769ec9",
    "org-id": "identity--f27df111-ca31-4700-99d4-2635b6c37851",
    "geoip": {
      "ip": "165.227.91.185",
      "latitude": 40.7214,
      "longitude": -74.0052,
      "location": {
        "lat": 40.7214,
        "lon": -74.0052
      },
      "timezone": "America/New_York",
      "postal_code": "10013",
      "country_name": "United States",
      "city_name": "New York",
    },
    "@timestamp": "2019-08-27T20:51:18.993Z",
    "tags": [ ... ],
    "host": "unr-honeypot-ec2-100-2-34-56.compute-1.amazonaws...",
    "duration": 120.02226901054382,
    "eventid": "cowrie.session.closed",
    "msg": "Connection lost after 120 seconds",
    "src_ip": "165.227.91.185"      }
```


Figure 1.1: **CTI Sample from a Linux SSH honeypot**, TAHOE format [1]. Suppose an organization uploads this indicator of an SSH brute force attack against a firewalled node. The log line contains a `source_ip` parameters of the remote source of the inbound SSH connection attempt, and the `host` name of the firewalled node emitting the log. Typically, other organizations are interested in whether an IP has been noted to be the `source_ip` of other attacks, and would search for that attribute type. But, an attacker may be interested in the `host` name to snoop on the contributor's IT assets, so the contributor should filter access to the `host` name field.

similar to work such as CryptDB, in that it creates a fully-functioning, end-to-end system available for general use, as well as open benchmarks and code.

The contributions of this work are the following.

- We show several novel security "safeguards" on encrypted CTI sharing, such as access-control by default (3.3), min-entropy measurement (3.4.2), and smart index selection and joining (3.4.1).

- We show how, by making new CTI-specific assumptions, searchable encryption schemes can be made faster, e.g. by varying the PRF/trapdoor schemes for deterministic encryption (3.5), by efficiently creating fuzzy query sets (4.2) , and by caching arithmetic circuits for homomorphic Hamming comparisons (4.5).

- We create and open-source new query traces based on collected data from real CTI-sharing use-cases. Using these traces, we benchmark our system with existing SQL-based systems.

Our system supports a larger subset of operators than existing CTI sharing systems, including exact matches, range queries, fuzzy subset queries, and substring searches. And, whereas other encrypted search schemes are generically useful for SQL DBs, our scheme is specifically adapted for unstructured CTI data (see 3.1.1 for why being "unstructured" is an important assumption). Combining multiple algorithms provides more query variety than the existing specially-adapted CTI sharing prototypes.

Chapter 2

Background

2.1 Privacy-preserving Search

Encrypted SQL databases, not specific to threat intelligence applications, are well-studied and boast some adoption for sensitive medical and other data [13] [14] [15]. These systems support matching (equality) and range queries, sums, and fuzzy string search queries using similar protocols and algorithms used in this work. Encrypted database systems allow clients and application servers to compute arithmetic and logical operations on data encrypted at rest, usually with the aid of a semi-trusted proxy. These systems work well while minimizing invasive changes to application code. In the threat model outlined by CryptDB [13], an adversary has complete access to the database server.

CryptDB's layered architecture, which supports computation of user-defined functions, is of particular interest for CTI analysis queries. A SQL interface is presented to the user, and queries are decomposed into layers of operations, and each logical SQL operation is associated with an encryption mechanism that satisfies the minimal

security properties required for the operation. CryptDB is designed with assumptions of a symmetric-key setting, i.e. that query users are fully trusted. This assumption is not valid in a CTI exchange: outsourced CTI data may be fused and queried, but CTI contributors need the ability to exercise controls and policies over shared data.

Existing privacy-preserving search systems also suit structured data, while CTI is mostly unstructured data. For example, in [13]'s onion encryption, each column was strongly associated with an algorithm picked by predicting the kind of queries that would be run on it. In [13], encryption protects against an adversary that has complete access to the database server. A SQL interface is decomposed into "layers" of operations and partial results may be stored. However, problems arise when DB users do not a-priori mark fields as sensitive, another constraint that would be hard to enforce in the constantly-shifting data schema of CTI contributors. CTI sharing systems do not enforce SQL-like relational schema, as explained in Section 3.1.1. And, not every organization chooses the same privacy protection for CTI data, so one data type may have different algorithms applied to it. Finally, enjoying the benefits of several years of advances in ORE and FHE, we update CryptDB's constructions and algorithms can be updated to ones with stronger security and performance.

The usage of partially homomorphic encryption in information retrieval as well as prediction and recommendation tasks has also been proposed [16] [17] [18] [19] [20]. These protocols showed semantic security, however, suffer from efficiency issues and not viable when the number of entities increases in the system.

Searchable encryption techniques, such as those proposed by [21] [22] [23], invert database indices and encrypt tags associated with database records. These implementations fall short in the context of CTI information formats due to the level of granularity required: in practice, CTI labels would reveal, if decrypted, as much information about the sharer, as would decrypted plaintext records. Further, the use

of symmetric keys complicates key distribution in the context of a cyber exchange, since even peers without the original CTI sharer's private key would be interested in analytics over encrypted data. A similar searchable encryption scheme by [24] is of greater relevance, due to the use of public keys to encrypt database records and metadata, making key sharing for analytics straightforward.

Deterministic encryption [25] is of particular interest for searchability. Bellare et. al demonstrate a RSA formulation with a deterministic (fixed) OAEP padding parameter. The security model established by RSA-OAEP is limited in that the ciphertext is partial information about the plaintext, implying no privacy is possible if the plaintext is known to come from a small space. Bellare et. al address this by only requiring privacy when the plaintext is drawn from a space of large min-entropy. For privacy, we find deterministic encryption to be most suitable algorithm to allow encrypted search.

2.2 Cybersecurity Exchange Privacy

A privacy-preservation schemes for CTI exchange utilizing Attribute-based Encryption was proposed in [26]. In this framework, a secure registrar establishes an access-control scheme by issuing keys to exchange participants based on their attributes (e.g. job title, organization name, office location). When an exchange participant shares CTI data, data is associated with a policy represented by an access tree [27]. During a proactive request for CTI information by an exchange participant, data is requested from the centralized server and returned *iff* that user's attributes satisfy the requested record's policy. One shortcoming of this approach is that CTI requesters process data on the client side, unless they delegate or escrow their keys with the centralized CTI storage service. Another shortcoming of this approach is that, over time, multiple

heterogeneous ciphertext policies may be applied to data in a single category, even by a single CTI sharer who chooses to revise their security policies or data sharing practices. These shortcomings may lead to severe performance bottlenecks on the as numerous records with varying policies must be decrypted and combined on-the-fly by the client. Alongside raw data, serializing expressive CP-ABE policies may also consume an inefficient amount of space in the CTI storage server.

Various protocols and specifications have been proposed to assist in CTI sharing, such as STIX, OpenIOC, and IODEF [1, 28, 29]. However, these interchange formats do not preserve privacy over sensitive incident information, risking unauthorized information disclosure for exchange participants. Experimentally, we use loosely-structured JSON documents in STIX 2.0 format, with MongoDB as a backend, which was shown as a suitable model for CTI data in [28]. NoSQL databases are particularly appropriate for CTI data due to the variety in content of CTI and indicators of compromise (IOC) being submitted by organizations. Additionally, privacy preservation and tracking techniques for CTI exchange users can be used to prevent unauthorized disclosure of CTI data, as shown in [30–32].

Chapter 3

Searching Exact Records

In this chapter, we define the model we use to securely share data. A basic search model, supporting exact query matches, is defined. We describe the trust relationships between participants in CYBEX-P as well as the data interchange formats used for CTI sharing. We also introduce the overarching encryption strategy.

3.1 Privacy Model

In our sharing model, we consider CYBEX-P Organizations as composed of two parties: Contributors (writers) and Users (readers). However, the Contributor-User dichotomy is not firm: an exchange participant may choose to perform either action at any given time. In fact, our model extends to a intra-organization use case, where a single organization can store and query outsourced, encrypted threat intelligence, e.g. between business divisions.

Nonetheless, as in Figure 3.1, the boundary between organization is one of *qualified trust*, whereas the boundary between a participant organization and the underlying App Server or Storage Server is one of *honest-but-curious mistrust*.

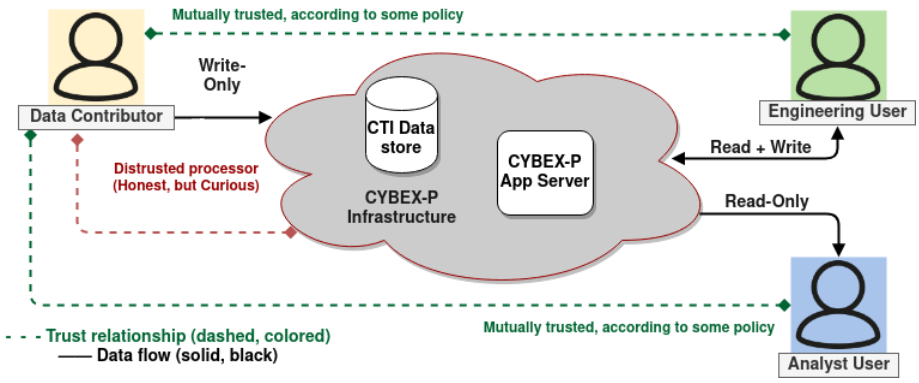

Figure 3.1: **CTI Sharing Model** 1) *Contributors* analyze, enrich, and report threat indicators from their infrastructure into a shared system. Privacy of these users' sensitive data is crucial, since threat indicators such as IP addresses can reveal IT assets. 2) *Engineering users* upload their own environment's threat signatures to CYBEX-P to find similar events and evaluate risk, simultaneously using and enhancing CYBEX-P's detection capabilities. 3) *Human analysts* derive "big-picture" insights from CYBEX, e.g. by searching for aggregations, clusters and statistics. A single organization may contain multiple types of users, switching roles frequently.

We model the CYBEX-P App Server as semi-honest, meaning that it only runs signed code and doesn't have access to encryption keys. The App Server may record metadata, but may not share this with CYBEX-P Organizations or collude in any other way. CYBEX-P Organizations are untrusted adversaries with bounded compute power. Organizations only have access to data if they accurately query it from CYBEX-P, but may perform inference and attack any data the server provides them.

In our model, a central server must process all read/write queries of CTI data. This application server exists in a segmented portion of the network, with full read/write access to a CTI database storing ground-truth documents. Writes to CYBEX-P are encrypted with the client's public key before being sent to CYBEX-P. CYBEX-P is assumed to be semi-honest, implying that organizational data is secure from CYBEX-P running analytical or inferential code. The CYBEX-P server inserts encrypted documents from organization CTI submissions into the database with minimal modification to DB middleware, and performs look ups on encrypted data from encrypted

queries submitted by clients. Documents are matched and returned to clients without decryption.

Organizations in a CYBEX-P framework, shown on the right edges of Figure 3.1 participate in submitting and ad-hoc querying CTI records. Organizations submitting data, shown towards the left, encrypt data with the appropriate DE algorithm available to all parties before sending the record to CYBEX-P. The secret keys are hidden from all organizations in the exchange. Organizations running exact match or count queries, shown towards the right, have some data to test or match against, which is deterministically encrypted using the same key and algorithm, before submitting the query to the CYBEX-P App Server. Clients receive encrypted results matching their query, encompassing documents possibly from multiple sources. Data retrievers may attempt to decrypt this data with the appropriate permissions.

3.1.1 Why TAHOE?

In our design and experiments, we use a variant of JSON called TAHOE [28]. TAHOE is well-suited for representing CTI data, more so than relational data. Recall that organizations decide to support CTI sharing based on the level of investment required. Relational formats are rigid and an "agreed-upon" format from a threat exchange may not fit an organization's entity model. For example, while some may consider IP addresses as part of a GeoIP block (as in Figure 1.1), others may separate it.Coercion into a standard relation would require organizational changes, as well as software to parse and format data. Ultimately, contribution is the bottleneck to CTI sharing, so ease of writing is key.

Also, organizations in a cybersecurity exchange have varying security policies. Some may provide very detailed web server access logs, whereas others choose to redact logs entirely. The JSON document model, and TAHOE in particular, sup-

ports optional and missing fields with ease. A CTI document format is not just a system to store and organize, but to aggregate and deliver insights for Engineering and Analyst users (Figure 3.1). So, with TAHOE's document model, CTI users can use slightly more complex search queries to extract useful matches and events even from incomplete data.

3.2 Deterministic Encryption (DE)

Now, in this section, we describe existing encryption primitives used to compose our system. For index creation, we use RSA-DOAEP with an Encrypt-and-Hash system as described in [25], with a key size of 2048 bits. A simple construction of RSA-DOAEP uses normal RSA encryption and keeps the padding for OAEP-mode fixed instead of randomly generated.

For some group of t organizations, let the CYBEX-P App Server generate an RSA public/private keypair PK_{DE}, SK_{DE} and distribute PK_{DE} to each organization P_i, where $i = 0 \dots t$. We notate the constituent algorithms of our Deterministic RSA encryption scheme Π_{DE} as key generation algorithm $\mathcal{K}_{DE}$, encryption algorithm $\mathcal{E}_{DE}$, and decryption algorithm $\mathcal{D}_{DE}$. The specific algorithms are defined in [25]. $\mathcal{E}_{DE}$ requires PK_{DE} and is computable by all organizations.

Indices cannot be decrypted because $\mathcal{D}_{DE}$ requires SK_{DE}, which is stored securely by the KMS, who does not disclose SK_{DE} to any organization or to the CYBEX-P App Server. The plaintext index can be recovered by an adversary with sufficient number of guesses; we show why our construction still satisfies the privacy requirements of a CTI-sharing environment in Section 4.5.

The Encrypt-and-Hash construction is particularly useful because hashes can be truncated to submitter-specified precision; when used for matching by the CYBEX-

P App Server, shorter hashes allow privacy when an organization submits with low entropy, at the cost of the querying organization receiving false positives for exact match queries, or noisy results for count queries.

Deterministic encryption [25] is of particular interest for searchability. Bellare et. al demonstrate a RSA formulation with a deterministic (fixed) OAEP padding parameter. The security model established by RSA-OAEP is limited in that the ciphertext is partial information about the plaintext, implying no privacy is possible if the plaintext is known to come from a small space. Bellare et. al address this by only requiring privacy when the plaintext is drawn from a space of large min-entropy.

Algorithm 1: Deterministic RSA-OAEP with Hash [25]

Param: PT: Plain-text byte sequence

$\text{PT}_{left} = x[1 \dots k_0]$
$\text{PT}_{right} = x[k_0 + 1 \dots n]$
$s_0 = \text{SHA-256}((N, e) \| \text{PT}_{right}) \oplus \text{PT}_{left}$
$t_0 = R((N, e) \| s_0$
$\text{Pre}_{left} = (s_1 \| t_0)[1 \dots n - k_1]$
$\text{Pre}_{right} = (s_1 \| t_0)[n - k_1 + 1 \dots n]$
$\text{CT} = \text{Pre}_{left} \| ((\text{Pre}_{right})^e \mod N)$
return CT

3.2.1 Min-Entropy as a Security Guarantee for DE

Deterministic encryption provides privacy for plaintexts that already have high min-entropy [25]. However, submitter-chosen indices aren't guaranteed to have sufficient entropy. To protect submitter privacy, the CYBEX-P client constructs indices that have sufficient min-entropy. A random variable X has min-entropy H_∞, where:

$$\max_x \Pr[X = x] = 2^{-k}$$

Following the definition in [25], X is said to have high min-entropy if the min-entropy grows faster than logarithmic with respect to the size of the sample space. If data type of a CTI document (e.g. a single MISP attribute) is modeled as a random variable with a uniform distribution over all possible values of that data type, then the min-entropy of index I with n independent fields $i_1, i_2, \ldots i_n$ is:

$$\begin{aligned} H_\infty(I) &= \sum_{y=0}^{n} \log_2 \frac{1}{\Pr[i_y]} \\ &= \sum_{y=0}^{n} \text{BitLengthOfDataType}(i_y) \end{aligned}$$

In practice, this calculation overestimates entropy since CTI data are not all random variables, nor do CTI data types all have fixed lengths. It is also possible that fields in an index contain dependencies, e.g. between ASNs and IP addresses, i.e. violating the i.i.d. assumption.

To measure entropy of a non-i.i.d. variable, an ensemble of entropy sampling methods based on best practices can be combined, including collision tests, counts, Markov approximation, compression tests, max frequency tests, and prediction tests [33]. The minimum of these tests can be used as an estimate of min-entropy of a given index, using existing documents with that index in the database. Fortunately, entropy tests can be easily computed on ciphertexts from deterministic encryption.

For CTI data of unbounded length, e.g. a URL, YARA rule, or text description, the inclusion of any such field into an index makes the index high-entropy. However, the inclusion of too many fields with unstructured and one-off descriptors of a cyber threat makes querying difficult; submitters are recommended against using such fields as indices.

For min-entropy to provide a strong security guarantee, we must show 1. that, theoretically, min-entropy bounds our threat model, and 2. that, experimentally, CTI data is actually high min-entropy.

It's shown in [25] that if the prior statements hold, DE is PRIV-CCA secure, which in our case, is equivalent to saying it's private to the CYBEX-P DB.

Now, we prove that min-entropy accurately bounds our threat model. The guessing adversary (GA) wants to determine at least one secret-encrypted value, where each value is chosen with some probability over the data domain. The GA knows the set of M possibilities, each possibility m with a probability distribution p_m. However, the GA wants to minimize the cost of guessing one specific value. The GA iterates with $\mathrm{k} \leq \mathrm{m}$ possible guesses, but may fail to guess if the guess is not one of the first k values chosen. The attacker can maximize their chances by guessing highest-probability values first. Then, as shown in [33], the expected number of guesses for the attack is:

$$E[G] = p_1 + 2p_2 + \cdots + (k-1)p_{k-1} + k(1 - \sum_{j=1}^{k-1} p_j)$$

The attack is successful if the secret value is found within the first k guesses. The likelihood of success is $P[C] = \sum_{j=1}^{k} p_j$. So, the number of guesses per success can be modeled as:

$$\frac{p_1 + 2p_2 + \cdots + (k-1)p_{k-1} + k(1 - P[C])}{P[C]} \tag{3.1}$$

The lower bound in work happens when the attacker's first guess is correct, i.e. when 3.1 is $\frac{1}{p_1}$. In general, the attacker must find the appropriate value of k, i.e. the number of guesses they should make to discover a value before moving on, maximizing the number of secret values successfully recovered. In [33], the value of k that minimizes 3.1 is close to H_{min} of M, the plaintext space. In [33], it's shown that the

difference between H_{min} of M and $\log_2$ 3.1 is bounded by:

$$0 \leq -\log_2 p_1 - \log_2 ExpectedWork \leq 1 - \log_2(p_1 + 1) \tag{3.2}$$

This approaches 1 as the size of M grows and p_1 approaches $\frac{1}{M}$. Taking the inverse of 3.1 and multiplying by an expected number of guesses k_t, in time interval t, gives us the number of successes per t:

$$E[C_t] = \frac{\sum_{j=1}^{k} p_j}{t(\sum_{i=1}^{k-1} ip_i + k(1 - \sum_{j=1}^{k-1} p_j))}$$

This measure, computable with simple distributional measures over M, can actually be used dynamically to tune rate limits and key rotations/re-cryptions. For example, the rotation interval must be picked so that $E[C]$ is less than the security level specified by organizations. A lower "security level" implies a stronger guarantee on DE min-entropy, and could involve a stricter rate limit, increasing the delay per guess k, or recrypting all values on a shorter rotation interval, reducing t.

In the context of CYBEX-P, the GA has a bounded number of queries it can make over a period of time, enforced as a rate-limit by the CYBEX-P App Server. We know that prediction error 3.2.1 is greatest when M is high and when p_1 is $\frac{1}{|M|}$. Given that, practical advice to achieving high min-entropy and therefore low guessability of DE-protected fields would encourage users to select attributes where the cardinality of the plaintext space is high and the value distribution is close to uniform.

3.3 Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

In CP-ABE, the Ciphertext policy tree specifies a set of Boolean conditions required for decryption. Each tree node x contains an integer threshold t_x of its child nodes (m_x in total) which must be satisfied for that node to be satisfied. Leaf nodes encode some attribute, corresponding to an organization's attributes. The tree is satisfied if the root node is satisfied. `&` ("AND") conditions can be encoded as a tree node x with threshold $t_x = m_x$. `|` ("OR") conditions can be encoded as $t_x = 1$.

A message in CP-ABE is encrypted alongside its policy tree. Policy tree size generally scales $n \log n$ relative to the number of terminal conditions. Encryption can be performed in time linear to the size of the access tree; [27] show how to compute the ciphertext in a traversal the access tree. In our experiment, we use the more efficient realization proposed in [34].

Organizations sharing sensitive data set CP-ABE access structures based on the data's privacy requirement and sensitivity level. In Figure 1.1, the submitter, the University of Nevada, Reno, could encrypt `Org` and `org_id` with the access structure `(org.id = 1 OR org.name = "NOCDC")`, so that other clients from the University could retrieve the threat data, or so that law enforcement could retrieve the data.

Let the CYBEX-P App Server generate a public/private keypair PK_{ABE}, MK_{ABE}, and distribute PK_{ABE} to each organization P_i. The techniques to generate CP-ABE public and master keys from some bilinear group are described in [27]. The security of the CP-ABE scheme is based on the hardness of the Decisional Bilinear Diffie-Hellman Exponent problem.

For each organization P_i, the Key Management Server will listen for and receive A_i, a set of organizational attributes. The KMS generates and returns some organi-

zational security key SK_i based on the input parameters A_i and MK_{ABE}.

Second, `attribute[2].dest_ip`, which reveals the IP of a node in the University DMZ, would have access structure `org.sector = 61 AND org.system = "NSHE" AND user.role = "Network Engineer" and user.level >= 4 OR org.name = "NOCDC"`, allowing specific Engineers, e.g. those responsible for network maintenance, in other Academic institutions in the NSHE system to view IPs targeted by an attacker, especially if those nodes are bridged between institutional intranets.

Finally, `attribute[0]` and `attribute[1]` would have a simpler access structure, like `org.id > 0`, ensuring that any organization actively participating in CYBEX-P with a valid, recent CP-ABE client key, can access the threat data. When ciphertext search returns several matching CTI documents, those documents' fields are still encrypted with an attribute-based access control policy. However, the App Server does not have to decrypt the matched record's privacy-protected fields: those encrypted fields can be returned to the requesting organization, who then decrypts fields where their organization public key policy satisfies the ciphertext attribute-based policy. For any attributes where decryption fails, the requesting organization receives no information about the ciphertexts, other than the fact that the policies don't match, which preserves the privacy of other organizations in CYBEX-P.

In this section, we show results from our implementation of the privacy-preserving protocol on storing/querying CTI data based on historical traces.

3.4 Practical Implementation Considerations

3.4.1 Index Selection

Previous applications of DE are unsuitable for CTI data because there can only be one valid index for a given document. However, DB indexing always evolves with usage. In CYBEX-P, users from one organization (readers) do not communicate with contributors from another organization (writers) as to what indexes are most desirable. So, contributors must have an opportunistic strategy to determine which indexes to use. The naive solution is to have multiple encryptions of the same document in one or many collections.

Instead, *Composite Indices* allow submitters to define multiple searchable fields [35]. Composite indices allow a contributor to share encrypted data even if they're unsure of how other users intend to use it. Each DE-protected field can be placed into an array. A query matching any field in the array returns the whole document. Composite indices are an efficient solution. Encrypting one document with multiple DE policies multiplies DB storage space by the number of potentially indexable fields, growing larger as CTI adds complex and detailed field types.

This approach is more ergonomic for contributors and users. It would be harder to write compound queries that match multiple fields, separately encrypted and indexed. Data inconsistencies may arise across collections, and de-duplication would be required since queries could match the same event multiple times. Any normally DE-protected field can be used in a composite index. Composite indices are also data type-agnostic. And, they can be used in a variety of NoSQL backends (e.g. MongoDB, CouchDB, HBase).

Algorithm 2: EncryptIndex(R, I)

Perform RSA-DOAEP on a CTI Document's Index Fields.
Param: R: CTI Record
Param: I: Composite Index Fields
$R_I = \{\}$
for F $\in$ I **do**
    R_I[F] = R[F]
end
D = {}
D[I.CompositeName] = $\mathcal{E}_{DE}(PK_{DE}, R_I)$
return Q

Algorithm 3: EncryptAndSend(R, C)

Process and send a CTI record to the CYBEX-P App Server.
Param: R: CTI Record
Param: C: Mapping between CTI Field $\rightarrow$ CP-ABE Policy
Initialize: B as an empty list
for I $\in$ GetIndices() **do**
    D = **EncryptIndex**(PK_{DE}, I, R)
    for F: Field $\in R \setminus I$ **do**
        P = C[F] or DEFAULT_ACL
        D[F] = $\mathcal{E}_{ABE}(PK_{ABE}, P, R[F])$
    end
    B += D
end
NoSQLWriteAll(B)

3.4.2 Fast Min-Entropy Measurement

In Figures 3.2 and 3.3, we show the results of min-entropy measurements on CTI data, finding that most unstructured CTI data fields are high-min entropy due to the cardinality of the sample space, which only improves when fields are concatenated and recombined as per Algorithm 2.

All MISP fields with H_{min} above the line in Figure 3.2 should be considered to have a min-entropy higher than that required for safe deterministic encryption. For a MISP stream, the entropy of threat data may change over time, implying that the set of values suitable for deterministic encryption may change over time. We claim that our entropy sample in Figure 3.2 shows sufficient entropy in a representative MISP stream for usage in CYBEX-P.

In general, CTI data follows a distribution far from random. Frequently, there are clusters of observed threat information which make the plaintexts used for CTI search non-random. The implication is that an attacker that has a good understanding of the distribution of threat intelligence and who can reverse the mapping between plaintext CTI fields and encrypted index fields can break their encryption.

We measure data size empirically by counting the unique values of a CTI field over time. After an initial monitoring period, we approximate the “data size” required to represent that CTI field to the number of bits needed to represent all unique values in the monitoring period. In practice, this assumption is

Additionally, we show a second experiment demonstrating the benefits of our "concatenation" strategy for lengthening plaintext indices. The goal of the strategy is to introduce additional variance, and therefore, entropy, into the distribution of plaintext indices, making breaking their encryption substantially more difficult. In 3.3, we see that index fields created from a concatenation of several MISP attributes have a higher min-entropy even with the same underlying data size.

Additionally, we note that both the vanilla encryptions and the entropy-boosted encryptions of MISP data grow faster than logarithmically.

In general, we expect entropy of a CTI field to increase monotonically with respect to the size of the field. The larger the space of possible threat indicators that can be represented in a CTI document, the less likely it is for a malicious CYBEX-P organization or for the CYBEX-P server to break the encryption scheme.

The official implementation of the NIST SP-800-90B tests handle, at maximum, 8-bit symbols (up to 256 unique values). We adapt this toolkit to handle arbitrary-precision symbols, excepting the Compression, Markov, and Collision tests, which only operate on 1-bit symbols. When a test requires a symbol bit-width different from that provided, we transform the input symbol sequence into a bit stream using `itoa` with radix 2, drawing a new symbol sequence from the stream at the specified bit-width. If the required bit-width is larger than the input bit-width, this transformation increases the amount of unique symbols.

Another limitation of the NIST tests is that they only handle integers. To measure min-entropy of arbitrary categorical symbols, defined as strings, floating-point numbers, or other objects supporting equality comparisons, we implement an encoder mechanism which maps distinct categorical values to integers up to a configurable maximum bit-width. For example, the sequence [“A”, “B”, “C”, “A”] is mapped to [00, 01, 10, 00]. More frequent values are assigned lower encodings, ensuring the top K most frequent values (with least entropy) are always encoded. Due to the bit-width constraint, the encoding is lossy: e.g. an encoding length of 16 can only capture 65536 unique symbols. Theoretically, mapping un-mapped values to an unassigned code inside the range would lower the min-entropy estimate, especially under a more uniform distribution. In our experiments, we find no significant difference in entropy statistics between assigning and dropping un-mapped values.

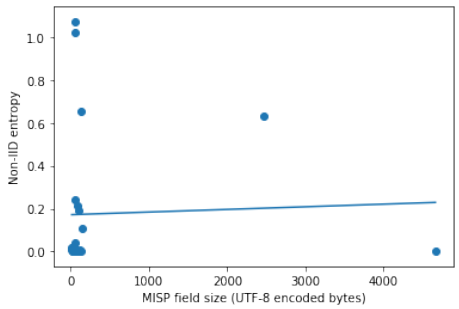

Figure 3.2: H_{min} for MISP attribute types by attribute length

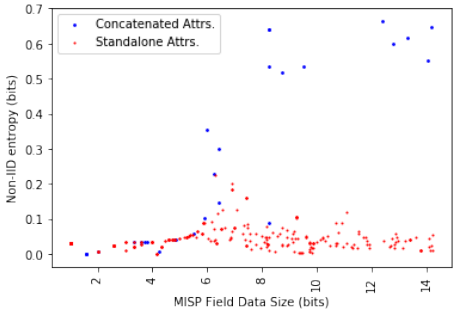

Figure 3.3: H_{min} for MISP attributes with Algorithm 2

3.4.3 Key Management

Prior to sending/searching over data, the Key Management Service (KMS) must perform a single RSA key generation round to generate the shared public/private keys and distribute public keys to all available parties. Organizations joining after the initial phase may request public keys from the KMS to send data/perform searches. Additionally, each organization must submit its organizational attributes S_i (as relevant to CYBEX-P) to the KMS, which has the ability to verify and sign those attributes, creating and returning a unique CP-ABE decryption key, SK_i, for that organization. This can be used by the organization to decrypt attached fields to a given query results.

Our strong security assumption is that the KMS is a trusted authority which cannot disclose secret keys. Even though our system allow CTI search on ciphertexts, there may be legitimate scenarios in unique cases requiring the use of secret key. For example, in the case of unauthorized key disclosure or key rotation, it may be prudent to re-encrypt existing CTI collections (particularly their index fields) with a new public key by first using the compromised secret key for decryption.

3.5 Performance Evaluation

We test on Pronghorn, UNR's HPC cluster, on a node with 16-core Intel Xeon E5-2683 @2.10GHz. We use a MongoDB backend for generic CTI document storage, LevelDB for embedded storage in the fuzzy search proxy. We implement the CYBEX-P Privacy Client in CPython, binding to C/C++ libraries for cryptography routines, including Charm [36] and SEAL [37] [38].

We create and publish novel query traces for CTI data based on actual query observation over a period of two months in 2020 [39]. Traces are reconstructed based on the DB writes executed by a front-end graph visualization tool, CYBEX-FE, created for the CYBEX-P project. For example, CYBEX-FE may issue the following operations: 1. Receive a user-submitted signal, such as observed IP address, ASN, URL, or other network identifier of an attack 2. Query for events containing the submitted signal 3. Gather additional signals from related events 4. Expand query to include additional, related signals 5. Keep querying and expanding the related event set, until the results no longer satisfy a user-defined relevance threshold.

In Figures 3.5 and 3.4, we show order statistics (minimum, median, maximum) for the distribution of end-to-end latencies (in milliseconds) for CTI search. We measure 1000 randomized queries against CTI data with and without indices, and against data stores of increasing size. End-to-end latency for search includes encryption of query documents, search, retrieval, and decryption of CTI documents. Figure 3.5 shows the total query time including CP-ABE encryption, which provides access control over CTI data. In general, Figures 3.5 and 3.4 show that query time increases quickly for un-indexed CTI storage; for indices created with deterministic encryption, query time scales slowly for data sizes up to 256 MB. We also compare this result to the same DE function running with another PRF. This demonstrates the efficiency of querying a deterministically encrypted index, even with the overhead of RSA-DOAEP.

DE Algorithm	Key Size (bits)	Encrypt Time
AES-CMC	256	3.0 ms
AES-SIV	256	3.1 ms
AES-CBC	256	3.1 ms
RSA-DOAEP	2048	14.9 ms
RSA-DOAEP	4096	20.2 ms

Table 3.1: Performance comparison for DE algorithms

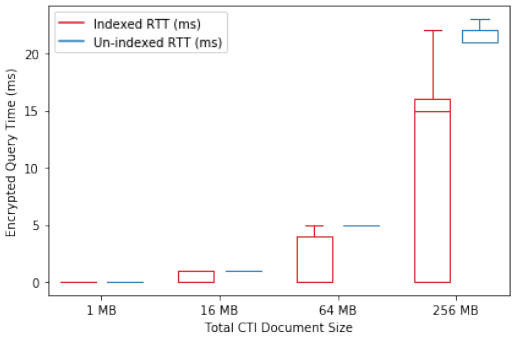

Figure 3.4: Deterministic Encryption performance for NoSQL search

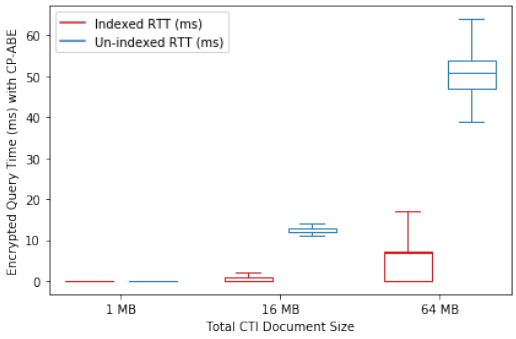

Figure 3.5: Deterministic Encryption + CP-ABE performance for NoSQL search

Compared to AES-based trapdoor permutations, RSA-DOAEP is slower by 2-3 orders of magnitude. In many cases, an AES key can be shared between parties during a single round of computation, i.e. once data has been written using that hybrid key. However, this comes at a cost of an out-of-band key-exchange algorithm, such as Diffie-Hellman KEX, to establish the AES secret, whereas the RSA-DOAEP key can be broadcast publically by the KMS. This choice comes down to the implementer and their particular threat model; and RSA-based trapdoor permutation provides a stronger guarantee when the CYBEX-P App Server can't be trusted.

End-to-end query performance roughly breaks down into the following major components:

- *CP-ABE*: This takes the bulk of the time; for both indexed and unindexed collections, CP-ABE encryption takes time proportional to the length of the policy. We experiment with the systems proposed in [27] and [34], eventually using the latter for performance.

- *RSA-DOAEP*: This takes 10-20 ms. As a performance optimization, for sufficiently large plaintexts, we associate the plaintexts with a symmetric AES key and use a hybrid encryption technique.

- *Encrypted Search*: Search is the fastest process of the pipeline: in general, encryption takes the majority of the round trip. This is likely due to the CP-ABE not being optimized for performance, while NoSQL is well-optimized for search on binary fields [35] [36].

Chapter 4

Advanced Queries on CTI

In the previous chapter, we covered how we can use Deterministic Encryption to match exact threat intelligence indicators that a cyber-defense observer wants to know more about. Once the data is matched and retrieved from an encrypted database system, it's necessary to perform additional filtering steps to learn even more about cyber threats. In this chapter, we'll cover a way to combine multiple encryption types to support advanced and fuzzy queries after the matching stage for encrypted threat intelligence queries.

4.1 Supporting Multiple Encryption Types

Data contributors may ensure their privacy by 1. redaction or pseudonymization of sensitive data, 2. adjusting data noise and granularity, either in the same channel or a larger radius, or 3. encryption algorithms, and their respective parameter selection. Since redaction reduces the utility derived from CTI, we approach privacy-preserving sharing by computing trusted or semi-trusted queries over encrypted data.

We divide encryption primitives into two classes. The first class, including de-

terministic encryption and attribute-based encryption, allows a contributor to share data with untrusted collaborators over an untrusted server. Queries by collaborators are semi-trusted according to some access-control policy. The second class, including order-revealing encryption and searchable homomorphic encryption, allows a contributor to share data to a trusted collaborator over an untrusted server. Ultimately, privacy algorithms must be selected commensurately with a contributors' trust model.

Data Type	Algorithm	Security Guarantee
All attributes	Deterministic Encryption 3.2	PRIV-CCA [25]
Sensitive or privileged attributes	Ciphertext-policy Attribute Based Encryption 3.3	IND-CCA [27] [34]
IPs, URLs, kill chain, executable path	Wildcard DE 4.2	PRIV-CCA [40]
Timestamps	Order-preserving Encryption 4.3	IND-OCPA [41]
File contents, YARA rules, Analyst notes	Somewhat-homomorphic Encryption 4.4	IND-CPA [42] [43]

Table 4.1: Encryption techniques by CTI attribute type and corresponding security guarantees

In this chapter, we show that fuzzy search techniques, falling into both the first and second categories, can be used in a private way in a CTI search system. In general, **fuzzy search** refers to retrieval of approximate or coarse-grained matches to a user query. There are two avenues for fuzzy search:

- The user submits a course-grained query and receives results that match a "radius" around that query, for example, approximate-matching, ranked matching, and semantic matching with inner products [44].
- The user submits a fine-grained query, and receives results that deviate from that query by some predefined distance metric, e.g. Euclidean distance or Hamming distance. [40] [45] [46].

4.2 Wildcard Encryption

Often, CTI users would like to know more about a threat, but don't know exactly where to start. This is to say, they may know an approximate or partial threat

indicator, but want to know event relatedness and significance. In a motivating example, a user may want to investigate a particular IP address range (particularly an IPv6 range), instead of a single address. Or, a user may have a URL contacted by a malware, but parts of the URL query string contain unnecessary data.

One approach is to encrypt many versions of the plaintext, such that any query "close to" a plaintext would return the record. In [40], the fuzzy search set is an edit-distance approximation based on wildcard characters. But, in this approach, the wildcard set for data size l and edit distance d grows at $O(l^d)$. However, for well-structured CTI fields like URLs and IPs, we can exploit patterns to improve fuzzy set creation.

First, define a truncation strategy. For IP addresses, octets are a natural segments. So, we can create a fuzzy set of the IP address plaintext with 4 host masks: /32 (exact address), /24, /16, and /8 (broadest search). For URLs, a similar strategy could successively remove (without replacement) the query string, path, and subdomain from the URL, leaving only the hostname for the broadest search. Each truncated subsequence may be DE-protected before being written ; search queries would use the same disjoint set. However, readers can request more "exact" results, e.g. only /32 and /24 net masks. Finally, a read query may perform a disjunctive search on this set. A "fuzzy match" to any of the wildcard set values would return the whole CTI record.

4.3 Order-Preserving Encryption (OPE)

Our system supports efficient range operations by allowing contributors to encrypt fields while revealing their order. We use order-revealing encryption to support encryption of timestamps, often used to narrow the scope or time window of a security

incident. However, any asymmetric order-revealing encryption algorithm is trivially vulnerable to a keyword-guessing attack [41]. Therefore, the security model changes for ORE: collusion between CYBEX-P and any contributor holding encryption keys would lead to total disclosure of all range-encrypted values.

One basic security guarantee for OPE is indistinguishability under an ordered chosen plaintext attack, whereby a malicious organization repeatedly encrypts chosen plaintexts and submits them to the CTI server, e.g. in a binary search pattern, observing which ciphertexts are returned to determine their exact values [45]. IND-OCPA is similar to semantic security [47]. Because no efficient OPE scheme can be IND-OCPA secure, Boneh et. al [48] introduce "order-revealing encryption," loosening the criteria that ciphertexts must be numerically comparable. Instead, ORE creates ciphertexts where there is some fast comparison function that can order them. We use a PRF-based scheme [41] which satisfies IND-OCPA security. It leaks no information except the amount by which the plaintexts differ and their relative ordering.

Algorithm 4: Order-preserving Encryption [41]

Using a pseudo-random function PRF, return an encryption of an integer allowing efficient comparison between ciphertexts
Param: $\mathrm{PT} \in \mathbb{Z}$, a plaintext integer message to encrypt
Param: λ, a security parameter for key length
Initialize: Ciphertext modulus $\mathrm{M} = 3$
Initialize: Random λ-bit secret key SK_{ORE} to be used by the PRF
Let: PT_i be the i-th bit of the plaintext
Let: n be the bit length of PT
for $i \in [n]$ **do**
    $\mathrm{u}_i = \mathrm{PRF}(\mathrm{SK}_{ORE}, (\mathrm{PT}_1 \| \ \mathrm{PT}_2 \ldots \| \ \mathrm{PT}_{i-1} \| 0^{n-i})) + \mathrm{PT}_i \mod \mathrm{M}$
    $\mathrm{CT} = \mathrm{CT} + \mathrm{u}_i \cdot \mathrm{M}^{n-i}$
end
return $\mathrm{CT} \in [0, \mathrm{M}^n - 1]$

4.3.1 Order-preserving Encryption's Security Guarantee

For ORE, we use a PRF-based scheme [41] which satisfies IND-OCPA security. It leaks no information except the amount by which the plaintexts differ and their relative ordering. In CYBEX-P, for example, IPs and ASNs can be encoded as sequential and ORE-protected, allowing users to check if, e.g., malicious network activity was seen from a CIDR range rather than an exact network location. "Difference" measures are useful for encoding purposes, but practically, knowing that IPs or ASN differ by some number of bits has no established meaning to an attacker. Also ORE-protected, timestamps for CTI reports are bounded (e.g. within a Unix epoch), so leaking relative differences is still acceptable. An attacker may be able to discern a "cadence" (e.g. n-hourly batch reports) for CTI contributions, not exact times-of-day or timezones.

4.4 Somewhat-Homomorphic Encryption (SHE)

Malware executables are common and useful CTI data. However, to date, CTI reporters have used file hashes to share malware samples. File hashes are private with respect to file contents, but unfortunately, only allow exact matches. However, malware rarely looks the same at a binary level due to a variety of obfuscation and randomization techniques to defeat detection mechanisms. However, sharing entire file contents may reveal victim-specific malware customization, compromising sharer privacy.

So, traditional CTI search must trade off privacy or usability. However, this problem may be addressed by homomorphic encryption that supports pattern matching. There are several variants of homomorphic encryption schemes, including ideal lattice-based schemes, NTRU encryption-based schemes, integer based schemes, and learning-with-errors schemes over rings. Fully-homomorphic encryption (FHE) fully

supports string pattern matching. However, these circuits aren't practical for thousands of millions of records because FHE ciphertexts grow too quickly. Somewhat-homomorphic encryption (SHE) relaxes the FHE constraint: under SHE, limited-depth circuits can be computed on ciphertexts, enabling simple arithmetic.

Distinct malwares often share common kernel gadgets or exploits. Even if entire executables don't match, sections, segments, or substrings may match. To address this, we adapt the encoding method presented in [42] to perform a substring check on homomorphically-encrypted file contents. We describe our modification to Yasuda's algorithm below.

Algorithm 5: Homomorphic substring search [42] [43]

Param: PK_{HE}, SK_{HE}: Keypair for BFV SHE scheme
Param: ptn: Pattern to search for, as a byte sequence
Let: ptn_j be the j-th bit of ptn under a fixed-width coding scheme
$\mathrm{PT}_{ptn} = \sum_{j=0}^{l-1} \mathrm{ptn}_j x^{n-j}$
$\mathrm{CT}_{ptn} = \mathrm{BFV.Encrypt}(\mathrm{PK}_{HE}, \mathrm{PT}_{ptn})$
for $CT_{txt} \in D_{HE}$ **do**
    $\mathrm{CT}_{HamDis,ptn,txt} = \mathrm{CT}_{txt} \times C_l + \mathrm{CT}_{ptn} \times C'_k + (-2\mathrm{CT}_{txt}) \times \mathrm{CT}_{ptn}$
end
Let: $\mathrm{CT}_{Ham,x,y} = \sum_{i=0}^{n-1} m_i x^i$ be the comparison of the x-th text with the y-th pattern
Let: m_i be the i-th coefficient of the decryption result of $CT_{Ham,x,y}$
$\mathrm{d}_{HamDis,x,y} = \min(\{\mathrm{m}_i | \ \mathrm{CT}_{Ham,x,y} \leftarrow \mathrm{BFV.Decrypt}(\mathrm{SK}_{HE}, \mathrm{CT}_{HamDis,x,y})\})$
return $\{ \mathrm{CT}_{HamDis,ptn,txt} | \ \mathrm{d}_{HamDis,ptn,txt} = l - k \}$

In fact, any homomorphic encryption encoding method that allows a bit-wise inner product can be adapted for a substring search. To see why this is the case, take a text `CreateProcessAsUser` and a smaller pattern `CreateProcess`, both common Win32 syscalls. If we overlay pattern with the first character of text and pad with null bytes, we can clearly see that the Hamming distance (d_H) is proportional to the size difference between text and pattern (the exact weight is based on the character encoding). And, this is a substring strict check: It's not possible for d_H to be less

than the difference because padding doesn't match anything, and if d_H were more than the difference, that would mean at least one character in pattern was missing from text. To our knowledge, this is the first use of Hamming distances on SHE ciphertexts to perform substring checks.

4.4.1 Lattice Security Guarantee

Yasuda, et. al define the following packing method, where combinations of the packed ciphertexts give us packed ciphertexts that enable efficient computations. Given two messages, named text and pattern respectively, let their elementwise multiplication be as follows:

$$\begin{aligned} m_{txt}(T) \cdot m_{ptn}(P) &= (\sum_{i=0}^{k-1} t_i x^i) \cdot (-\sum_{j=0}^{l-1} p_j x^{n-j}) \\ &= -\sum_{j=0}^{l-1} \sum_{h=0}^{l-1} t_h +_j p_j x^{n+h} (h = i - j) \\ &= \sum_{h=0}^{k-l} \sum_{j=0}^{l-1} t_h +_j p_j x^h \end{aligned}$$

Under this encoding scheme, our ciphertexts are polynomials modulo t, such that the following messages each correspond with the given polynomials.

$$\begin{aligned} CT_{txt}(T, pk) &= BFV.Encrypt(m_{txt}(T), pk) \\ m_{txt}(T) &= \sum_{i=0}^{k-1} t_i x^i \\ CT_{ptn}(P, pk) &= BFV.Encrypt(m_{ptn}(P), pk) \end{aligned}$$

$$m_{ptn}(P) = \sum_{j=0}^{l-1} p_j x^{n-j}$$

The latter encoding of pattern p can be thought of as a bit reversal and right shift by the $n-k$ bits. However, the underlying polynomial coefficients remain $modulo x^n + 1$. The homomorphic multiplication of the above, packed ciphertexts corresponds to an elementwise multiplication. Each element is a single bit of the ciphertext, so the output produced is the inner product between text and pattern. Therefore, the security of the Hamming scheme is based on the security of the underlying Ring-LWE scheme.

4.5 Performance Evaluation

For substring search, we use the BFV encryption scheme [43] from SEAL [38]. We use Analyst field notes from a stream of 360 MISP events observed in 2019-2020. Each event may have multiple notes fields (designated as "text"); we collect 696 unique text fields. We randomly select a substring (note that, for pattern matching, the length of the pattern must always be less than the length of the text). Text fields are often small notes, e.g. indicating the name of the import tool, the attack campaign an event was associated with, or malware/file annotations.

The experimental baseline is to leave parameters to the defaults suggested in [42]: $(n, t, q, \sigma) = (4096, 65bit, 4096, 8)$. We use a batch encoder to encode the first 512 characters of a text field into a polynomial. In practice, over 650 text fields fit into the encoded polynomial, indicating a good choice of polynomial modulus $n = 4096$. Doubling the polynomial modulus would double the maximum supported text size to 1Kb. Because underlying MISP events are shuffled, each search runs the circuit on a different number of ciphertexts before finding a match.

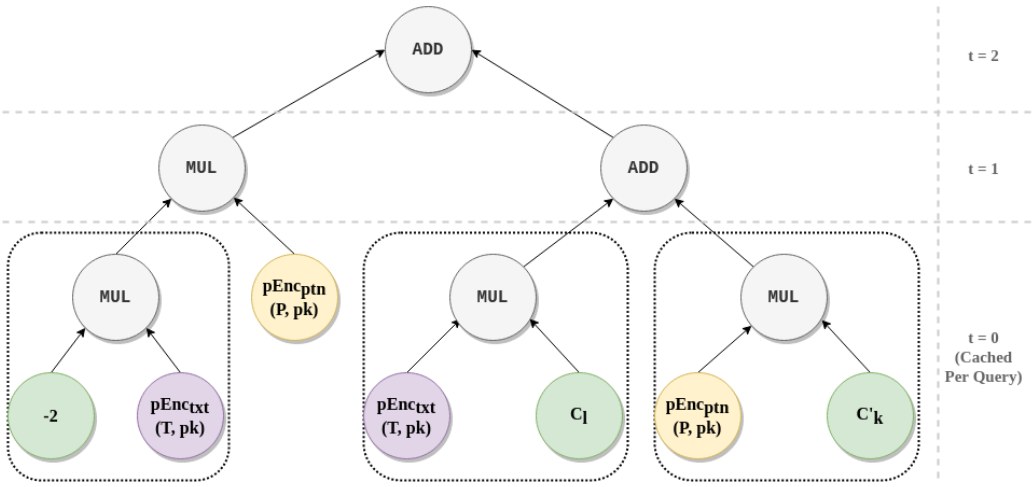

Figure 4.1: Cached circuit layout. Green elements can be computed independently by the client, yellow elements can be computed independently by the storage server, and purple elements are public parameters of the system.

The 4.3 shows the effect of increasing CTI field size on lookup performance. Searching plaintext fields averaging up to 800Kb can still be searched with sub-second latency, though outlying CTI fields such as free-form text are still the slowest. The last figure shows that the penalty on database storage sizes is moderate, increasing for larger plaintexts. In all 3 figures, caching shows a performance boost over full circuit comparison, with the difference being most pronounced in ciphertext sizes.

Arithmetic Circuit Caching

We try two comparison circuit variants.

- For the first variant, the comparator takes in two byte strings, one representing the stored text and one representing the query pattern. Each step of the comparison circuit is computed from the start, returning 0 if the text matches and -1 otherwise.

- For the second variant, the comparator takes two serialized circuits, each with partial computations. The stored, serialized circuit contains the encrypted values of $-2pEnc_{txt}(T, pk)$ and $pEnc_{txt}(T, pk) \times C_l$; all entrypoints to write to the

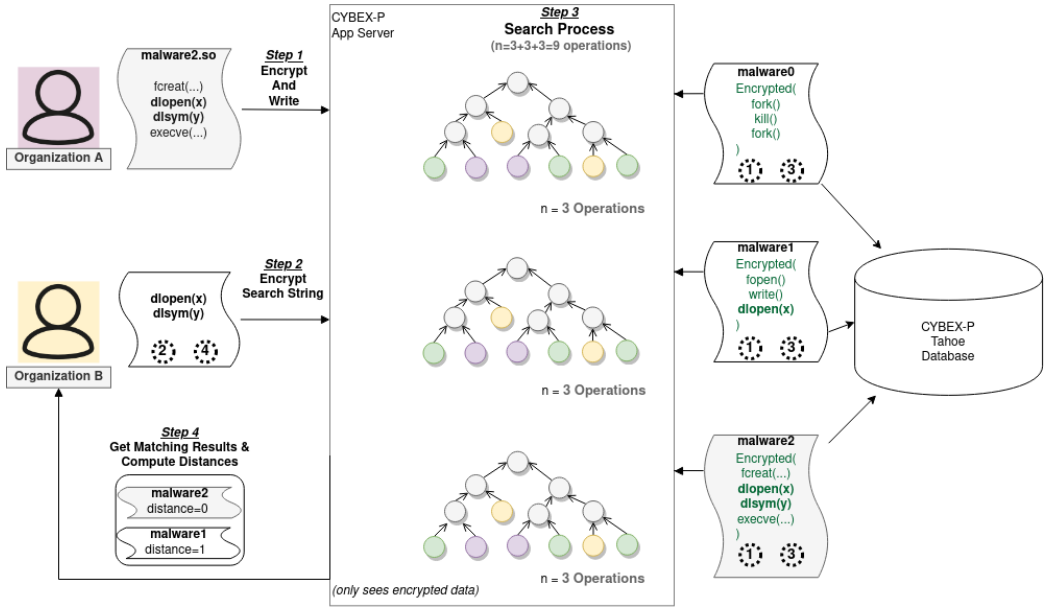

Figure 4.2: Database search with cached circuits. In Step 1, a CTI contributor runs homomorphic encryption on, e.g., a malware sighting, submitting it to the CYBEX-P App Server. The app server combines this with C_l, caching the sub-circuit. In Step 2, an CYBEX-P user encrypts a search term, e.g. sequence of syscalls from malware, submitting it t

database must encrypt and encode T before writing. The query circuit contains the encrypted values of $pEnc_{ptn}(P, pk)$ and $pEnc_{ptn}(P, ptn) \times C'_k$.

In 4.3, we examine the performance impact of arithmetic circuit caching on substring comparison queries over CTI data. For each experiment, search query time is a key lookup performed with a randomly selected key from the input set.

We find that both cached and uncached comparisons are slower than a plaintext search by at least 3 orders of magnitude, with plaintext taking, on average, 10μ s. On average, cached comparison circuits execute in 30ms, while uncached comparison circuits execute in 50ms per ciphertext. We anticipate that the worst case behavior is due to the particular arrangement of ciphertexts before finding a match.

One constraint of this protocol is that each comparison result must be interactively decrypted, and only a single coefficient, i.e. the smallest coefficient, from the whole plaintext polynomial is required for the substring check. Given that this comparison

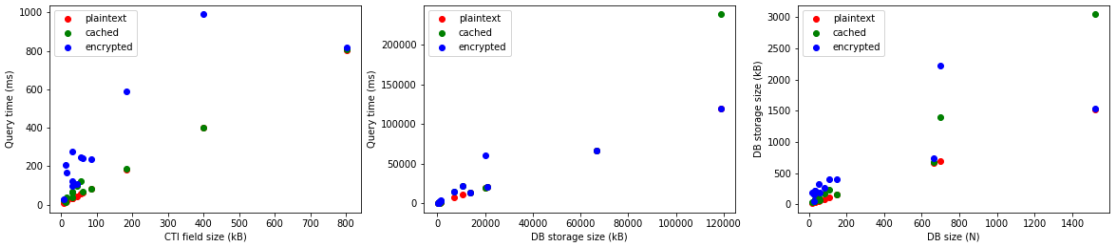

Figure 4.3: Performance of Plaintext Compare vs. Naive and Cached Circuits

is undoubtedly slower than both OPE and DE, it should be applied only on records that already satisfy other search predicates. Further, any optimization to extract a single polynomial modulus from the BFV plaintext would lead to performance gains.

It may be possible to use a CKKS homomorphic encryption encoding scheme to achieve better performance with some loss of accuracy. Each integer in an encoded text fields represents a single substring chunk, with inner product computation between two vectors functioning as a Hamming metric. However, since CKKS is an approximate encoding scheme, clients may encounter false positives.

Chapter 5

Conclusion

We proposed an end-to-end privacy-preserving mechanism for CTI sharing. We show how several encryption techniques can be combined to protect data while still enabling organizations to fully utilize aggregated threat intelligence. Our system supports equality comparisons, range queries, wildcard/disjunctive queries, and substring checks, as well as access control. We outline our protocol and review its baseline defenses against misuse, such as default encryption, automatic index selection, and min-entropy. Overall, several useful techniques for searchable encryption in the context of CTI data have been described, but these techniques are ultimately limited by the trust placed in a central key management system. Future work in threat intelligence should define a way of distributing key management among all CYBEX-P organizations to reduce implicit trust between parties. This includes investigating key rotation, especially a way to automate rotation over encrypted data. Additionally, it should be possible to distribute the generation of both symmetric and asymmetric keys. Finally, for symmetric keys used in the advanced query section, future work should investigate if those keys may be shared by proxy, e.g. by a one-time key exchange. Our goal in this paper was to show that searchable encryption techniques

can be practical for unstructured CTI. However, there are several remaining questions relating to mutual and organizational trust, especially relating to key generation and management. We also create a benchmark for CTI data read/write queries based on real observations of end-to-end CTI systems. We evaluate our system's performance on these systems and show how, with several CTI-specific assumptions, our system performs better than existing systems. We hope this provides a basis for future performance profiling and optimization to make CTI data search as close to real-time, or close to plaintext search, as possible.

Bibliography

[1] F. Sadique, K. Bakhshaliyev, J. Springer, and S. Sengupta, "A system architecture of cybersecurity information exchange with privacy (CYBEX-P)," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0493–0498.

[2] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," *Mitre Corporation*, vol. 11, pp. 1–22, 2012.

[3] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. ACM, 2016, pp. 49–56.

[4] E. W. Burger, M. D. Goodman, P. Kampanakis, and K. A. Zhu, "Taxonomy model for cyber threat intelligence information exchange technologies," in *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*. ACM, 2014, pp. 51–60.

[5] D. Shackleford, "Who's using cyberthreat intelligence and how?" *SANS Institute*, 2015.

[6] G. Fisk, C. Ardi, N. Pickett, J. Heidemann, M. Fisk, and C. Papadopoulos, "Privacy principles for sharing cyber security data," in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 193–197.

[7] D. Tosh, S. Sengupta, C. A. Kamhoua, and K. A. Kwiat, "Establishing evolutionary game models for cyber security information exchange CYBEX," *Journal of Computer and System Sciences*, vol. 98, pp. 27–52, 2018.

[8] D. Wichers, "OWASP Top-10 2013," *OWASP Foundation, February*, 2013.

[9] C. Sullivan and E. Burger, ""in the public interest": The privacy implications of international business-to-business sharing of cyber-threat intelligence," *Computer law & security review*, vol. 33, no. 1, pp. 14–29, 2017.

[10] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Computers & Security*, vol. 87, p. 101589, 2019.

[11] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives," 2017.

[12] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Computers & security*, vol. 72, pp. 212–233, 2018.

[13] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 85–100.

[14] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in *Proceedings of the VLDB Endowment*, vol. 6, no. 5. VLDB Endowment, 2013, pp. 289–300.

[15] M. Egorov and M. Wilkison, "Zerodb white paper," *arXiv preprint arXiv:1602.07168*, 2016.

[16] S. Badsha, X. Yi, and I. Khalil, "A practical privacy-preserving recommender system," *Data Science and Engineering*, vol. 1, no. 3, pp. 161–177, 2016.

[17] S. Badsha, X. Yi, I. Khalil, and E. Bertino, "Privacy preserving user-based recommender system," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1074–1083.

[18] A. Kelarev, X. Yi, S. Badsha, X. Yang, L. Rylands, and J. Seberry, "A multistage protocol for aggregated queries in distributed cloud databases with privacy protection," *Future Generation Computer Systems*, vol. 90, pp. 368–380, 2019.

[19] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, E. Bertino, and K. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[20] S. Dara, S. T. Zargar, and V. Muralidhara, "Towards privacy preserving threat intelligence," *Journal of information security and applications*, vol. 38, pp. 28–39, 2018.

[21] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 965–976.

[22] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *The VLDB Journal*, vol. 21, no. 3, pp. 333–358, Jun 2012. [Online]. Available: https://doi.org/10.1007/s00778-011-0245-7

[23] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007, pp. 350–364.

[24] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 506–522.

[25] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Annual International Cryptology Conference*. Springer, 2007, pp. 535–552.

[26] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Attribute based sharing in cybersecurity information exchange framework," in *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. IEEE, 2017, pp. 1–6.

[27] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[28] F. Sadique, S. Cheung, I. Vakilinia, S. Badsha, and S. Sengupta, "Automated structured threat information expression (stix) document generation with privacy preservation," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)(IEEE UEMCON 2018)*, 2018.

[29] F. Sadique, R. Kaul, S. Badsha, and S. Sengupta, "An automated framework for real-time phishing url detection," in *2020 IEEE 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020.

[30] S. Badsha, I. Vakilinia, and S. Sengupta, "Privacy preserving cyber threat information sharing and learning for cyber defense," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0708–0714.

[31] ——, "Blocynfo-share: Blockchain based cybersecurity information sharing with fine grained access control," in *2020 IEEE 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020.

[32] A. Thakkar, S. Badsha, and S. Sengupta, "Game theoretic approach applied in cybersecurity information exchange framework," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020.

[33] E. Barker and J. Kelsey, "Recommendation for the entropy sources used for random bit generation (nist sp 800-90b)," *NIST special publication*, 2012.

[34] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*. Springer, 2011, pp. 53–70.

[35] C. Kvalheim, "The Little MongoDB Schema Design Book," *The Blue Print Series*, 2015.

[36] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013. [Online]. Available: http://dx.doi.org/10.1007/s13389-013-0057-3

[37] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 3–18.

[38] "Microsoft SEAL (release 3.5)," Apr. 2020, microsoft Research, Redmond, WA. [Online]. Available: https://github.com/Microsoft/SEAL

[39] R. Kaul, "Cybex-p," https://github.com/raghavkaul/unr, 2020.

[40] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–5.

[41] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," in *International conference on fast software encryption*. Springer, 2016, pp. 474–493.

[42] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Secure pattern matching using somewhat homomorphic encryption," in *Proceedings of the 2013 ACM workshop on Cloud computing security workshop*, 2013, pp. 65–76.

[43] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.

[44] L. Chen, X. Sun, Z. Xia, and Q. Liu, "An efficient and privacy-preserving semantic multi-keyword ranked search over encrypted cloud data," *International Journal of Security and Its Applications*, vol. 8, no. 2, pp. 323–332, 2014.

[45] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 224–241.

[46] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 139–152.

[47] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[48] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 563–594.