Structured Encryption and Controlled Disclosure

Melissa Chase
Microsoft Research
melissac@microsoft.com

Seny Kamara
Microsoft Research
senyk@microsoft.com

Abstract

We consider the problem of encrypting structured data (e.g., a web graph or a social network) in such a way that it can be efficiently and privately queried. For this purpose, we introduce the notion of structured encryption which generalizes previous work on symmetric searchable encryption (SSE) to the setting of arbitrarily-structured data. In the context of cloud storage, structured encryption allows a client to encrypt data without losing the ability to query and retrieve it efficiently. Another application, which we introduce in this work, is to the problem of *controlled disclosure*, where a data owner wishes to grant access to only part of a massive dataset.

We propose a model for structured encryption, a formal security definition and several efficient constructions. We present schemes for performing queries on two simple types of structured data, specifically lookup queries on matrix-structured data, and search queries on labeled data. We then show how these can be used to construct efficient schemes for encrypting graph data while allowing for efficient neighbor and adjacency queries.

Finally, we consider data that exhibits a more complex structure such as labeled graph data (e.g., web graphs). We show how to encrypt this type of data in order to perform focused subgraph queries, which are used in several web search algorithms. Our construction is based on our labeled data and basic graph encryption schemes and provides insight into how several simpler algorithms can be combined to generate an efficient scheme for more complex queries.

1 Introduction

The most common use of encryption is to provide confidentiality by hiding all useful information about the plaintext. Encryption, however, often renders data useless in the sense that one loses the ability to operate on it. In certain settings this is undesirable and one would prefer encryption schemes that allow for some form of computation over encrypted data.

One example is in the context of remote data storage, or so-called "cloud storage", where a data owner wishes to store *structured* data (e.g., a collection of web pages) on an untrusted server and only retain a constant amount of information locally. To guarantee confidentiality, the owner could encrypt the data before sending it to the server but this approach is unsatisfactory because the data loses its structure and, in turn, the owner loses the ability to query it efficiently.

To address this problem we introduce the notion of *structured encryption*. A structured encryption scheme encrypts structured data in such a way that it can be queried through the use of a query-specific token that can only be generated with knowledge of the secret key. In addition, the query process reveals no useful information about either the query or the data. An important consideration in this context is the efficiency of the query operation on the server side. In fact, in the context of cloud storage, where one often works with massive datasets, even linear time operations can be infeasible.

Roughly speaking, we view structured data as a combination of a data structure δ and a sequence of data items $\mathbf{m} = (m_1, \ldots, m_n)$ such that δ encodes the data's structure and $\mathbf{m}$ represents the actual data. For example, in the case of graph-structured data such as a social network, δ is a graph with n

nodes and the ith element of $\mathbf{m}$ is the data associated with node i. To query the data efficiently, one queries δ to recover a set of pointers $I \subseteq [1, n]$ and then retrieves the items in $\mathbf{m}$ indexed by I.

At a high level, a structured encryption scheme takes as input structured data $(\delta, \mathbf{m})$ and outputs an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c} = (c_1, \ldots, c_n)$. Using the private key, a token τ can be constructed for any query such that pointers to the encryptions of $(m_i)_{i \in I}$ can be recovered from γ and τ. Furthermore, given the private key, one can decrypt any ciphertext c_i.

A certain class of symmetric searchable encryption (SSE) schemes [18, 12, 15] can be viewed as structured encryption schemes for the special purpose of private keyword search over encrypted document collections. Of course, the functionality provided by structured encryption can be achieved using general techniques like oblivious RAMs [20], secure two-party computation [37] and fully-homomorphic encryption (FHE) [17]. In our context, however, we are interested in solutions that are non-interactive and, at worst, linear in the number of data items as opposed to linear in the length of the data. All the schemes described in this work are non-interactive and optimal in that the query time is linear in the number of data items to be returned.

Informally, a basic notion of security for structured encryption guarantees that (1) an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$ reveal no partial information about the data $(\delta, \mathbf{m})$; and that (2) given, in addition, a sequence of tokens $(\tau_1, \ldots, \tau_t)$ for queries $\mathbf{q} = (q_1, \ldots, q_t)$ no information is revealed about either $\mathbf{m}$ or $\mathbf{q}$ beyond what can be inferred from some limited leakage which is a function of δ, $\mathbf{m}$ and $\mathbf{q}$. A stronger notion, introduced in [15], guarantees that (2) holds even when the queries are generated *adaptively*.

All known constructions that can be considered efficient structured encryption schemes (i.e., the index-based SSE schemes [18, 12, 15]) reveal some limited information about the data items and queries. In particular, for any query they reveal at least (1) the *access pattern*, which consists of the pointers I; and (2) the *query pattern*, which reveals whether two tokens were for the same query [1].

1.1 Applications of Structured Encryption

Private queries on encrypted data. The most immediate application of structured encryption is for performing private queries on encrypted data. In this setting, a client encrypts its (structured) data $(\delta, \mathbf{m})$ resulting in an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$. It then sends $(\gamma, \mathbf{c})$ to the server. Whenever the client wishes to query the data, it sends a token τ to the server which the latter uses to recover pointers J to the appropriate ciphertexts. Using a structured encryption scheme in this manner enables the client to store its data remotely while simultaneously guaranteeing confidentiality against the server (in the sense outlined above) and efficient querying and retrieval. While this problem has received considerable attention for the special case of document collections [35, 18, 5, 36, 12, 1, 15, 3, 33, 7], as far as we know, it has never been considered for other kinds of data.

Controlled disclosure for local algorithms. While the original motivation for structured encryption was to perform private queries on encrypted data (or more precisely, private *searches* on encrypted data), we introduce here a new application which we refer to as *controlled disclosure*.

In this setting, the client not only wants to store its data remotely but expects the server (or some third party) to perform some computation over the data. In particular, while the client is willing to reveal the information necessary for the server to perform its task, the client does not want to reveal anything else. Consider, e.g., a client that stores a large-scale social network remotely and that, at some point, needs the server to analyze a small subset of the network. If the social network were encrypted

[1] While the public-key encryption scheme with keyword search of [7] yields a SSE scheme that hides the access and query patterns, it is interactive.

using a classical encryption scheme the client would have to reveal the entire network, leaking extra information to the server. Ideally, what we want in this setting is a mechanism that allows the client to encrypt the data and later disclose the "pieces" of it that are necessary for the server to perform its task.

Another application of controlled disclosure is to the emerging area of (cloud-based) data brokerage services, such as Microsoft's Windows Azure Marketplace [14] and Infochimps [23]. Here, the cloud provider acts as a broker between a data provider that wishes to sell access to a dataset and a data consumer that needs access to the data. The data is stored "in the cloud" and the cloud operator manages the consumer's access to the provider's data. Using controlled disclosure, the provider could encrypt its data before storing it in the cloud and release tokens to the consumer as appropriate. Such an approach would have several advantages including (1) enabling the producer to get an accurate measure of the consumer's use of the data; and (2) ensuring the producer that the consumer can only access the authorized segments of data, even if the consumer and the cloud operator collude.

Clearly, if the algorithm executed by the server (or the data consumer) is "global", in the sense that it needs to read all the data, then controlled disclosure provides no security. On the other hand, if the algorithm is "local", in that it only needs to read part of the data, then controlled disclosure preserves the confidentiality of the remaining data. There are numerous algorithms that exhibit this kind of local behavior and they are used extensively in practice to solve a variety of problems. For example, many optimization problems like the traveling salesman problem or vertex cover are handled in practice using local search algorithms (e.g., hill climbing, genetic algorithms or simulated annealing). Several link-analysis algorithms for web search such as Kleinberg's seminal HITS algorithm [27] (and the related SALSA [28] algorithm) are local. Finally, the recent work of Brautbar and Kearns on "jump and crawl" algorithms [11] motivates and proposes several local algorithms for social network analysis, including for finding vertices with high-degree and high clustering coefficient.

Controlled disclosure can be viewed as a compromise between full security on the one hand and efficiency and functionality on the other. In settings where computation needs to be performed on massive datasets and "fully secure" solutions like multi-party computation [37, 19, 13] and fully-homomorphic encryption [17] are prohibitively expensive, controlled disclosure provides a practical solution without completely compromising security.

1.2 Our Results

Performing private queries on encrypted data is an important goal that is well motivated by the recent trend towards cloud storage. Giving clients the means to encrypt their data without losing the ability to efficiently query and retrieve it provides obvious benefits to the client but also frees the cloud provider from many legal exposures (see [2, 24, 34] for discussion of these issues). It additionally provides a mechanism by which clients from regulated industries can make use of cloud storage (e.g., to store medical records or financial documents) while remaining compliant.

While the recent work on searchable encryption constitutes an important step towards this goal, we note that a noticeable fraction of the data generated today is *not* text data. Indeed, many large-scale datasets (e.g., image collections, social network data, maps or location information) exhibit a different and sometimes more complex structure that cannot be handled properly using searchable encryption. To address this, we:

1. introduce the notion of structured encryption, which generalizes index-based symmetric searchable encryption [18, 12, 15] to arbitrarily-structured data and propose a novel application of structured encryption (and therefore of SSE) to the problem of controlled disclosure. We also provide a detailed comparison of structured encryption to the recent paradigm of functional encryption.

2. extend the adaptive security definition of [15] to the setting of structured encryption,

3. give constructions of adaptively-secure structured encryption schemes for a variety of structures and queries including:

   (a) (lookup queries on matrix-structured data) given a matrix and pair (i, j), return the value stored at row i and column j. This captures, e.g., lookup queries on digital images or retrieval of maps.

   (b) (search queries on labeled data) given a set of labeled items and keyword w, return the items labeled with w. This captures the familiar setting of searchable encryption. We briefly note that our construction exhibits a combination of useful properties that, as far as we know, no previous scheme achieves.

   (c) (neighbor queries on graph-structured data) given a graph and a node i, return all the nodes adjacent to i. This captures, e.g., retrieving a user's "friend list" in a social network.

   (d) (adjacency queries on graph-structured data) given a graph and two nodes i and j, return 1 if they are adjacent and 0 otherwise. This captures, e.g., testing whether two users are "friends" in a social network.

While the previous constructions are useful in their own right, an important goal with respect to structured encryption is to construct schemes that are able to encrypt complex structures and to handle expressive queries that take full advantage of the complexity of the data's structure. As an example, consider the case of web graphs (i.e., subgraphs of the Web) which are composed of pages with both text and hyperlinks. Encrypting the pages of a web graph using a searchable encryption scheme will only enable keyword search over the encrypted pages. Web graphs, however, exhibit a much richer structure and we typically want to perform more complex queries on them. Towards this goal, our final contribution is to show how to encrypt web graphs and, more generally, what we refer to as *labeled graph data*. In particular, we:

4. give a structured encryption scheme for labeled graphs that handles *focused subgraph* queries. Roughly speaking, for a given search keyword, a focused subgraph query on a web graph returns a subgraph that encodes enough information about it to yield a good ranking of the pages for that search. These queries are an essential part of Kleinberg's seminal HITS algorithm [27] (and its many successors).

   Our construction uses as building blocks some of the schemes mentioned above. We stress, however, that it is not sufficient to use the schemes "as-is" and we show a novel way of combining structured encryption schemes for simple structures in order to build schemes that handle more complex data and more expressive queries. The approach is general and can be adapted to other complex data types.

2 Related Work

We already mentioned work on oblivious RAMs, secure two-party computation and FHE so we restrict the following discussion to searchable and functional encryption.

Searchable encryption. Searchable encryption was first explicitly considered by Song, Wagner and Perrig [35]. Structured encryption is a generalization of the notion of a secure index first proposed by Goh [18] for the purpose of building symmetric searchable encryption schemes. In [18], Goh gives a

game-based security definition for secure indexes and a construction based on Bloom filters. Search is linear in the number of documents and it returns false positives. This was followed by [12] who gave a simulation-based definition and a construction with linear search time and no false positives. In [15], it was observed that previous security definitions were insufficient for the setting of SSE and the stronger notion of security against adaptive chosen-keyword attacks was introduced and formalized. Furthermore, it was shown in [15] how to achieve sub-linear search time (in fact, optimal time) without false positives. Our notion of adaptive security for structured encryption in Section 4 generalizes the one in [15] to arbitrarily-structured data. Searchable encryption has also been considered in the public-key setting [5, 36, 1, 10, 7, 9].

Functional encryption. Functional encryption [8] is a recent paradigm that generalizes work on a variety of problems including identity-based encryption [31, 6], attribute-based encryption [30, 22, 4], and predicate encryption [26, 32].

Using a functional encryption scheme a user can encrypt messages and, independently, generate tokens for various functions. Any party holding a token can then apply it to any ciphertext to evaluate the corresponding function on the encrypted message, without learning any additional information. In some cases it is also required that the function itself be hidden [32]. Roughly speaking, a structured encryption scheme can be viewed as a functional encryption scheme for which each token can only be used on a single ciphertext. We provide a more detailed comparison between the two approaches in Appendix A.

3 Notation and Preliminaries

Notation. The set of all binary strings of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$. $[n]$ is the set of integers $\{1,\ldots,n\}$, and $\mathcal{P}[n]$ is the corresponding power set. We will use $\emptyset$ or $\perp$ to refer to the empty set. We write $x \leftarrow \chi$ to represent an element x being sampled from a distribution χ, and $x \xleftarrow{\$} X$ to represent an element x being sampled uniformly at random from a set X. The output x of an algorithm $\mathcal{A}$ is denoted by $x \leftarrow \mathcal{A}$. Given a sequence $\mathbf{v}$ of n elements, we refer to its i^{th} element as v_i. If S is a set then $|S|$ refers to its cardinality. On the other hand, if s is a string then $|s|$ refers to its bit length. If f is a function with domain $\mathcal{U}$ and $S \subseteq \mathcal{U}$, then $f[S]$ refers to the image of S under f. The set of all $\lambda_1 \times \lambda_2$ matrices over a set S is denoted $S^{\lambda_1 \times \lambda_2}$. $\mathcal{G}_n$ and $\vec{\mathcal{G}}_n$ are the sets of all undirected and directed graphs of size n, respectively. An undirected graph $G = (V, E)$ consists of a set of vertices V and a set of edges $E = \{(i, j)\}$ where $i, j \in V$. We denote by $\deg(i)$ the degree of node i. If G is directed, then the pairs (i, j) are ordered and we refer to i as the tail and to j as the head of the edge. In addition, we denote i's in and out degrees by $\deg^-(i)$ and $\deg^+(i)$, respectively.

Throughout, $k \in \mathbb{N}$ will denote the security parameter and we will assume all algorithms take k implicitly as input. A function $\nu : \mathbb{N} \to \mathbb{N}$ is negligible in k if for every positive polynomial $p(\cdot)$ and all sufficiently large k, $\nu(k) < 1/p(k)$. We write $f(k) = \mathsf{poly}(k)$ to mean that there exists a polynomial $p(\cdot)$ such that $f(k) \leq p(k)$ for all sufficiently large $k \in \mathbb{N}$; and similarly write $f(k) = \mathsf{negl}(k)$ to mean that there exists a negligible function $\nu(\cdot)$ such that $f(k) \leq \nu(k)$ for all sufficiently large k.

Data types. An *abstract data type* is a collection of objects together with a set of operations defined on those objects. Examples include sets, dictionaries (also known as key-value stores or associative arrays) and graphs. The operations associated with an abstract data type fall into one of two categories: query operations, which return information about the collection; and update operations, which modify the collection. If the abstract data type supports only query operations it is *static*, otherwise it is

dynamic. In this work we only consider static data types. A data *structure* is a particular representation of a data type $\mathscr{T}$. For example, the dictionary data type can be represented using various data structures, including linked lists, hash tables and binary search trees.

For simplicity and visual clarity we define data types as having a single operation but this can be extended to model data types with multiple operations in the natural way. Formally, a data type $\mathscr{T}$ is defined by a universe $\mathcal{U} = \{\mathcal{U}_k\}_{k\in\mathbb{N}}$ and an operation $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to \mathcal{R}$, where $\mathcal{Q} = \{\mathcal{Q}_k\}_{k\in\mathbb{N}}$ is the operation's query space and $\mathcal{R} = \{\mathcal{R}_k\}_{k\in\mathbb{N}}$ is its response space. The universe, query and response spaces are ensembles of finite sets indexed by the security parameter k. In this work, we assume the universe is a totally ordered set, and that the response space includes a special element $\perp$ denoting failure.

Basic cryptographic primitives. A private-key encryption scheme is a set of three polynomial-time algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that Gen is a probabilistic algorithm that takes a security parameter k and returns a secret key K; Enc is a probabilistic algorithm takes a key K and a message m and returns a ciphertext c; Dec is a deterministic algorithm that takes a key K and a ciphertext c and returns m if K was the key under which c was produced. Informally, a private-key encryption scheme is CPA-secure if the ciphertexts it outputs do not reveal any partial information about the plaintext even to an adversary that can adaptively query an encryption oracle. A private-key encryption scheme has an elusive range if for all keys K, $\mathsf{Dec}\big(K', \mathsf{Enc}(K, m)\big)$ returns $\perp$ with all but negligible probability (in k) over the choice of K'. We refer the reader to [29] for a formal definition and an efficient construction.

In addition to encryption schemes, we also make use of pseudo-random functions (PRF) and permutations (PRP), which are polynomial-time computable functions that cannot be distinguished from random functions by any probabilistic polynomial-time adversary. We refer the reader to [25] for formal definitions of CPA-security, PRFs and PRPs.

4 Definitions

In this section we formalize structured encryption schemes and present our main security definition. Before doing so, however, we make explicit two properties of structured encryption which we will make use of throughout this work.

Induced permutation. Unlike most previous work on searchable encryption we choose to include the data items (i.e., the documents in the case of searchable encryption) and their encryptions in our definitions. We prefer this approach because explicitly capturing each component of the system can bring to light subtle interactions between them. As an example, consider the correlation between the location of the data items in the sequence $\mathbf{m}$ and the locations of their corresponding ciphertexts in $\mathbf{c}$. More precisely, let π be the permutation over $[n]$ such that for all $i \in [n]$, $m_i := \mathsf{Dec}_K(c_{\pi(i)})$. We refer to π as the permutation *induced* by $\mathbf{m}$ and $\mathbf{c}$.

We note that, as far as we know, all SSE constructions (with the exception of oblivious RAMs) use an induced permutation that is the identity function. This means that in order to efficiently retrieve items $\{m_i : i \in I\}$ the server must know I, and as a result all these constructions reveal the access pattern [2]. Our constructions hide part of the access pattern essentially because they break this correlation by inducing a random permutation between $\mathbf{m}$ and $\mathbf{c}$.

[2] Recall that the access pattern for a set of queries on $(\delta, \mathbf{m})$ consists of the set of pointers produced by each query on δ.

redefine m

Associativity. We also make explicit a property possessed by some constructions (e.g., the non-adaptively secure SSE construction of [15]) that we refer to as *associativity*. Intuitively, a scheme is associative if one can associate an item v_i with data item m_i in such a way that a query operation returns, in addition to the pointers $J = \pi[I]$, the strings $(v_i)_{i \in I}$. We capture this by re-defining the message space of our encryption algorithms to take, in addition to a data structure δ, a sequence $\mathbf{M} = ((m_1, v_1), \ldots, (m_n, v_n))$ of pairs that consist of a private data item m_i and a semi-private [3] item v_i. We sometimes refer to the sequences $(m_1, \ldots, m_n)$ and $(v_1, \ldots, v_n)$ as $\mathbf{m}$ and $\mathbf{v}$, respectively.

Associativity is useful for several reasons. The most direct application is to provide the client the ability to associate some meta-data with the ciphertexts that may be useful to the server (e.g., file name or size). In situations where the client wishes to grant the server access to the data, the semi-private items could even be decryption keys for the associated ciphertexts. As we will see in Section 6, however, associativity can also be used to "chain" structured encryption schemes together in order to construct complex schemes from simpler ones.

Definition 4.1 (Private-key structured encryption)**.** *Let $\mathcal{T}$ be an abstract data type supporting operation* $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to \mathcal{R}$ *where $\mathcal{R} = \mathcal{P}[n]$ for $n \in \mathbb{N}$, and let $\mathcal{M}sg$ be a message space. An associative private-key structured encryption scheme for $\mathcal{T}$ and $\mathcal{M}sg$ is a tuple of five polynomial-time algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Query_e}, \mathsf{Dec})$ *such that:*

> $K \leftarrow \mathsf{Gen}(1^k)$*: is a probabilistic algorithm that takes as input a security parameter k and outputs a private key K.*
>
> $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \delta, \mathbf{M})$*: is a probabilistic algorithm that takes as input a private key K, a data structure δ of type $\mathcal{T}$, and sequences of private and semi-private data $\mathbf{M} \in \mathcal{M}sg$. It outputs an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$. We sometimes write this as $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}_K(\delta, \mathbf{M})$.*
>
> $\tau \leftarrow \mathsf{Token}(K, q)$*: is a (possibly probabilistic) algorithm that takes as input a private key K and a query $q \in \mathcal{Q}$ and outputs a token τ. We sometimes write this as $\tau \leftarrow \mathsf{Token}_K(q)$.*
>
> $(J, \mathbf{v}_I) := \mathsf{Query_e}(\gamma, \tau)$*: is a deterministic algorithm that takes as input an encrypted data structure γ and a token τ. It outputs a set of pointers $J \subseteq [n]$ and a sequence of semi-private data $\mathbf{v}_I = (v_i)_{i \in I}$, where $I = \pi^{-1}[J]$.*
>
> $m_j := \mathsf{Dec}(K, c_j)$*: is a deterministic algorithm that takes as input a secret key K and a ciphertext c_j and outputs a message m_j. We sometimes write this as $m_j := \mathsf{Dec}_K(c_j)$.*

We say that Π is correct if for all $k \in \mathbb{N}$, for all K output by $\mathsf{Gen}(1^k)$, for all $\delta \in \mathcal{U}_k$, for all $\mathbf{M} \in \mathcal{M}sg$, for all $(\gamma, \mathbf{c})$ output by $\mathsf{Enc}(K, \delta, \mathbf{M})$, there exists a permutation π such that for all $q \in \mathcal{Q}_k$, for all τ output by $\mathsf{Token}(K, q)$, for $(J, \mathbf{v}_I)$ output by $\mathsf{Query_e}(\gamma, \tau)$

$$J = \pi\left[\mathsf{Query}(\delta, q)\right] \bigwedge \mathsf{Dec}_K(c_j) = m_{\pi^{-1}(j)} \text{ for all } j \in [n],$$

where π is the permutation induced by $\mathbf{m}$ and $\mathbf{c}$. If in addition $\mathsf{Dec}_{K'}(c_j) \neq \perp$ with at most negligible probability over the choice of K' then we say that Π is elusive[4]*.*

[3] We refer to the items $(v_1, \ldots, v_n)$ as semi-private since, unlike $(m_1, \ldots, m_n)$, they can be recovered given an appropriate token.

[4] Here we use the idea of symmetric key encryption with elusive range from [29].

4.1 Alternative Definitions

In addition to the definition given above, there are at least three alternative definitions of structured encryption that may be useful in certain settings. All of the constructions we present could be converted to satisfy any of these definitions. We will sometimes refer to the above formulation as a "pointer-output" scheme.

Structure-only. This variant of structured encryption considers only data structures (i.e., it does not encrypt any associated data items). Most searchable encryption schemes are structure-only schemes [18, 12, 15, 5] and assume that any associated data items can be handled separately using a private- or public-key encryption scheme.

Structure-only schemes consist of only four algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Query_e})$ such that Gen and Token are defined as above and Enc and $\mathsf{Query_e}$ are as follows:

> $\gamma \leftarrow \mathsf{Enc}(K, \delta)$: is a probabilistic algorithm that takes as input a key K and a data structure δ and outputs an encrypted data structure γ.
>
> $a := \mathsf{Query_e}(\gamma, \tau)$: is a deterministic algorithm that takes as input an encrypted data structure γ and a token τ and outputs an answer $a \in \mathcal{R} \subseteq \mathcal{P}[n]$.

Correctness for structure-only schemes requires that for all $k \in \mathbb{N}$, for all K output by $\mathsf{Gen}(1^k)$, for all $\delta \in \mathcal{U}_k$, for all γ output by $\mathsf{Enc}(K, \delta)$, for all $q \in \mathcal{Q}_k$, for all τ output by $\mathsf{Token}(K, q)$, $\mathsf{Query_e}(\gamma, \tau) = \mathsf{Query}(\delta, q)$

Ciphertext-output. In a ciphertext-output scheme, $\mathsf{Query_e}$ takes as input γ, τ and $\mathbf{c}$ and returns a ciphertexts instead of a set of pointers. Correctness is re-defined to require that this ciphertext be the encryptions of the appropriate data items. Any pointer-output schemes can be easily transformed into ciphertext-output schemes by simply requiring the encryption algorithm to return the ciphertexts indexed by the returned pointers.

Plaintext-output. In a plaintext-output scheme, the $\mathsf{Query_e}$ algorithm takes as input γ, τ and $\mathbf{c}$ and returns unencrypted data items. We note that this notion leads to strictly weaker security than those above, as the party performing the $\mathsf{Query_e}$ algorithm will learn the plaintext of all the data items returned. Any associative pointer-output scheme can be converted into a plaintext-output scheme by encrypting each data item with a different private key, and then including these keys as the corresponding semi-private data.

4.2 Security of Structured Encryption

The intuitive security guarantee we seek is that (1) given an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$, no adversary can learn any partial information about $\mathbf{m}$; and that (2) given, in addition, a sequence of tokens $\tau = (\tau_1, \ldots, \tau_t)$ for an adaptively generated sequence of queries $\mathbf{q} = (q_1, \ldots, q_t)$, no adversary can learn any partial information about either $\mathbf{m}$ or $\mathbf{q}$ beyond what is revealed by the semi-private data $(\mathbf{v}_{I_1}, \ldots, \mathbf{v}_{I_t})$.

This exact intuition can be difficult to achieve and in some settings is unnecessarily strong. Consider, e.g., the fact that the number of data items n is immediately revealed to the adversary since it receives the ciphertexts $\mathbf{c}$. Another example is in the setting of SSE where, as discussed earlier, all known efficient and non-interactive schemes [18, 12, 15] reveal the access and query patterns. We would therefore like to weaken the definition appropriately by allowing some limited information about the

messages and the queries to be revealed. On the other hand, it is not clear that such leakage is always necessary in order to achieve efficiency (e.g., the number of data items can be easily hidden by padding) so we prefer not to "hardcode" this leakage in our definition. To formalize this we parameterize the definition with two stateful leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$ that capture precisely what is being leaked by the ciphertext and the tokens.

We now present our security definition for adaptive adversaries which is a generalization of the definition of [16]. Intuitively, we require that the view of an adversary (i.e., the encrypted data structure, the sequence of ciphertexts, and the sequence of tokens) generated from any adaptive query strategy be simulatable given the leakage information and the semi-private data. Here, we define security for associative schemes but a similar definition for a non-associative schemes can be easily derived by omitting all mention of the semi-private information.

Definition 4.2 (CQA2-security)**.** *Let* $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Query_e}, \mathsf{Dec})$ *be an associative private-key structured encryption scheme for data of type* $\mathscr{T}$ *supporting operation* $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to \mathcal{P}[n]$*, for some* $n \in \mathbb{N}$*, and consider the following probabilistic experiments where* $\mathcal{A}$ *is an adversary,* $\mathcal{S}$ *is a simulator and* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *are (stateful) leakage algorithms:*

Real$_{\Sigma,\mathcal{A}}(k)$*: the challenger begins by running* $\mathsf{Gen}(1^k)$ *to generate a key* K*.* $\mathcal{A}$ *outputs a tuple* $(\delta, \mathbf{M})$ *and receives* $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}_K(\delta, \mathbf{M})$ *from the challenger. The adversary makes a polynomial number of adaptive queries and, for each query* q*, receives a token* $\tau \leftarrow \mathsf{Token}_K(q)$ *from the challenger. Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

Ideal$_{\Sigma,\mathcal{A},\mathcal{S}}(k)$*:* $\mathcal{A}$ *outputs a tuple* $(\delta, \mathbf{M})$*, where* $\mathbf{M}$ *is composed of sequences* $\mathbf{m}$ *and* $\mathbf{v}$ *as described above. Given* $\mathcal{L}_1(\delta, \mathbf{M})$*,* $\mathcal{S}$ *generates and sends a pair* $(\gamma, \mathbf{c})$ *to* $\mathcal{A}$*. The adversary makes a polynomial number of adaptive queries and for each query* q *the simulator is given* $(\mathcal{L}_2(\delta, q), \mathbf{v}_I)$*, where* $I := \mathsf{Query}(\delta, q)$*. The simulator returns a token* τ*. Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

We say that Σ *is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against adaptive chosen-query attacks if for all* PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *such that*

$$|\Pr[\mathbf{Real}_{\Sigma,\mathcal{A}}(k) = 1] - \Pr[\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k) = 1]| \leq \mathsf{negl}(k).$$

Remark (Lower bound on token length). We note that this type of definition automatically implies some limits on the efficiency of the scheme. In the **Ideal** experiment, the simulator must first generate a pair $(\gamma, \mathbf{c})$ without seeing the data to be encrypted. Then, for an arbitrary query q, it must produce an appropriate token τ, i.e., one for which $\mathsf{Query_e}(\gamma, \tau)$ outputs values $(J, \mathbf{v}_I)$ that are consistent with the semi-private information and the leakage. Note that the simulator must be able to do this no matter which δ the adversary has chosen. Thus, suppose the algorithm Token is stateless. Then, if we let A_q be the set of consistent output values that result from applying query q to all $\delta \in \mathcal{U}$, then for any value in A_q, the simulator must be able to produce an appropriate token. This directly implies that the space from which the simulated tokens are drawn must be at least as large as A_q (or smaller by at most a negligible fraction). This in turn means that the length of the token for a given query q must be at least $\log(|A_q| \cdot (1 - \mathsf{negl}(k)))$ [5]. If Token is not stateless, then this lower bound still applies to the token generated on the first query q and lower bounds for subsequent tokens can be derived similarly.

[5]Note that this does not hold in the random oracle model, where the simulator has the additional flexibility of programming the random oracle to help guarantee that $\mathsf{Query_e}(\gamma, \tau) = a$.

For example, if we consider encryption schemes that handle search queries on labeled data (i.e. the setting of SSE), and make no restrictions on the possible association of labels to documents, then the set A_q contains all subsets of documents, and the length of the tokens must be nearly as large as the number of documents. Similarly, if we consider encrypted graphs that allow for neighborhood queries and expect the scheme to work for arbitrary graphs of any degree, then the set A_q contains all subsets of vertices and again the length must be as large as the number of vertices stored. More generally, if we only require the scheme to work for graphs of degree d, then A_q consists of all subsets of up to d vertices.

Leakage. The $\mathcal{L}_2$ leakage of our constructions mainly consist of the query and intersection patterns. Intuitively, the query pattern reveals when a query is repeated while the intersection pattern reveals when the same items are accessed. Note that the intersection pattern reveals when the same items are accessed but not *which* items are accessed (i.e., their locations in $\mathbf{m}$). The latter is hidden in our definition below by applying a random permutation to the item's locations in $\mathbf{m}$.

Definition 4.3 (Query and intersection patterns)**.** *Let* $\mathbf{q}$ *be a non-empty sequence of queries. For any* $q_t \in \mathbf{q}$*, the query pattern* $\text{QP}(q_t)$ *is a binary vector of length* t *with a* 1 *at location* i *if* $q_t = q_i$ *and a* 0 *otherwise. The intersection pattern* $\text{IP}(q_t)$ *is a sequence of length* t *with* $f[I]$ *at location* t*, where* f *is a fixed random permutation over* $[n]$ *and* $I := \mathsf{Query}(\delta, q_t)$*.*

5 Structured Encryption for Basic Structures

In this Section we present constructions of structured encryption schemes for data with simple structures. In Section 6 we will use some of these as building blocks to design schemes for data that exhibits a more complex structure. We stress, however, that the constructions presented here are of independent interest.

5.1 Lookup Queries on Matrices

We describe a structured encryption scheme for matrix-structured data which consists of an $\lambda_1 \times \lambda_2$ matrix M of pointers into a sequence of n data items $\mathbf{m}$. Here, the matrix type has universe $\mathcal{U} = [n]^{\lambda_1 \times \lambda_2}$ and supports the lookup operation $\mathsf{Lkp} : [n]^{\lambda_1 \times \lambda_2} \times [\lambda_1] \times [\lambda_2] \to [n]$ that takes as input a matrix M and a pair (α, β) and returns $M[\alpha, \beta]$.

Matrix-structured data is ubiquitous and includes any kind of two-dimensional data. Consider, e.g., the case of digital images which can be viewed as a pair $(M, \mathbf{m})$, where M is a matrix such that the cell at location (α, β) points to some m_i that encodes the color of the pixel at location (α, β) in the image.

Our construction, described in Figure 1 below, is associative. At a high level, encryption is done by (1) padding the data items to be of the same length; (2) randomly permuting the location of the data items, (3) randomly permuting the location of the matrix cells using a PRP; and (4) encrypting the contents of the cells (and the semi-private data) using the output of a PRF. The purpose of the last two steps are immediate. Steps (1) and (2) are what allow us hide part of the access pattern by inducing a pseudo-random permutation between $\mathbf{m}$ and $\mathbf{c}$.

Lookup queries are handled by sending the permuted location of a cell (which can be recovered by the client since it stores the key to the PRP) and the output of the PRF used to encrypt the contents (which can also be recovered since the client stores the key to the PRF).

Let ω, n, ℓ be integers and let $\mathcal{M}sg$ be a message space such that for all $(\mathbf{m}, \mathbf{v}) \in \mathcal{M}sg$, $|\mathbf{m}| = |\mathbf{v}| = n$, and $|v_i| \leq \omega$, and $|m_i| \leq \ell$. Let $F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^{\log n + \omega}$ be a pseudo-random function, $P : \{0,1\}^k \times [\lambda_1] \times [\lambda_2] \to [\lambda_1] \times [\lambda_2]$ be pseudo-random permutation and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Our encryption scheme $\mathsf{Matrix} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Lkp_e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate two random k-bit keys K_1, K_2 and a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$. Set $K := (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, M, \mathbf{M})$: construct a $\lambda_1 \times \lambda_2$ matrix C as follows:
  1. parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$
  2. choose a random permutation $\pi : [n] \to [n]$
  3. for all $(\alpha, \beta) \in [\lambda_1] \times [\lambda_2]$,

     store $\langle \pi(i), v_i \rangle \oplus F_{K_1}(\alpha, \beta)$ where $i := M[\alpha, \beta]$, at location $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$ in C.

     If $M[\alpha, \beta] = \perp$, then $\langle \pi(i), v_i \rangle$ above is replaced with a random string of appropriate length.

  Let $\mathbf{m}^*$ be the sequence that results from padding the elements of $\mathbf{m}$ so that they are of the same length and permuting them according to π. For $1 \leq j \leq n$, let $c_j \leftarrow \Pi.\mathsf{Enc}_{K_3}(m^*_j)$. Output $\gamma := C$ and $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{Token}(K, \alpha, \beta)$: output $\tau := (s, \alpha', \beta')$, where $s := F_{K_1}(\alpha, \beta)$ and $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$.
- $\mathsf{Lkp_e}(\gamma, t)$: parse τ as (s, α', β'); compute and output $(j, v) := s \oplus C[\alpha', \beta']$.
- $\mathsf{Dec}(K, c_j)$: return $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Figure 1: An associative structured encryption scheme for matrices.

In Theorem 5.1 below we show that the construction above is secure against adaptive chosen-query attacks.

Theorem 5.1. *If F and P are pseudo-random and if Π is CPA-secure then Matrix is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, where $\mathcal{L}_1(M, \mathbf{M}) = (\lambda_1, \lambda_2, n, \ell)$ and $\mathcal{L}_2(M, \alpha, \beta) = (\textsc{qp}(\alpha, \beta), \textsc{ip}(\alpha, \beta))$.*

Proof Sketch. We define a simulator that proceeds as follows. Given $\mathcal{L}_1(M, \mathbf{M}) = (\lambda_1, \lambda_2, n, \ell)$, the simulator generates a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$ and a $\lambda_1 \times \lambda_2$ matrix C filled with random values. It then computes ciphertexts $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i \leftarrow \Pi.\mathsf{Enc}_{K_3}(0^\ell)$. For each query q, recall that the simulator is given $(\mathcal{L}_2(M, q), 1, v)$. It uses the query pattern in $\mathcal{L}_2(M, q)$ to determine if the query is new. If not, it outputs the same token that it generated previously otherwise it uses the intersection pattern to determine whether the location accessed by the current query has been accessed before. If not, it sets j to be a random value in $[n]$ that has not been used yet, otherwise it sets j to be the index chosen previously. It then chooses a random location (α, β) that has not yet been used and produces token $\tau = (s, \alpha, \beta)$, where $s = C[\alpha, \beta] \oplus (j, v)$.

To show that this simulator satisfies our security definition, we first argue that the random choices of α, β and j will be indistinguishable from the outputs of P and π. Then we argue that since all entries of the matrix C are chosen at random, $s = C[\alpha, \beta] \oplus (j, v)$ will be randomly distributed and thus indistinguishable from the output of the PRF F. The result follows directly. □

5.2 Search Queries on Labeled Data

We now present a structured encryption scheme for labeled data which consists of a "labeling" L and a sequence of data items $\mathbf{m}$. Informally, a labeling just associates a set of keywords to each data item.

More formally, the labeling data type has as universe $\mathcal{U}$ the set of all binary relations between $[n]$ and W, where W is a set of keywords. In addition, it supports the operation $\mathsf{Search} : \mathcal{U} \times W \to \mathcal{P}[n]$ that takes as input a labeling and a keyword w and returns the set $L(w) = \{i \in [n] : (i, w) \in L\}$.

Our construction, described in Figure 2, is efficient, associative and adaptively secure and, as far as we know, is the first scheme to achieve all three properties. It is based on the first scheme of [15] (SSE-1) which is efficient and associative but not adaptively secure [6]. The second scheme of [15], on the other hand, is adaptively secure but is inefficient and not associative.

Our construction makes use of a dictionary which is a data structure that stores pairs (a, b) such that given a, the corresponding value b can be recovered efficiently. We refer to a as the "search key" and to b as the value. Dictionaries can be implemented in a variety of ways, including using search trees or hash tables. Intuitively, encryption proceeds as follows in our scheme. As in our previous construction, we pad and permute the data items with a random permutation π. For each keyword w an array is constructed where each cell stores (1) a pointer j from the set $L^*(w) = \pi[L(w)]$ and (2) the corresponding semi-private item v_i. The array is then padded up to a standard length, encrypted using the output of a PRF and is stored in a dictionary using as search key the output of another PRF on the keyword. Search queries are handled by sending the search key (which can be recovered by the client using the key to the second PRF) and the output of the PRF used to encrypt the array (which can be recovered using the key to the first PRF). The efficiency of the search operation depends on how the underlying dictionary is implemented but in this context any solution based on hash tables is appropriate and will give search time that is $O(|I|)$, which is optimal.

Let $|L|$ be the number of words w such that $L(w)$ is nonempty, and let $\max(L)$ be the size of the largest set $L(w)$.

Theorem 5.2. *If F and G are pseudo-random and if Π is CPA-secure then Label is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, where $\mathcal{L}_1(L, \mathbf{M}) = (|L|, \max(L), n, \ell)$ and $\mathcal{L}_2(L, w) = (|I|, \text{QP}(w), \text{IP}(w))$.*

Proof Sketch. We define a simulator that proceeds as follows. Given $\mathcal{L}_1(L, \mathbf{M}) = (|L|, \max(L), n, \ell)$, the simulator generates a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$, constructs a dictionary T that holds n random pairs (κ, s), and computes ciphertexts $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i \leftarrow \Pi.\mathsf{Enc}_{K_3}(0^\ell)$. For each query w, recall that the simulator receives $(\mathcal{L}_2(L, w), \mathbf{v}_I)$. It uses the query pattern from $\mathcal{L}_2(L, w)$ to determine if the query is new. If not, it returns the token it generated previously, otherwise we consider two cases. If $I = \emptyset$, it randomly chooses a search key κ that is not stored in T and a random value t and outputs token $\tau = (t, \kappa)$. Otherwise, it uses the intersection pattern to determine whether the items accessed by the current query have been accessed before. If not, it sets J to be a set of $|I|$ random locations that have not been used, otherwise it sets J to include all the locations that have been previously accessed together with enough random new locations so that $|J| = |I|$. It then randomly chooses a search key κ in T that has not yet been used and returns the token $\tau = (t, \kappa)$, where $t := (J, \mathbf{v}_I) \oplus s$ and s is the value associated with κ in T.

To show that this simulator satisfies our security definition, we first argue that each random choice of κ and j will be indistinguishable from the output of G and π. Then we argue that since all values s in the table T are chosen at random, $t = s \oplus (J, \mathbf{v}_I)$ will be randomly distributed and thus indistinguishable from the output of PRF F. The result follows directly. □

[6] While our scheme achieves the same efficiency as SSE-1 with respect to search time, SSE-1 is more efficient with respect to storage.

Let ω, n, ℓ be integers and let $\mathcal{M}sg$ be a message space such that for all $(\mathbf{m}, \mathbf{v}) \in \mathcal{M}sg$, $|\mathbf{m}| = |\mathbf{v}| = n$, and $|v_i| \leq \omega$, and $|m_i| \leq \ell$. Let $F : \{0,1\}^k \times W \to \{0,1\}^{\max(L)\cdot(\log n + \omega)}$ and $G : \{0,1\}^k \times W \to \{0,1\}^k$ be pseudo-random functions and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Our scheme $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: sample two random k-bit keys K_1, K_2, and generate a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$. Set $K := (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, L, \mathbf{M})$: construct a dictionary T as follows:

  1. parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$.
  2. choose a random permutation $\pi : [n] \to [n]$.
  3. for each $w \in W$ such that $L(w) \neq \emptyset$, let $\kappa_w := G_{K_2}(w)$ and

     store $\langle (\pi(i), v_i)_{i \in L(w)} \rangle \oplus F_{K_1}(w)$ in T with search key κ_w.

     Use padding to ensure that the strings $\langle (\pi(i), v_i)_{i \in L(w)} \rangle$ are all of the same length.

  Let $\mathbf{m}^*$ be the sequence that results from padding the elements of $\mathbf{m}$ so that they are of the same length and permuting them according to π. For $1 \leq j \leq n$, let $c_j \leftarrow \Pi.\mathsf{Enc}_{K_3}(m^*_j)$. Output $\gamma := T$ and $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{Token}(K, w)$: output $\tau := (F_{K_1}(w), G_{K_2}(w))$.
- $\mathsf{Search}_\mathsf{e}(\gamma, \tau)$: parse τ as (α, β) and compute $s := T(\beta) \oplus \alpha$, where $T(\beta)$ refers to the value stored in T with search key β. If β is not in T then output $J = \emptyset$ and $v_I = \perp$. Otherwise parse s as $\langle (j_1, v_{i_1}), \ldots, (j_t, v_{i_t}) \rangle$ and output $J = (j_1, \ldots, j_t)$ and $\mathbf{v}_I = (v_{i_1}, \ldots, v_{i_t})$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Figure 2: An associative structured encryption scheme for labeled data.

5.3 Neighbor Queries on Graphs

We now consider encryption of graph-structured data and, in particular, of graphs that support neighbor queries. Formally, the graph type we consider has universe $\mathcal{U} = \mathcal{G}_n$ and supports the neighbor operation $\mathsf{Neigh} : \mathcal{G}_n \times [n] \to \mathcal{P}[n]$ that takes as input an undirected graph G with n nodes and a node i and returns the nodes adjacent to i.

Our approach here is to encode the graph as a labeling and to apply a structured encryption scheme for labeled data (such as the one described in the previous Section). Given some graph-structured data $(G, \mathbf{m})$, where $G = (V, E)$, we construct the labeled data $(L, \mathbf{m})$ such that L assigns to each data item m_i a label set corresponding to the set of nodes adjacent to the ith node. Neighbor queries are handled by sending a token for "keyword" $i \in V$ which allows the server to recover pointers to all the data items associated with i by the labeling. Our construction is described in detail in Figure 3 below.

Theorem 5.3. *If* Label *is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against adaptive chosen-query attacks, then* Graph *is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against adaptive chosen-query attacks as well.*

The theorem follows by construction. Note that if Label is instantiated with the scheme from Section 5.2, then $\mathcal{L}_1$ leaks the size of the graph, the maximum degree, the number of data items and the length of the largest data item while $\mathcal{L}_2$ leaks the degree of the node and the query and intersection patterns.

We now discuss a slight variation of this construction to handle incoming and outgoing neighbor queries on directed graphs. This will be useful as a building block for the construction we describe in Section 6. An incoming neighbor query is: given a node i return all the nodes that point to it; and an outgoing neighbor query is: given a node i return all the nodes that it points to. We stress that the changes we describe do not affect security in any way.

Let $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ be an associative structured encryption scheme for labeled data. Our scheme $\mathsf{Graph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate and output $K \leftarrow \mathsf{Label.Gen}(1^k)$.
- $\mathsf{Enc}(K, G, \mathbf{M})$: parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$ and construct a labeling L that associates to each m_i the set $\{j \in [n] : (i,j) \in E\}$, where E is the set of edges in G. Output $(\gamma, \mathbf{c}) \leftarrow \mathsf{Label.Enc}_K(L, \mathbf{M})$.
- $\mathsf{Token}(K, i)$: compute and output $\tau \leftarrow \mathsf{Label.Token}_K(i)$.
- $\mathsf{Neigh}_\mathsf{e}(\gamma, \tau)$: output $J := \mathsf{Label.Search}(\gamma, \tau)$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \mathsf{Label.Dec}_K(c_j)$.

Figure 3: A structured encryption scheme for graphs supporting neighbor queries.

Consider the scheme $\mathsf{Graph}^+ = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ defined exactly as Graph except that the Enc algorithm constructs the labeling in the following manner: instead of associating a data item m_i to the set of nodes adjacent to node i, associate m_i to the nodes that are pointed to by node i. Similarly, a scheme Graph^- can be constructed by associating to data item m_i the set of nodes that point to node i.

5.4 Adjacency Queries on Graphs

In this Section we give a simple scheme to encrypt graphs supporting adjacency queries based on any matrix encryption scheme. The approach is straightforward and, at a high level, consists of encrypting the graph's adjacency matrix. Given data $(G, \mathbf{m})$, where $G = (V, E)$ is a directed graph of size n and each data item m_i is assigned to some edge in E, encryption proceeds as follows. We create a matrix M that holds at location (α, β) a pointer to the data item associated with edge $(\alpha, \beta) \in E$ (or $\perp$ when there is no such edge). We then use the matrix encryption scheme on $(M, \mathbf{m})$. Our construction is described in detail in Figure 4.

Let $\mathsf{Matrix} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Lkp}_\mathsf{e}, \mathsf{Dec})$ be an associative encryption scheme for matrix-structured data. Our scheme $\mathsf{Graph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Adj}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate and output $K \leftarrow \mathsf{Matrix.Gen}(1^k)$.
- $\mathsf{Enc}(K, G, \mathbf{M})$: construct a matrix M as follows: if $(\alpha, \beta) \in E$, then $M[\alpha, \beta]$ stores a pointer to the item assigned to edge (α, β); if $(\alpha, \beta) \notin E$ then $M[\alpha, \beta] = \perp$. Output $(\gamma, \mathbf{c}) \leftarrow \mathsf{Matrix.Enc}_K(M, \mathbf{M})$.
- $\mathsf{Token}(K, \alpha, \beta)$: compute and output $\tau \leftarrow \mathsf{Matrix.Token}_K(\alpha, \beta)$.
- $\mathsf{Adj}_\mathsf{e}(\gamma, \tau)$: output $J := \mathsf{Matrix.Lkp}_\mathsf{e}(\gamma, \tau)$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \mathsf{Matrix.Dec}_K(c_j)$.

Figure 4: A structured encryption scheme for graphs supporting adjacency queries.

Theorem 5.4. *If Matrix is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, then so is Graph.*

Again, the theorem follows by construction. If Matrix is instantiated with the construction from Section 5.1, then $\mathcal{L}_1$ leaks the size of the graph, the number of edges [7] the number of data items and the length of the largest data item. $\mathcal{L}_2$ leaks the query and intersection patterns.

[7] The number of edges can be hidden by padding $\mathbf{m}$ with $n^2 - |E|$ random strings whose lengths are distributed similarly to real data items.

6 Structured Encryption for Labeled Graphs

In this Section we describe an adaptively secure structured encryption scheme for data that is both labeled and associated with a graph-structure. As an example, consider a web graph where each page is labeled with a set of keywords (which could be the set of all the words in the page) and points to a set of other pages. Another example is social network data which consists of user profiles (with some associated meta-data) that link to other users.

While the constructions from the previous Section can be used to encrypt this type of data, the queries they support (i.e., keyword search, adjacency, and neighbor queries) are limited in this setting since they are only relevant to part of the data's structure. Indeed, if we were to encrypt a web graph using a scheme for labeled data, then we could only perform keyword search. Similarly, if we were to use a graph encryption scheme that supports only neighbor queries then we could only retrieve pages that are linked from a particular page. But web graphs, and labeled graph data in general, exhibit a much richer structure and ideally we would like to design schemes that support more complex queries that take advantage of this structure.

Focused subgraph queries. One example of complex queries on web graphs are focused subgraph queries. These queries are an essential part of a certain class of search engine algorithms which includes Kleinberg's seminal HITS algorithm [27] and the SALSA algorithm [28]. At a high level, they work as follows. Given a keyword w a keyword search is performed over the web pages. This results in a subset of pages called the *root* graph. A focused subgraph is then constructed by adding all the pages that either link to pages in the root graph or are linked from pages in the root graph. An iterative algorithm is then applied to the focused subgraph which returns, for each page, a score that quantifies its relevance with respect to keyword w. The key property of these "link-analysis" algorithms (and the reason for their success) is that they take advantage not only of the information provided by the keywords associated with the pages, but also of the implicit information embedded in the graph structure (i.e., the links) of the web graph. Here we will consider encryption schemes for labeled graphs that support focused subgraph queries. We first consider what can be obtained via a simple approach, then present our more complex construction.

A simple approach. The simplest approach to handling complex queries is to pre-compute the answers to all possible queries and to use a structured encryption scheme that handles lookups on dictionaries. The latter can be built using two pseudo-random functions F and P as follows. For each query/answer pair (q, a), insert the pair $(F_{K_1}(q), P_{K_2}(q) \oplus a)$ in a dictionary. A token for query q is $\tau := (\tau_1, \tau_2) := (F_{K_1}(q), P_{K_2}(q))$ and the answer is retrieved by querying the dictionary on τ_1 and XORing the result with τ_2. In the random oracle model, the token can be made as small as $2k$ by storing $(F_{K_1}(q), H(P_{K_2}(q)) \oplus a)$ and sending $F_{K_1}(q)$ and $P_{K_2}(q)$ as the token, where H is a random oracle and P outputs k-bit strings. When handling simple queries (e.g., lookups on matrices) the fact that each answer has to be pre-computed is not a real concern. When the queries are more complex, however, this pre-computation and the size of the encrypted data structure can become large.

Our approach. At a high level, our approach is to decompose the complex structure into simpler structures (e.g., in the case of a web graph into its graph and its labeling) and then use different structured encryption schemes to handle each "sub-structure". We note, however, that the sub-structures cannot be handled in isolation. In particular, for this approach to work the individual schemes have to be combined in a particular way. This is where we make essential use of associativity, which will allow us to "chain" the schemes together in order to obtain the functionality we want (this technique will be illustrated in our discussion below).

In order for this approach to work we require two additional properties from the underlying "simple" structured encryption schemes. The first requires that the $\mathcal{L}_1$ leakage (i.e., the information that is revealed by the encrypted data structure and ciphertexts) depend only on the data items and not on the semi-private data. This is because our approach (as described below) will be to use the semi-private data from one structured encryption scheme to hold tokens from a second. If the leakage from the first scheme is allowed to depend arbitrarily on the semi-private data, then this combination might reveal arbitrary information about these tokens, which in turn might reveal too much about the structure of the data.

The second property ensures that performing queries in different orders does not change the information that is leaked. This is essential because we will be pre-computing tokens for many queries on the second structured encryption scheme. These tokens will be revealed (potentially in a different order) as queries are made on the complex data structure so we need to guarantee that the tokens that we pre-compute will generate the same leakage that would have been revealed had the queries been performed "on the fly" as the complex data structure was being queried. We formalize these properties as follows.

Definition 6.1 (Chainability)**.** *A structured encryption scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Dec})$ *is chainable if it is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against adaptive chosen-query attacks and the leakage functions* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *satisfy the following properties:*

- *(semi-private independence) there exists some function* $\mathcal{L}_1'$ *such that*

$$\mathcal{L}_1(\delta, (\mathbf{m}, \mathbf{v})) = \mathcal{L}_1'(\delta, \mathbf{m}).$$

- *(order independence) there exists a transformation* $\mathcal{T}$ *such that for any data structure* δ*, for any set of queries* $q_1, \ldots, q_t$*, and for any permutation* p*,*

$$\mathcal{T}(\mathcal{L}_2(\delta, q_1), \ldots, \mathcal{L}_2(\delta, q_t), p) = (\mathcal{L}_2(\delta, q_{p(1)}), \ldots, \mathcal{L}_2(\delta, q_{p(t)})).$$

We note that all the constructions presented in the previous sections are chainable.

Our construction. We now illustrate our second approach for the case of web graphs but note that our construction applies to any labeled graph data. A detailed description of our construction is given in Figure 5. We note that it is not associative. A web graph will be viewed as a tuple $(G, L, \mathbf{m})$, which consists of a directed graph $G \in \vec{\mathcal{G}}_n$ of size n, a labeling L over a keyword space W, and text pages $\mathbf{m}$. The graph G encodes the link structure of the web graph and the labeling assigns keywords to each page [8]. The focused subgraph operation $\mathsf{Subgraph} : \vec{\mathcal{G}}_n \times W \to \mathcal{G}_{\leq n}$ takes as input a directed graph G of size n and a keyword w and returns the subgraph $G(w)$ that consists of (1) the nodes i in $L(w)$; (2) any node that links to the nodes in $L(w)$; and (3) any node that is linked from the nodes in $L(w)$.

Our construction makes use of three structured encryption schemes: Label that supports search over labeled data, Graph^- that supports incoming neighbor queries over graph-structured data, and Graph^+ that supports outgoing neighbor queries over graph-structured data. We stress that Label must be associative. Given a web graph $(G, L, \mathbf{m})$ we encrypt $(G, \mathbf{m})$ using both Graph^+ and Graph^-, resulting in ciphertexts $\mathbf{c}^+$ and $\mathbf{c}^-$. Now, for each node i in G, we generate a pair of tokens (τ_i^+, τ_i^-). We then use Label to encrypt $(L, \mathbf{m})$ using the token pairs (τ_i^+, τ_i^-) as semi-private data (recall that Label is associative). We then output the encryption $\mathbf{c}^{\mathsf{l}}$ of $(L, \mathbf{m})$.

A focused subgraph query on keyword w is handled as follows. A token $\tau^{\mathsf{l}} \leftarrow \mathsf{Label}.\mathsf{Token}_K(w)$ is generated and sent to the server. When used with the ciphertext $\mathbf{c}^{\mathsf{l}}$, this token will reveal to the

[8] If we wish to perform full-text search then the labeling can simply assign a page to all of its words.

server (1) pointers to all the (encrypted) web pages labeled with keyword w; and (2) for each of these encrypted pages c_j, the semi-private information which consists of tokens (τ_j^+, τ_j^-). For each encrypted page, the server can then use the token pairs with ciphertexts $\mathbf{c}_j^+$ and $\mathbf{c}_j^-$ to recover pointers to any incoming and outgoing neighbors of page c_j.

For the labeling L, let $|L|$ be the number of words w such that $L(w)$ is nonempty, and let $\max(L)$ be the size of the largest set $L(w)$.

Let $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ be an elusive and associative encryption scheme for labeled data, and $\mathsf{Graph}^+ = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ and $\mathsf{Graph}^- = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ be elusive graph encryption schemes that support neighbor queries. Our scheme $\mathsf{LabGraph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Subgraph}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate three keys $K_1 \leftarrow \mathsf{Graph}^+.\mathsf{Gen}(1^k)$, $K_2 \leftarrow \mathsf{Graph}^-.\mathsf{Gen}(1^k)$ and $K_3 \leftarrow \mathsf{Label}.\mathsf{Gen}(1^k)$. Let $K = (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, G, \mathbf{m})$:
  1. compute $(\gamma^+, \mathbf{c}^+) \leftarrow \mathsf{Graph}^+.\mathsf{Enc}_{K_1}(G, \mathbf{m})$,
  2. compute $(\gamma^-, \mathbf{c}^-) \leftarrow \mathsf{Graph}^-.\mathsf{Enc}_{K_2}(G, \mathbf{m})$,
  3. for $1 \leq i \leq n$,
     (a) compute $\tau_i^+ \leftarrow \mathsf{Graph}^+.\mathsf{Token}_{K_1}(i)$,
     (b) compute $\tau_i^- \leftarrow \mathsf{Graph}^-.\mathsf{Token}_{K_2}(i)$,
  4. let L be the labeling generated from all the words in $\mathbf{m}$ (i.e., each m_i is labeled with the words it contains) and let $\mathbf{v} = (\tau_i^+, \tau_i^-)_i$,
  5. compute $(\gamma^\mathsf{l}, \mathbf{c}^\mathsf{l}) \leftarrow \mathsf{Label}.\mathsf{Enc}_{K_3}(L, \mathbf{M})$, where $\mathbf{M}$ is composed of $\mathbf{m}$ and $\mathbf{v}$,
  6. output $\gamma = (\gamma^+, \gamma^-, \gamma^\mathsf{l})$ and $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-, \mathbf{c}^\mathsf{l})$.
- $\mathsf{Token}(K, w)$: output $\tau \leftarrow \mathsf{Label}.\mathsf{Token}_{K_3}(w)$.
- $\mathsf{Subgraph}_\mathsf{e}(\gamma, \tau)$:
  1. compute $(J^\mathsf{l}, \mathbf{v}_I) := \mathsf{Label}.\mathsf{Search}(\gamma^\mathsf{l}, \tau)$
  2. parse $\mathbf{v}_I$ as $(\tau_j^+, \tau_j^-)_j$
  3. for all $j \in J^\mathsf{l}$,
     (a) compute $J_j^+ := \mathsf{Graph}^+.\mathsf{Neigh}(\gamma^+, \tau_j^+)$,
     (b) compute $J_j^- := \mathsf{Graph}^-.\mathsf{Neigh}(\gamma^-, \tau_j^-)$,
  4. output $J = \left(J^\mathsf{l}, \left(J_j^+, J_j^-\right)_{j \in J^\mathsf{l}}\right)$.
- $\mathsf{Dec}(K, c_j)$:
  1. compute $m_j^\mathsf{l} := \mathsf{Label}.\mathsf{Dec}_{K_1}(c_j)$, $m_j^+ := \mathsf{Graph}^+.\mathsf{Dec}_{K_2}(c_j)$, and $m_j^- := \mathsf{Graph}^-.\mathsf{Dec}_{K_3}(c_j)$
  2. output the message m in $\{m_j^\mathsf{l}, m_j^+, m_j^-\}$ such that $m \neq \perp$.

Figure 5: A structured encryption scheme for web graphs supporting focused subgraph queries.

Theorem 6.2. *If* Label *is an elusive and chainable structured encryption scheme that is* $(\mathcal{L}_1^l, \mathcal{L}_2^l)$*-secure against adaptive chosen query attacks, and if* Graph^+ *and* Graph^- *are elusive and chainable schemes that are respectively* $(\mathcal{L}_1^+, \mathcal{L}_2^+)$*-secure and* $(\mathcal{L}_1^-, \mathcal{L}_2^-)$*-secure against adaptive chosen query attacks, then*

the scheme described above is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, where

$$\mathcal{L}_1(G, L, \mathbf{m}) = \Big(\mathcal{L}_1^{\mathsf{I}}\big(L, \mathbf{m}\big), \mathcal{L}_1^{+}(G, \mathbf{m}), \mathcal{L}_1^{-}(G, \mathbf{m}) \Big),$$

and

$$\mathcal{L}_2(G, L, w) = \Big(\mathcal{L}_2^{\mathsf{I}}(L, w), \big(\mathcal{L}_2^{+}(G, i)\big)_{i \in R(w)}, \big(\mathcal{L}_2^{-}(G, i)\big)_{i \in R(w)} \Big).$$

If Label, Graph^{+} and Graph^{-} are instantiated with the schemes in Sections 5.2 and 5.3, then $\mathcal{L}_1(G, L, \mathbf{m}) = (n, |L|, max(L), n, \ell)$ and

$$\mathcal{L}_2(G, L, w) = \Big(|R(w)|, \textsc{qp}(w), \textsc{ip}(w), \big(\deg^{+}(i), \textsc{qp}^{+}(i), \textsc{ip}^{+}(i)\big)_{i \in R(w)}, \big(\deg^{-}(i), \textsc{qp}^{-}(i), \textsc{ip}^{-}(i)\big)_{i \in R(w)} \Big),$$

where $\textsc{qp}^{+}(i)$ and $\textsc{qp}^{-}(i)$ reveal whether the ith node of $R(w)$ is in the root graph of any previous query w and $\textsc{ip}^{+}(i)$ and $\textsc{ip}^{-}(i)$ reveal whether any of i's outgoing and incoming neighbors were also neighbors of some node in the root graphs of the current and previous queries.

Proof. Let $\mathcal{S}_{\mathsf{I}}$, $\mathcal{S}_{+}$ and $\mathcal{S}_{-}$ be the simulators guaranteed to exist by the CQA2-security of Label, Graph^{+} and Graph^{-}, respectively. Consider the simulator $\mathcal{S}$ that proceeds as follows:

1. given $\mathcal{L}_1(G, L, \mathbf{m}) = (\mathcal{L}_1^{+}, \mathcal{L}_1^{-}, \mathcal{L}_1^{\mathsf{I}})$ it outputs $\gamma = (\gamma^{+}, \gamma^{-}, \gamma^{\mathsf{I}})$ and $\mathbf{c} = (\mathbf{c}^{+}, \mathbf{c}^{-}, \mathbf{c}^{\mathsf{I}})$ such that:
   (a) $(\gamma^{+}, \mathbf{c}^{+}) \leftarrow \mathcal{S}_{+}(\mathcal{L}_1^{+})$,
   (b) $(\gamma^{-}, \mathbf{c}^{-}) \leftarrow \mathcal{S}_{-}(\mathcal{L}_1^{-})$,
   (c) and $(\gamma^{\mathsf{I}}, \mathbf{c}^{\mathsf{I}}) \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_1^{\mathsf{I}})$.
2. for each keyword w, given $\mathcal{L}_2(G, L, w) = \big(\mathcal{L}_2^{\mathsf{I}}, \mathcal{L}_{2,1}^{+}, \ldots, \mathcal{L}_{2,r}^{+}, \mathcal{L}_{2,1}^{-}, \ldots, \mathcal{L}_{2,r}^{-}\big)$, it:
   (a) computes $\tau_j^{+} \leftarrow \mathcal{S}_{+}(\mathcal{L}_{2,j}^{+})$ and $\tau_j^{-} \leftarrow \mathcal{S}_{-}(\mathcal{L}_{2,j}^{-})$ for all $j \in [r]$
   (b) and outputs $\tau_w \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_2^{\mathsf{I}}, \mathbf{v}_w)$, where $\mathbf{v}_w = (\tau_j^{+}, \tau_j^{-})_{j \in [r]}$.

We now show that for all PPT adversaries $\mathcal{A}$ the $\mathbf{Real}(k)$ and $\mathbf{Ideal}(k)$ experiments of Definition 4.2 will output 1 with negligibly-close probabilities. We show this by considering the following sequence of games:

Game_0: this game corresponds to an execution of the $\mathbf{Real}(k)$ experiment. Recall that (1) the challenger generates a key $K = (K_1, K_2, K_3)$; (2) the adversary outputs data $(G, L, \mathbf{m})$ and receives $\gamma = (\gamma^{+}, \gamma^{-}, \gamma^{\mathsf{I}})$ and $\mathbf{c} = (\mathbf{c}^{+}, \mathbf{c}^{-}, \mathbf{c}^{\mathsf{I}})$ where $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}_K(G, L, \mathbf{m})$; and (3) the adversary adaptively outputs queries, and for each query w it receives a token $\tau_w \leftarrow \mathsf{Label.Token}_{K_3}(w)$. Finally, $\mathcal{A}$ returns a bit b that is output by the game.

Game_1: is the same as Game_0 except that in step 5 of the encryption algorithm, γ^{I} and $\mathbf{c}^{\mathsf{I}}$ are replaced with the output of $\mathcal{S}_{\mathsf{I}}(\mathcal{L}_1^{\mathsf{I}}(L, \mathbf{m}))$ and the tokens τ_w for each keyword query w are computed as follows. Let $R(w)$ be w's root graph and set $\mathbf{v}_w := (\tau_i^{+}, \tau_i^{-})_{i \in R(w)}$, where the (τ_i^{+}, τ_i^{-}) are from the set of tokens generated in step 3 of the encryption algorithm. Finally, compute $\tau_w \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_2^{\mathsf{I}}(L, w), \mathbf{v}_w)$.

Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference between the probabilities that Game_0 and Game_1 output 1 is non-negligible. We then show that there exists a PPT adversary $\mathcal{B}$ that breaks the CQA2-security of Label.

$\mathcal{B}$ generates keys $K_1 \leftarrow \mathsf{Graph}^+.\mathsf{Gen}(1^k)$ and $K_2 \leftarrow \mathsf{Graph}^-.\mathsf{Gen}(1^k)$ and simulates $\mathcal{A}$. Upon receiving $(G, L, \mathbf{m})$ from $\mathcal{A}$ it proceeds as follows. For $1 \leq i \leq n$, it computes $\tau_i^+ \leftarrow \mathsf{Graph}^+.\mathsf{Token}(K_1, i)$ and $\tau_i^- \leftarrow \mathsf{Graph}^-.\mathsf{Token}(K_2, i)$. It then outputs $(L, \mathbf{M})$, where $\mathbf{M} = (\mathbf{m}, \mathbf{v})$ and $\mathbf{v} = (\tau_i^+, \tau_i^-)_{i \leq n}$. Upon receiving $(\gamma^{\mathsf{I}}, \mathbf{c}^{\mathsf{I}})$ from its experiment, it sends $\gamma = (\gamma^+, \gamma^-, \gamma^{\mathsf{I}})$ and $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-, \mathbf{c}^{\mathsf{I}})$ to $\mathcal{A}$, where

1. $(\gamma^+, \mathbf{c}^+) \leftarrow \mathsf{Graph}^+.\mathsf{Enc}(K_1, G, \mathbf{m})$,
2. $(\gamma^-, \mathbf{c}^-) \leftarrow \mathsf{Graph}^-.\mathsf{Enc}(K_2, G, \mathbf{m})$.

$\mathcal{B}$ answers each query w of $\mathcal{A}$ by sending the query w to its oracle and returning the token τ_w it receives. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Note that if $\mathcal{B}$ is running in a $\mathbf{Real}(k)$ experiment, then $\mathcal{A}$'s view is the same as its view in Game_0. On the other hand, if $\mathcal{B}$ is running in an $\mathbf{Ideal}(k)$ experiment, then $\mathcal{A}$'s view is the same as its view in Game_1. Sine $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs, by our original assumption, the difference in the probabilities that the two experiments output 1 is non-negligible.

Game_2: this is the same as Game_1 with the exception that we do not compute the τ_i^+'s and τ_i^-'s until they are needed. More precisely, step 3 of the encryption algorithm is omitted and, for each keyword w, the token τ_w is computed as follows:

1. for all $i \in R(w)$, set $\tau_i^+ \leftarrow \mathsf{Graph}^+.\mathsf{Token}(K_1, i)$,
2. for all $i \in R(w)$, set $\tau_i^- \leftarrow \mathsf{Graph}^-.\mathsf{Token}(K_2, i)$,
3. compute $\tau_w \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_2^{\mathsf{I}}(L, w), \mathbf{v}_w)$, where $\mathbf{v}_w := (\tau_i^+, \tau_i^-)_{i \in R(w)}$.

Since $\mathsf{Graph}.\mathsf{Token}$ is stateless, this is identical to Game_1.

Game_3: this is the same as Game_2 with the exception that γ^+ and $\mathbf{c}^+$ are replaced with the output of $\mathcal{S}_+(\mathcal{L}_1^+(G, \mathbf{m}))$ and each τ_i^+ that is computed is replaced with the output of $\mathcal{S}_+(\mathcal{L}_2^+(G, i))$.

Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference between the probabilities that Game_2 and Game_3 output 1 is non-negligible. We then show that there exists a PPT adversary $\mathcal{B}$ that breaks the CQA2-security of Graph^+.

$\mathcal{B}$ generates key $K_2 \leftarrow \mathsf{Graph}^-.\mathsf{Gen}(1^k)$ and simulates $\mathcal{A}$. Upon receiving $(G, L, \mathbf{m})$ from $\mathcal{A}$, it outputs $(G, \mathbf{m})$. Upon receiving $(\gamma^+, \mathbf{c}^+)$ from its experiment, $\mathcal{B}$ sends $\gamma = (\gamma^+, \gamma^-, \gamma^{\mathsf{I}})$ and $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-, \mathbf{c}^{\mathsf{I}})$ to $\mathcal{A}$, where:

1. $(\gamma^-, \mathbf{c}^-) \leftarrow \mathsf{Graph}^-.\mathsf{Enc}(K_2, G, \mathbf{m})$,
2. $(\gamma^{\mathsf{I}}, \mathbf{c}^{\mathsf{I}}) \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_1^{\mathsf{I}}(L, \mathbf{m}))$.

$\mathcal{B}$ answers each query w of $\mathcal{A}$ as follows:

1. for all $i \in R(w)$, it outputs i and receives τ_i^+,
2. for all $i \in R(w)$, it computes $\tau_i^- \leftarrow, \mathsf{Graph}^-.\mathsf{Token}(K_2, i)$,
3. it sends to $\mathcal{A}$ the token $\tau_w \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_2(L, w), \mathbf{v}_w)$, where $\mathbf{v}_w := (\tau_i^+, \tau_i^-)_{i \in R(w)}$.

Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

As above, it follows from our assumption on $\mathcal{A}$ that $\mathcal{B}$ will break the CQA2-security of Graph^+.

Game_4: this is the same as Game_3 with the exception that γ^- and $\mathbf{c}^-$ are replaced with the output of $\mathcal{S}_-(\mathcal{L}_1^-(G,\mathbf{m}))$ and each τ_i^- that is computed is replaced with the output of $\mathcal{S}_-(\mathcal{L}_2^-(G,i))$.

Suppose there exists a PPT adversary $\mathcal{A}$ such that the difference between the probabilities that Game_3 and Game_4 output 1 is non-negligible. We then show that there exists a PPT adversary $\mathcal{B}$ that breaks the CQA2-security of Graph^-.

$\mathcal{B}$ simulates $\mathcal{A}$. Upon receiving $(G, L, \mathbf{m})$ from $\mathcal{A}$, it outputs $(G, \mathbf{m})$. Upon receiving $(\gamma^-, \mathbf{c}^-)$ from its experiment, $\mathcal{B}$ sends $\gamma = (\gamma^+, \gamma^-, \gamma^{\mathsf{I}})$ and $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-, \mathbf{c}^{\mathsf{I}})$ to $\mathcal{A}$, where:

1. $(\gamma^+, \mathbf{c}^+) \leftarrow \mathcal{S}_+(\mathcal{L}_1^+(G,\mathbf{m}))$
2. $(\gamma^{\mathsf{I}}, \mathbf{c}^{\mathsf{I}}) \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_1^{\mathsf{I}}(L,\mathbf{m}))$.

$\mathcal{B}$ answers each query w of $\mathcal{A}$ as follows:

1. for all $i \in R(w)$, it computes $\tau_i^+ \leftarrow \mathcal{S}_+(\mathcal{L}_2^+(G,\mathbf{m}))$,
2. for all $i \in R(w)$, it outputs i and receives τ_i^-,
3. it sends to $\mathcal{A}$ the token $\tau_w \leftarrow \mathcal{S}_{\mathsf{I}}(\mathcal{L}_2^{\mathsf{I}}(L,w), \mathbf{v}_w)$, where $\mathbf{v}_w := (\tau_i^+, \tau_i^-)_{i \in R(w)}$.

Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

As above, it follows from our assumption on $\mathcal{A}$ that $\mathcal{B}$ will break the CQA2-security of Graph^-.

Game_5: is the same as Game_4 except that $\mathcal{L}_1^{\mathsf{I}}(L,\mathbf{M})$, $\mathcal{L}_1^+(G,\mathbf{m})$, $\mathcal{L}_2^-(G,\mathbf{m})$ and, for each keyword w, $\mathcal{L}_2^{\mathsf{I}}(L,w)$ and $\big(\mathcal{L}_2^+(G,i), \mathcal{L}_2^-(G,i)\big)_{i \in R(w)}$ are provided by an oracle as opposed to being computed from G and w. Observe that this game corresponds to an execution of the $\mathbf{Ideal}(k)$ experiment with simulator $\mathcal{S}$.

Clearly, the outputs of Game_4 and Game_5 are identically distributed.

□

7 Conclusions and Future Directions

Searchable encryption is an important problem that is well motivated by the recent trend towards cloud storage. Much of the data generated today, however, is not simple text data but exhibits a much richer structure. In this work we introduced the notion of structured encryption, which generalizes searchable encryption to data that exhibits an arbitrary structure. We showed how to perform private queries on a variety of data types including matrices, graphs and web graphs. In addition, we showed that structured encryption has applications beyond performing private queries on encrypted data and proposed the novel application of controlled disclosure.

Several interesting future directions are suggested by this work. The most immediate is whether efficient and non-interactive structured encryption can be achieved while leaking less than the query and intersection pattern. The construction of efficient *dynamic* structured encryption schemes (i.e., that allow for updates to the encrypted data) is another direction left open by this work. Of course, the construction of schemes that handle other types of structured data and more complex queries on the data types considered here would also be interesting.

Acknowledgements

We are grateful to Kristin Lauter for encouragement during the early stages of this work, to Sherman Chow and Satya Lokam for useful discussions regarding graph encryption and to Susan Hohenberger for insisting on a thorough comparison with functional encryption. We are also grateful to Adam O'Neill for several helpful discussions on functional encryption and to Charalampos Papamanthou for suggesting the simple approach in Section 6. Finally, we thank Emily Shen for useful feedback on the manuscript and the anonymous reviewers for helpful suggestions.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. M. Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In V. Shoup, editor, *Advances in Cryptology – CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] J. Bardin, J. Callas, S. Chaput, P. Fusco, F. Gilbert, C. Hoff, D. Hurst, S. Kumaraswamy, L. Lynch, S. Matsumoto, B. O'Higgins, J. Pawluk, G. Reese, J. Reich, J. Ritter, J. Spivey, and J. Viega. Security guidance for critical areas of focus in cloud computing. Technical report, Cloud Security Alliance, April 2009.

[3] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *Advances in Cryptology – CRYPTO '07*, Lecture Notes in Computer Science, pages 535–552. Springer, 2007.

[4] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[5] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[6] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[7] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith. Public-key encryption that allows PIR queries. In A. Menezes, editor, *Advances in Cryptology – CRYPTO '07*, volume 4622 of *Lecture Notes in Computer Science*, pages 50–67. Springer, 2007.

[8] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. Technical Report 2010/543, IACR ePrint Cryptography Archive, 2010. See `http://eprint.iacr.org/2010/543`.

[9] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference (TCC '07)*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[10] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[11] M. Brautbar and M. Kearns. Local algorithms for finding intersting individuals in large networks. In *Innovations in Computer Science (ICS '10)*, 2010.

[12] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security (ACNS '05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer, 2005.

[13] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *ACM symposium on Theory of computing (STOC '88)*, pages 11–19. ACM, 1988.

[14] Micrsoft Corp. Windows azure marketplace. http://www.microsoft.com/windowsazure/marketplace/.

[15] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *ACM Conference on Computer and Communications Security (CCS '06)*, pages 79–88. ACM, 2006.

[16] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. Journal version (under submission), 2010.

[17] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC '09)*, pages 169–178. ACM Press, 2009.

[18] E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003. See `http://eprint.iacr.org/2003/216`.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *ACM Symposium on the Theory of Computation (STOC '87)*, pages 218–229. ACM, 1987.

[20] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM conference on Computer and communications security (CCS '06)*, pages 89–98, New York, NY, USA, 2006. ACM.

[23] Infochimps. http://www.infochimps.org.

[24] S. Kamara and K. Lauter. Cryptographic cloud storage. In *Workshop on on Real-Life Cryptographic Protocols and Standardization*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer, 2010.

[25] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.

[26] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[27] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Symposium on Discrete Algorithms (SODA '08)*, pages 668–677. Society for Industrial and Applied Mathematics, 1998.

[28] R. Lempel and S. Moran. SALSA: The stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 19(2):131–160, April 2001.

[29] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[30] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[31] A. Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985.

[32] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Theory of Cryptography Conference (TCC '09)*, pages 457–473, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, Washington, DC, USA, 2007. IEEE Computer Society.

[34] C. Soghoian. Caught in the cloud: Privacy, encryption, and government back doors in the web 2.0 era. *Journal on Telecommunications and High Technology Law*, 8(2), 2010.

[35] D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Research in Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[36] B. Waters, D. Balfanz, G. Durfee, and D. Smetters. Building an encrypted and searchable audit log. In *Network and Distributed System Security Symposium (NDSS '04)*. The Internet Society, 2004.

[37] A. Yao. Protocols for secure computations. In *IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE Computer Society, 1982.

A Comparison with Functional Encryption

Functional encryption [8] is a recent paradigm which generalizes a variety of primitives including identity based encryption [31, 6], attribute based encryption [30, 22, 4], and predicate encryption [26, 32]. The idea is that a user should be able to encrypt messages and, independently, generate tokens for various functions which allow an untrusted party to evaluate those function on each of the encrypted messages without learning any additional information. (In some cases it is also required that the function itself be hidden [32].) One application which is often cited is that of a mail server, which is not trusted to read messages, but which is required to check for certain properties of each message in order to determine how to handle it. (E.g. the flag "urgent" might require a special response.) Using functional encryption, the users would generate a token for whatever functions the server is expected to compute, and send these tokens to the server. Then as the server receives each piece of encrypted mail, it would apply the tokens to the ciphertext to evaluate the given functions and determine how to handle the message. Thus, the key points are that tokens and ciphertexts should be able to be generated independently, and that each token should produce the correct result when applied to any ciphertext.

A.1 Defining security for functional encryption

All of the above works consider an indistinguishability based definition which says that an adversary cannot distinguish encryptions of two different messages unless he requests a token for a function that would trivially allow him to distinguish between them, i.e. a function f such that $f(m_0) \neq f(m_1)$.

Another standard way of defining a secrecy property is by the existence of a simulator who is given only the information which the scheme is intended to reveal, and must simulate a convincing interaction. (This is the approach taken in Section 4.) In traditional encryption, it has been shown that the natural indistinguishability and simulation based definitions are equivalent [21]. However, in many cases this equivalence does not hold, and it is not clear whether an indistinguishability definition for functional encryption could be equivalent to a natural simulation based one. Furthermore, it is often much easier to verify that a simulation based definition captures the desired intuitive notion of security. Finally, in many cases a simulation based definition can make arguing about composition simpler, so that it is easier to use a primitive as part of a larger protocol. For all of these reasons we choose to focus on a simulation based definition of security in this paper.

A.2 Implication of lower bound

When we consider functional encryption in the context of the lower bound described in Remark 1 in Section 4, we see that there may have been a reason that it is traditionally defined using an indistinguishability approach. In particular, consider a functional encryption scheme as described above, which allows a user to encrypt arbitrarily many different messages m_i to generate many ciphertexts c_i. A given token should produce a correct value on all of these ciphertexts. Now, suppose we want to consider a simulation based definition. Let S be the set of values that can result from applying f to any message in the message space. If ℓ ciphertexts have been generated, then the token for f applied to ciphertexts $c_1, \ldots, c_\ell$ should produce the appropriate values $f(m_1), \ldots, f(m_\ell) \in S^\ell$. In the notation used in Remark 1, the answer space A_f for query f will be S^ℓ, which means that in order to satisfy a simulation based definition the scheme must generate tokens which are at least $\log(|S|^\ell) = \ell \log(|S|)$ bits long.

In other words, if we want to achieve a simulation based definition without relying on the random oracle model, our tokens must have length proportional to the number of ciphertexts encrypted. This leads to a couple of conclusions:

- If we want simulation based security in a functional encryption scheme, the length of each token must be proportional to the maximum number of ciphertexts that will be encrypted.
- If we want to be able to generate an unlimited number of ciphertexts, simulation based security is impossible. (To see why, recall that a token τ must work for all ciphertexts, including those which are generated later. If after token τ has been generated, we encrypt an additional $|\tau| + 1$ messages, then we break the lower bound.)

A.3 Comparison with Structured Encryption

Here we are focused on a particular setting, where a client wants to encrypt some large volume of data and store it on a server, while still taking advantage of any underlying structure. For example, we might want to store a large number of encrypted files on a server and still efficiently retrieve desired files in response to a given query (e.g. search for all files containing a given keyword). There are two different ways one might enable this with a functional encryption scheme:

Encrypting structured data using many ciphertexts. This has been the standard approach proposed for using functional encryption to allow searching over encrypted data. For example, one might allow keyword search queries by encrypting the keywords for each document using a functional encryption scheme and storing the resulting ciphertexts together with an encryption (under a standard encryption scheme) of the entire document. Then to allow a search for a given keyword w, the client would simply generate the token for the function f_w which outputs 1 on input w, and 0 on any other input.

This approach has some nice features, in particular that it easily supports updates to the set of documents. However, it also has several inherent limitations. First, it limits the efficiency, in that it will necessarily require linear time to compute a query on any set of files, because the server must apply the token to each ciphertext. In some cases this may be unavoidable, but in others (e.g. the keyword search application above or the applications described in Sections 5 and 6), we can do better using other approaches [15].

It also restricts the set of queries which can be answered to those which can be evaluated locally, i.e. by evaluating some function on each message individually. Thus, it may rule out functions which capture some global properties of the data. (For example, it is not obvious how to use such an approach to achieve the functionality described in Section 6.)

Finally, because of the lower bound presented in the previous Section, this functional encryption approach will either be restricted to an indistinguishability based notion of security, or it will require tokens whose length grows linearly in the number of individual ciphertexts generated. (E.g. in the keyword search application, the token would have to be of size linear in the sum of the number of keywords in each document.) In contrast, a structured encryption approach like that in [12, 15] or the scheme described in Section 5.2 can use tokens whose size depends only on the maximum number of documents which can contain a given keyword (and the security parameter).

Encrypting structured data using a single ciphertext. A different approach would be to consider the entire data structure to be a single message and to consider the allowable functions to be the supported queries. This is essentially approach we take. The main different would be that in our case, we restrict the encryption further and say that tokens need only work for a single instance of the data structure (i.e. a single ciphertext). This seems to be a fairly natural condition, and specifying it explicitly allows us to construct efficient schemes for some interesting and practical functions. (See Sections 5 and 6.) It also, as noted above, allows us to realize the stronger and perhaps more natural simulation-based security definition without extraordinarily long tokens.