ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Design and Analysis of Graph Encryption Schemes

Master's Thesis

Anselme Goetschmann

September 23, 2020

Advisors: Prof. Dr. Kenny Paterson, Dr. Sikhar Patranabis

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

With the advent of cloud storage and computing, the related privacy issues have attracted the attention of researchers. This has resulted in the development of encrypted databases which can be queried without the server learning the content of the database or of the queries. Although provably secure by some definitions, such schemes rely on a trade-off between security and efficiency and thus leak a controlled amount of information to the server. The analysis of the various types of leakages has evolved in parallel to the development of new schemes and has revealed important potential attacks.

In this Master's thesis we focus on encrypted databases designed for data in the form of graphs such as social networks or web graphs. Since the leakage resulting from encrypted graph databases has not yet been subject to thorough anaylsis we explore this topic. More specifically, we consider schemes running shortest path queries on graphs and in particular encrypted sketch-based distance oracles like the GRECS scheme introduced by Meng et al. in *ACM CCS* 2015. To our knowledge, the sketch pattern leakage occuring in this type of construction has not yet been analysed.

Based on two different algorithms to build distance sketches, we investigate how much information the sketch pattern reveals about the sketches themselves. Using our observations, we explore potential attacks which could be mounted by a malicious server. In particular, we propose a query recovery attack feasible by an attacker with partial database knowledge. Our results show that the sketch pattern leakage can reveal sensitive information to the server and we propose a modification to the sketching algorithm to mitigate the impact of the sketch pattern leakage on privacy.

Contents

Chapter 1

Introduction

With the advent of cloud storage and computing, many companies outsource their data to a third party. The cloud providers do not only store and manage their customers' data but also allow them to perform queries on the data and run complex tasks on their infrastructure. This simplifies the access to large storage space and high computing power for smaller companies and allows them to focus on their core activity.

Since a data leak usually has devastating effects on their business [JA20], the companies have to put a lot of trust in their cloud provider. In the meantime the complexity of the systems lead to weaknesses and breaches are frequent [EHF16].

The privacy concerns arising with cloud databases and the possible solutions to provide some security guarantees have been an active area of research. Currently, the main challenge is to encrypt the data in order to store it on an untrusted server without losing the ability to work with it efficiently. For example, a user of a Facebook-like social network may want to search for the friends of a friend, which is equivalent to a neighouring query on a graph, without the server gaining knowledge about the social network.

1.1 Overview and Motivation

Ideally, encrypted databases allow a client to send data to a server and query it without the server learning anything about the data or the queries. Encrypting the data with a block cipher before sending it to the server is not a viable solution since the client has to download the full data for each query, which is not feasible with the size of most databases, or share the decryption key to let the server execute the query, which defeats confidentiality purposes.

Some schemes such as fully homomorphic encryption [GO96] or oblivious

RAM [Gen09] achieve this goal but at a high computational cost or communication overhead, which makes them impractical in their current state. In order to design usable schemes, a compromise between security and efficiency has to be made. In practice, the security guarantees are weakened by leaking a controlled amount of data to the server.

Many schemes allowing to search over encrypted data with a determined leakage have been developed in recent years [G+03, CGKO11, KPR12, CJJ+14]. Numerous attacks exploiting the different leakage profiles have naturally followed [IKK12, CGPR15, KKNO16] and researchers have been trying to optimise the trade-off between usability and privacy [SPS13, LZWT14].

The aforementioned schemes handle either unstructured data in the form of a collection of documents or data organised in the relational model, with a fixed structure. However, some types of data such as social networks or web graphs require more flexibility. Unlike relational databases, graph databases treat relationships like values and are optimised for operations such as neighboring queries or path traversals in a graph. Implementations based on this principle and used in the industry include graph databases such as Neo4j[1], GraphBase[2] or OrientDB[3].

Some encrypted graph databases which can be queried with privacy guarantees have been designed [CK10, MKNK15], but each type of graph query requires a specific construction and the current schemes remain very limited. In this thesis, we will focus on shortest path queries and, more specifically, on the analysis of the leakage resulting from sketch-based encrypted graph database schemes.

1.2 Our Contributions

We present the analysis of the leakage resulting from the graph encryption scheme by Meng et al. [MKNK15]. Their **gr**aph **e**n**c**ryption for **s**hortest distance queries (GRECS) construction relies on a sketch-based distance oracle to solve approximate shortest path queries. A sketch is a data structure for a vertex of a graph which contains the distance to a subset of other selected vertices, named seeds. The shortest path length between two vertices is then approximated by taking the minimum sum among the overlapping seeds between their sketches.

In the GRECS scheme the sketches are encrypted, but the server nevertheless learns a hidden version of the seed set of a sketch when its vertex is queried. Hence, the server can deduce the pattern of common seeds across sketches

[1] https://neo4j.com/
[2] https://graphbase.ai/
[3] https://orientdb.org/

and count how often an encrypted seed appears. This leakage is referred to as *sketch pattern* and it has, to our knowledge, not been analysed yet.

Distance estimation from the sketch pattern leakage. Using the sketch pattern leakage, we develop a technique to approximate the content of sketches. Indeed, the sketch for a vertex is generated using an algorithm which selects seeds to ensure a good performance of the distance oracle. The number of sketches in which a seed appears thus reveals roughly its average distance in the sketches. We formulate a distance estimate and evaluate how well it performs on the sketches of self-generated and real-life graphs.

Leakage-based attacks on GRECS. Based on our distance estimate, we investigate potential attacks which could expose private information to the server. Our main result concerns the recovery of queries by an attacker having some known queries and partial database knowledge. The known queries allow the attacker to form vectors of distances to seeds for vertices which can be queried. A similar distance estimate vector can be constructed for a newly queried vertex using the sketch pattern leakage. The attacker can then compare the distance estimate vector with the known distance vectors and deduce which vertex was queried.

Evaluation of our attacks. In order to evaluate our attacks, we perform experiments using our implementation[4] of the algorithms to generate sketches and some of the datasets used by Meng et al. [MKNK15]. Combined with these experiments, our novel analysis confirms that the sketch pattern leakage can lead to serious privacy breaches and should not be overlooked.

Possible countermesure. Lastly, we consider a possible countermeasure modifiyng the underlying distance oracle to make the sketch pattern leakage less important. Fake seeds are inserted in the sketches to jam the frequency of seeds among sketches. Our experiments show that the countermeasure efficiently hides the sketch pattern at the cost of larger sketches.

1.3 Outline of the Thesis

In the rest of this chapter we introduce the necessary notation. We review the existing encrypted database schemes and the attacks exploiting their various leakage profiles in Chapter 2. We also explain in details the GRECS scheme. In Chapter 3 we investigate one of the leakage types from the GRECS construction, namely the sketch pattern. In Chapter 4 we use the

[4]Part of the code is available on https://github.com/agoetschm/master-thesis

results from Chapter 3 to explore possible attacks and propose a countermesure to lower the impact of the sketch pattern leakage. Chapter 5 concludes the thesis and proposes future work.

1.4 Notation

In this section we introduce the notation and conventions adopted in the following chapters. Graphs are by default finite, undirected and unweighted. For a graph $G = (V, E)$, V is the set of vertices, E the set of edges and $n = |V|$ the number of vertices. The average degree $\bar{k}$ of a graph G is the average of the vertex degree over the set of vertices V. The random graphs $G_{n,p}$ we use are based on the Erdős-Rényi model: they have n vertices and each of the $\binom{n}{2}$ pairs of vertices are linked by an edge with probability p.

We use the symbol $\|$ to denote concatenation. Square brackets [] are used either for access to an element in an array, or as short notation for the set $[N] = \{1, ..., N\}$ where N is a positive integer. Angle brackets are used for tuples $\langle v_1, v_2, ... \rangle$ in which each element v_i has a fixed size. The symbol $\widehat{\ }$ denotes an estimate, e.g. the function $\widehat{f}$ approximates f. The arrow in $a \leftarrow b$ stands for an assignment of the value of b to variable a and $x \xleftarrow{\$} \mathcal{X}$ means that x is sampled uniformly at random from the set $\mathcal{X}$.

Chapter 2

Background and Related Work

The purpose of this chapter is to introduce the graph encryption scheme we attempt to attack in the next chapters, namely the GRECS scheme from Meng et al. which runs approximate shortest path queries on an encrypted graph [MKNK15]. In order to do so, we first review the different types of encrypted databases, existing constructions and attacks on them.

We start by giving an overview of the existing techniques to encrypt relational database, among which we find various types of *searchable encryption*. After describing the different classes of leakages occuring in those encryption schemes we go over some leakage-abuse attacks. Next, we move to schemes optimised for handling graph operations, a less popular research area we want to explore further. Finally, we take a closer look at sketch-based distance oracles solving approximate shorest path queries on a graph. The GRECS scheme is based on the latter and introduced in the last section.

2.1 Relational Databases

Databases storing a large amount of data are usually running on a dedicated system, hence in most settings we have one or more clients querying a database on a server. With the advent of cloud storage and computing, the server tends to be managed by a separate entity and the owner of the data might be concerned with privacy issues if they do not fully trust the server. Since encrypting the data before sending to the server would also mean downloading it for every read or write operation, researchers have been actively looking for a more efficient solution.

2.1.1 Schemes

ORAM, FHE and FE. In our context, the goal of encrypted databases is to be able to store data on a server and query it without the server being able

to learn anything about the data or the queries. Some techniques such as *oblivious RAM* (ORAM) [GO96], *fully homomorquic encryption* (FHE) [Gen09] or *functional encryption* (FE) [BSW11] satisfy these requirements, but they come at a high cost. The large number of communication rounds or the high computing power requirements make these techniques not usable in practice.

One way to make schemes more efficient is to relax their security requirements. If we allow the server to learn some controlled amount of data, namely some *leakage*, we can achieve encryption schemes with a lower communication and computation overhead, making them usable.

SE. One of the first type of schemes going in this direction is called *searchable encryption* (SE) and enables keyword search queries over an encrypted collection of documents. The scheme described by Song et al. [SWP00] uses symmetric encryption and a set of keys to manage the access control. The main issue with their scheme is that it requires a search time of $\mathcal{O}(n)$ which is not good enough on a large amount of data.

Secure index. To address this issue a precomputed *secure index* can be used [G+03]. The idea is to prepare an encrypted data structure storing the documents where a keyword appears and which can be queried efficiently for a keyword.

SSE. Curtmola et al. formalised the concept and propose constructions satisfying their definitions [CGKO11]. In particular, we give a simplified overview of their definition of *searchable symmetric encryption* (SSE), illustrated in Figure 2.1. An encryption function takes a collection of documents and produces a *secure index* which can be sent to the server. A *trapdoor* function takes a keyword on which the client wants to run a search query and returns a *token* revealing nothing about the keyword. The server can then use the *token* to run the search on the *secure index* and return an encrypted collection of document identifiers which contain the keyword.

A number of researchers have published articles about SSE [CM05, vLSD+10, KO12]. They worked among other things on making SSE dynamic to allow updating the database [KPR12, KP13, SPS13, CJJ+14] and improve its expressiveness to allow boolean queries on keywords [CJJ+13, CJJ+14] as well as range queries [FJK+15].

STE. Chase and Kamara introduced *structured encryption* (STE), which generalizes SSE [CK10]. They propose an encrypted data structure similar to a secure index but which can handle various types of data. For example, they describe a construction encrypting the adjacency list of a graph

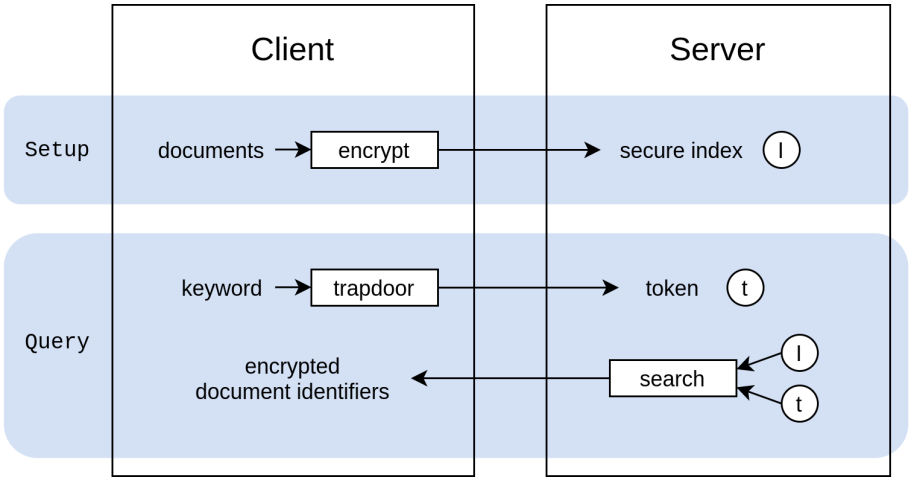


Figure 2.1: *Searchable symmetric encryption* as described in [CGKO11].

and solving neighbouring queries. Kamara et al. improved STE to support a subset of SQL queries [KM18] and diminish the amount of data leaked [KMO18, KM19].

PPE. In parallel to schemes based on an encrypted index, other work proposes to encrypt the data records in a way which preserves some properties allowing the server to perform queries on the data. *Order-preserving encrption* (OPE) [AKSX04, BCLO09], *property-preserving encryption* (PPE) [PR12] and *order-revealing encryption* (ORE) [LW16] follow this idea. The resulting schemes are more legacy-friendly since the structure of the data remains the same but have $\mathcal{O}(n)$ querying time. In the same category we consider deterministic encryption, which by definition reveals when two ciphtertexts are encryptions of the same plaintext and thus allow the server to perform queries. Bellare et al. proposed deterministic *efficiently searchable encryption* (ESE) [BBO07] based on public-key encryption.

PEKS. Another family of schemes uses public-key (or asymmertric) instead of symmetric cryptography to allow the server to run queries on the data. *Public-key encryption for keyword search* (PEKS) [BDCOP04, ABC$^+$05, BKOS07, BW07] is a multi-user searchable encryption where the public-key enables anybody to insert data while only the owner of the private key can search the database. An issue regarding PEKS is that it relies on expensive cryptographic primitives, which significantly lowers its efficiency.

Overview. Figure 2.2 summarises the previously mentioned types of encrypted database schemes. It also situates graph encryption and the GRECS scheme we will look at in details in the next section. Note that the hier-

archy in Figure 2.2 only gives an overview and is not strict or exhaustive: for example searchable encryption (SE) would englobe SSE and PEKS, but STE generalises to queries other than search and PEKS does not use an encrypted index. A more comprehensive diagram would include overlaps and be more complex.

2.1.2 Leakage profiles

Starting with a paper by Chang and Mitzenmacher [CM05], simulation-based security definitions ensure some privacy guarantees for searchable encryption schemes. Definitions like IND-CKA2 in [CGKO11] establish that an attacker with chosen keyword queries cannot distinguish the real scheme from a simulation having access to a specific leakage only. Thus the attacker does not learn more than the leakage.

The proof that a scheme satisfies a security definition provides some guarantees, but it does not measure the impact of the leakage on privacy. Cryptanalysis is for now the only known way to verify the effect of the data leaked by a scheme on its security.

Cash et al. [CGPR15] introduced the concept of *leakage profile* to categorise the leakages of various schemes. This allows attacks to target a class of schemes with a given leakage profile instead of a specific scheme. We summarise some of the leakage profiles in Table 2.1.

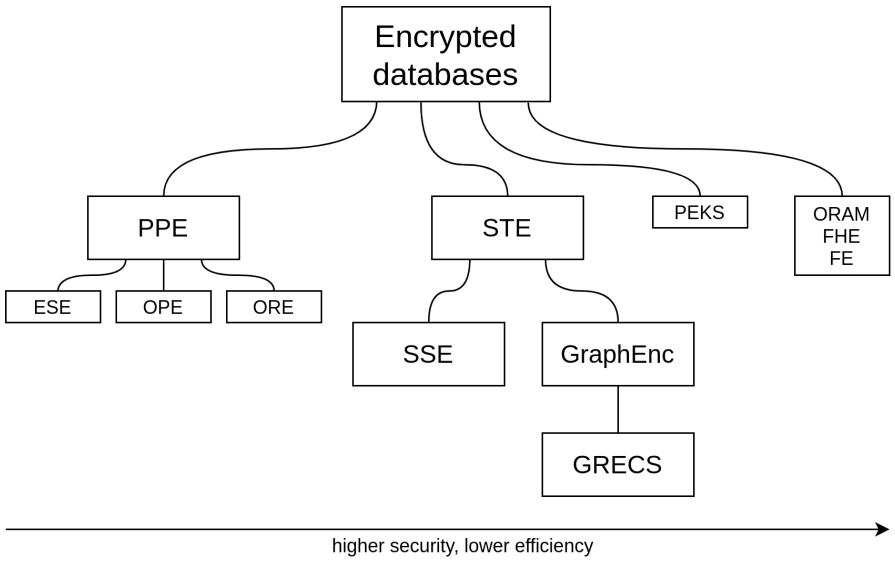


Figure 2.2: Hierarchical overview of the various encrypted database schemes we mention. Reading the diagram from left to right, the schemes provide more security and less efficiency.

Table 2.1: Some leakage profiles and examples of schemes in which they appear.

Leakage profile	Leaked data	Schemes
Access pattern	reveals common records in query results	[SWP00] [CGKO11]
Query/search pattern	reveals whether two queries are identical	[CGKO11]
Volume pattern	reveals the size of query results	[FJK+15]
Rank pattern	reveals the number of records with a value smaller than a given record	[LW16]

2.1.3 Existing attacks

Using the previously mentioned leakage profiles and making various assumptions, researchers have shown that an attacker can recover partial or full information about an encrypted database content or the queries on it. We now review some of these attacks to give a flavour of the kind of analysis we want to make on the new leakage we introduce later.

Access pattern. Islam et al. showed how a malicious server can use the access pattern to recover queries [IKK12]. They additionally require some knowledge about the plaintext in the form of a set of m keywords and a $m \times m$ co-occurrence matrix C which contains the probability that a pair of keywords appear in a same document. Observing q unique search queries, the server can use the access pattern to learn how many documents each pair of tokens appear together and build a co-occurrence matrix $\hat{C}$. Since $\hat{C}$ is a permuted submatrix of C, the server can find the best token to keyword mapping and learn which keywords where queried.

This attack showed for the first time that leakage which seems harmless for the privacy of the database could have a devastating impact when used by an attacker with a reasonable amount of additional information.

Search pattern. Liu et al. [LZWT14] investigated how the search pattern can lead the server to recover queries keywords. Assuming statistical knowledge about users' search habits, an attacker observing a large enough amount of queries can match the queried keywords having the highest frequency with the statistically most searched keywords. The prior knowledge about search habits could come from a public tool like Google Trends.

Volume pattern. Cash et al. worked on another query recovery attack [CGPR15]. They use the volume pattern – i.e. the size of the query response,

which is part of the access pattern – and assume prior knowledge about the number count$[w]$ of documents a keyword w appears in. The server can then find the best match between count and the size of the responses to learn the content of the queries. Cash et al. show that the assumed knownledge count does not need to be exact for their attack to yield good results.

File injection. Assuming a more powerful attacker able to insert documents in a database using searchable encryption, Zhang et al. mount a powerful query recovery attack [ZKP16]. The idea is to inject documents with subsets of keywords and observe whether a new query matches an inserted document, thus revealing if the queried keyword is part of a subset.

Range queries. Another family of attacks targets schemes allowing queries over ranges of ordered records [KKNO16, GLMP18, LMP18, GJW19]. A range query returns records for which an ordered field (like an integer) has a value in a given range. Knowning the access pattern and observing enough range queries, the idea is to deduce the order of the records from range overlaps [KKNO16]. The volume pattern can be used as well by matching the result sizes to ranges, again assuming enough queries. Lacharité et al. [LMP18] used the rank pattern, i.e. how many records have a lower value than a given one, to improve those kind of attacks.

We can conclude that leakage analysis plays an important role in evaluating the security of database encryption schemes. We now shift our focus to graph databases in order to later perform the same kind of analysis on the data some of them leak.

2.2 Graph Databases

The encrypted databases we have reviewed in the previous section follow the relational model, which means that the data is organised in tables with a fixed structure. Some types of data require more flexibility and the NoSQL[1] family of databases has emerged.

Graph databases are a subset of the NoSQL family where the relationships between data records are stored like values. More concretely the vertices and edges of a graph are handled similarly to allow a flexible structure. Social networks and web graphs are typical types of data for which graph databases are suited.

One could argue that a graph can be stored in relational database in the form of an adjacency list or matrix, but some graph operations requiring path traversal – i.e. going from one vertex to another using edges – would

[1] https://nosql-database.org/

be very inefficient. Thus graph databases such as Neo4j[2], GraphBase[3] or OrientDB[4] have been developped and adopted by the industry.

2.2.1 Encrypted Graph Databases

Graph databases stored by a third party raise the same privacy issues as relational databases. The server storing the database is not fully trusted and the client would like to keep the data private while still being able to query it. Schemes with this functionality have been designed, but since there is a wide range of graph operations the schemes only support a subset of them.

Chase and Kamara [CK10] introduced graph encryption along STE (see section 2.1.1) and described constructions to run neighbouring and adjacency queries. Cao et al. [CYW+11] used a feature-based index to run subgraph queries as well as k-nearest neighbours queries. Meng et al. [MKNK15] worked on a scheme running approximate shortest path queries on a graph, which was generalized to constrained shortest path queries by Shen et al. [SMZ+17] and made more accurate by Zhang et al. [ZZX+20].

In this thesis we will focus on the GRECS construction from Meng et al. [MKNK15] for approximate shortest path queries. Before examining the details of their scheme, we review the distance oracle on which it is based.

2.3 Sketch-based Distance Oracle

Given two vertices u and v in a graph, the length of the shortest path – or distance – from u to v is the minimal number of edges one has to follow to link u to v. The distance query is a basic operation in many graph algorithms but also has applications of its own, such as the number of introductions necessary for two person two meet in a social network or the similarity of two websites in a web graph.

Standard algorithms to find the distance between two vertices like Dijkstra's require to explore the various paths between the vertices and their complexity depends on the topology of the graph. One way to solve the problem is to precompute the shortest paths between each pair of vertices, but this is computationally too expensive on large graphs and requires a quadratic storage space.

2.3.1 Distance sketches

Researchers have developped a technique to precompute a logarithmic data structure, called *sketch*, which allows to approximate the shortest path be-

[2] https://neo4j.com/
[3] https://graphbase.ai/
[4] https://orientdb.org/

tween two vertices in constant time [TZ05][DSGNP10][CDF+13]. Informally, a sketch for a vertex consists of a collection of selected vertices called *seeds* and the distance to each of them. The oracle takes two sketches and searches among the common seeds (Figure 2.3) for the one building the combined path of minimal length as illustrated.

More formally, we use a notation similar to [MKNK15] and define a distance oracle as a pair of algorithms $(\mathsf{Setup}, \widehat{\mathsf{Dist}})$. Setup takes a graph $G = (V, E)$ and returns the set of sketches for each vertex. The sketch Sketch_v for vertex v is a set of tuples pairing a seed w_i with the distance $\mathsf{Dist}(v, w_i)$, i.e. the length of the shortest path between v and w_i. In short,

$$\begin{aligned}\mathsf{Setup}(G) &:= \{\mathsf{Sketch}_v\}_{v \in V} \\ \mathsf{Sketch}_v &:= \{(w_1, \mathsf{Dist}(v, w_1)), ..., (w_l, \mathsf{Dist}(v, w_l)))\},\end{aligned}$$

where l is the size of the sketch. $\widehat{\mathsf{Dist}}$ takes the sketches for two vertices u and v and returns the minimal sum of distance over the seeds common to Sketch_u and Sketch_v as an approximation of the distance $\mathsf{Dist}(\mathsf{u}, \mathsf{v})$. That is

$$\widehat{\mathsf{Dist}}(\mathsf{Sketch}_u, \mathsf{Sketch}_v) := \min_{\substack{(w_i, d_i) \in \mathsf{Sketch}_u \\ (w_j, d_j) \in \mathsf{Sketch}_v \\ w_i = w_j}} \{d_i + d_j\}.$$

The resulting distance is always an upper bound to the actual shortest path length [DSGNP10]. The tightness of this upper bound depends on how the seeds are chosen for each sketch and on the number of seeds. Next we look at two possible Setup algorithms for chosing the seeds.

2.3.2 Sketching algorithms

Das Sarma et al. The first method we look at for selecting seeds for a sketch was published by Das Sarma et al. [DSGNP10]. The idea is to have

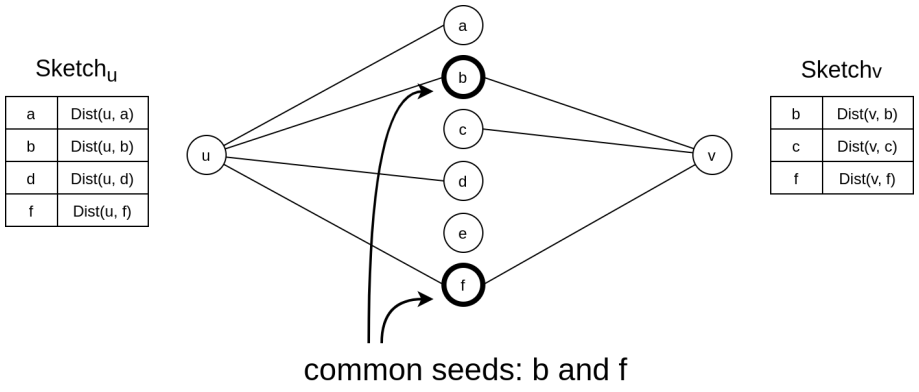


Figure 2.3: Common seeds between two sketches.

sets of vertices, from which the seeds are selected, built in a way to ensure a good balance between close seeds and seeds which appear in many sketches.

Let $n = |V|$, $r = \lfloor \log_2(n) \rfloor$ and let $S_0, S_1, ..., S_r$ be sets of seeds chosen uniformly at random over V with $|S_i| = 2^i$. We further define $(w_i, \mathsf{Dist}(v, w_i)) = \mathsf{closest}(v, S_i)$ where w_i is the vertex in S_i closest to v. This allows us to construct the minimal sketch of v:

$$\mathsf{SketchMin}_v := \bigcup_{0 \leq i \leq r} \mathsf{closest}(v, S_i).$$

To increase the precision of the distance oracle, a full sketch is the union of σ minimal sketches built each with independently sampled sets of seeds. We now have $(r+1) \times \sigma$ sets S_i^j, where $0 \leq i \leq r$, $1 \leq j \leq \sigma$, and we can construct the full sketch

$$\mathsf{Sketch}_v := \bigcup_{\substack{0 \leq i \leq r \\ 1 \leq j \leq \sigma}} \mathsf{closest}(v, S_i^j).$$

Das Sarma et al. showed that chosing $\sigma = \tilde{\Theta}(n^{2/(\alpha+1)})$ guarantees α-precision of the approximation of the distance, i.e. $\widetilde{\mathsf{Dist}}(u, v) \leq \alpha \cdot \mathsf{Dist}(u, v)$.

Cohen et al. The second sketching technique was designed by Cohen et al. [CDF+13]. The idea is again to include a seed with higher probability if it is closer, but this time the vertices are assigned a rank $\mathsf{rank}(v) \in [0, 1]$ influencing the inclusion probability instead of being chosen from predefined sets. When building the sketch for vertex v, the probability that another vertex w is included as seed in its sketch is higher when it is close to v or when its rank is small.

Let us define the Cohen et al. Setup algorithm more formally. First, the rank function is chosen to assign a value uniformly at random in $[0, 1]$ for each $v \in V$. Further, let $N_d(v)$ be the set of vertices inside the ball of radius d around vertex v, i.e. all vertices at a distance smaller than d from v. The parameter ρ allows to tune the quality of the approximation and, for a set of vertices $W \subseteq V$, we denote by $\rho^{\text{th}}_{\mathsf{rank}}(W)$ the ρ^{th} value in the ordered list of ranks of the vertices in W. We can now build a sketch for vertex v by including all seeds w which have a rank smaller than the ρ^{th} smallest rank in the set of nodes closer to v than w:

$$\mathsf{Sketch}_v := \{w \in V : \mathsf{rank}(w) < \rho^{\text{th}}_{\mathsf{rank}}(N_{\mathsf{Dist}(v,w)}(v))\}.$$

Cohen et al. showed that chosing $\rho = \Theta(n^{2/(\alpha+1)})$ guarantees α-precision in the distance approximation.

Notation. From now on we will refer to the algorithms above as the Das Sarma et al. and the Cohen et al. sketching algorithms, respectively.

2.4 GRECS

Similarly to structured encryption, the GRECS scheme encrypts the data structure storing the structure of the graph in a way that the server can still run queries on it. We will now see in details the third construction in [MKNK15], the only one computationally efficient and with optimal communication complexity, as well as its leakage.

2.4.1 Definitions

Before delving into the scheme itself we review some standard definitions of cryptographic primitives (e.g. see [KL14]).

In the following PPT stands for *probabilistic polynomial time*. A function $\nu : \mathbb{N} \to \mathbb{N}$ is *negligible* in t if for every positive polynomial $p(\cdot)$ and all sufficiently large t, $\nu(t) < 1/p(t)$. A *keyed* function F is a two-input function $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ where the first input is called *key* and is usually fixed to form a single-input function F_k such that $F_k(x) := F(k,x)$.

Definition 2.1 (Pseudo-random function – PRF) *We say a keyed function $F_k : \{0,1\}^n \to \{0,1\}^m$ is pseudo-random if it satifies the following:*

- *it is efficiently computable, i.e. given $x \in \{0,1\}^n$ and $k \in \{0,1\}^t$ there is a PPT algorithm which computes $F_k(x)$,*
- *no distiguisher has a non-negligible advantage, i.e. for any PPT algorithm D there exists a negligible function* negl *such that*

$$|\Pr[D^{F_k}(1^t) = 1] - \Pr[D^f(1^t) = 1]| \leq \mathsf{negl}(t),$$

where $k \xleftarrow{\$} \{0,1\}^t$ and $f \xleftarrow{\$} \{f : \{0,1\}^n \to \{0,1\}^m\}$.

Definition 2.2 (Pseudo-random permutation – PRP) *We say a keyed permutation $P_k : \{0,1\}^n \to \{0,1\}^n$ is pseudo-random if it satisfies the definition of a PRF with $m = n$.*

Definition 2.3 (Somewhat homomorphic encryption) *We say a somewhat homomorphic encryption (SWHE) scheme is a tuple of algorithms* $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *such that* $(\mathsf{Enc}, \mathsf{Dec})$ *is a valid encryption scheme and*

$$\mathsf{Enc}(x_1 \odot x_2) = \mathsf{Eval}(\odot, \mathsf{Enc}(x_1), \mathsf{Enc}(x_2))$$

for all x_1, x_2 and some operation $\odot$.

In the context of GRECS we need a SWHE scheme which supports any number of additions and one single multiplication as described in [MKNK15]. A possible implementation is the scheme by Gentry et al. [GHV10].

2.4.2 Construction

The construction is a pair of algorithms $(\mathsf{Setup}, \mathsf{Query})$. The Setup algorithm takes the sketches $\{\mathsf{Sketch}_v\}_{v \in V}$ of a graph $G = (V, E)$ and returns an array A and a dictionary D along with the generated keys. Informally, A contains the encrypted seed/distance pairs of the sketches as nodes of a permuted linked lists. D maps an encryption of each vertex identifier to the start of its sketch in A. Each sketch is thus "spread" over A but can be recovered by following the linked nodes starting from its head at which D points. To allow the server to actually compute the shortest path length without learning the distance stored in the sketches, the distances are encrypted with a somewhat homomorphic encryption (SWHE) scheme.

Let us describe the algorithm in more details. We leave out the sizes of the keys and of the domains refering to [MKNK15] for a formal definition. Before starting, we need an SWHE scheme $\mathsf{SWHE} = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, a PRP P, a PRF F and a random oracle H as defined in 2.4.1.

Setup. First, the keys (pk, sk, K_1, K_2) are sampled, where (pk, sk) is a public/private key pair for SWHE, K_1 is a key for the PRP P and K_2 is a key for the PRF F. A collision-resistant hash function h and a permutation π over $[Z]$ are sampled as well, where Z is the total number of seed/distances pairs, i.e. the sum of all sketch sizes. The algorithm then iterates over the seed/distance pairs in all sketches and a global counter ctr increases for each pair. When processing the seed/distance (w_i, d_i), Setup first encrypts d_i with SWHE as c_i – we will explain how exactly later. A pointer to the next node nxt_i is calculated, set to $\pi(\mathsf{ctr} + 1)$ or NULL if it is the last pair of the sketch. It then builds a node $\mathsf{N}_i := \langle h(w_i) \| c_i \| \mathsf{nxt}_i \rangle$. The node is encrypted and stored in A:

$$A[\pi(\mathsf{ctr})] := \langle \mathsf{N}_i \oplus H(K_v \| r_i), r_i \rangle,$$

where K_v and r_i are random values, sampled respectively for each vertex and seed/distance pair.

A pointer to the start of each sketch v is then stored in D, using the PRP P and PRF F to encrypt it:

$$D[P_{K_1}(v)] := \langle \pi(\mathsf{ctr}_v) \| K_v \rangle \oplus F_{K_2}(v),$$

where ctr_v is the value of the global counter when processing the first seed/distance pair of $\mathsf{Sketch_v}$.

Figure 2.4 illustrates the encryption of a simple sketch with three seeds. When the client has run Setup, they keep the keys and send (A, D) to the server.

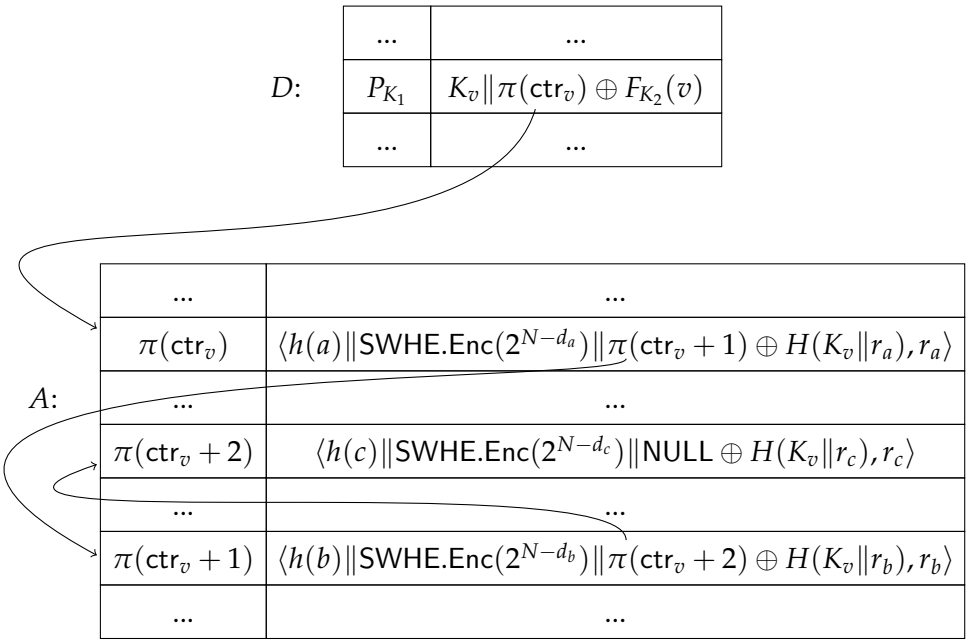


Figure 2.4: Encryption of $\mathsf{Sketch}_v = \{(a, d_1), (b, d_2), (c, d_3)\}$. The arrows indicate how the server recovers the pointers to the array entries and reconstructs the sketch.

SWHE. Coming back on the distance encryption, Setup does not directly encrypt d_i, but instead

$$c_i := \mathsf{SWHE.Enc}_{pk}(2^{N-d_i}),$$

where N is a large enough constant. More precisely, for D the maximal distance in all the sketches, $N = 2 \cdot D + 1$. Roughly, this allows the Query algorithm to find an approximation of the minimum combined distance between two sketches only with addition and multiplication operations, which can be evaluated by SWHE.Eval on ciphertexts.

Query. To query the distance between vertices u and v, the client has to send $(P_{K_1}(u), P_{K_1}(v), F_{K_2}(u), F_{K_2}(v))$ to the server. The server starts by reading the pointers to the start of both sketches

$$\langle a_{u,0} \| K_u \rangle := D[P_{K_1}(u)] \oplus F_{K_2}(u)$$
$$\langle a_{v,0} \| K_v \rangle := D[P_{K_1}(v)] \oplus F_{K_2}(v).$$

The encrypted sketches are then reconstructed by recursively recovering the nodes

$$\langle \sigma_i, r_i \rangle := A[a_{u,i}]$$

$$\mathsf{N}_{u,i} := \sigma_i \oplus H(K_u \| r_i),$$

which can be parsed as $h_{u,i} \| c_{u,i} \| a_{u,i+1} := \mathsf{N}_{u,i}$, and similarly for v. The server can then iterate over the hashed seed identifiers and sum the products of the corresponding encrypted ditances using SWHE.Eval:

$$s := \sum_{h_{u,i}=h_{v,j}} c_{u,i} \cdot c_{v,j}.$$

s is sent to the client, which can decrypt it as $d := 2N - \log_2 \mathsf{SWHE.Dec}_{sk}(s))$. Since the distances were encrypted as $\mathsf{SWHE.Enc}_{pk}(2^{N-d_i})$, we have

$$\begin{aligned} d &= 2N - \log_2 \left(\sum_{w_{u,i}=w_{v,j}} 2^{N-d_{u,i}} \cdot 2^{N-d_{v,j}} \right) \\ &= 2N - \log_2 \left(2^{2N} \cdot \sum_{w_{u,i}=w_{v,j}} 2^{-(d_{u,i}+d_{v,j})} \right) \\ &= -\log_2 \left(\sum_{w_{u,i}=w_{v,j}} 2^{-(d_{u,i}+d_{v,j})} \right) \\ &\leq \min_{w_{u,i}=w_{v,j}} d_{u,i} + d_{v,j}, \end{aligned}$$

where a proof of the last step as well as a lower bound for d can be found in [MKNK15].

2.4.3 Leakage

The GRECS scheme leaks two types of information: the query pattern, which we already know from previous schemes (cf Table 2.1), and the sketch pattern.

Since the encryption of the queries is deterministic, the server can detect when queried vertices are identic. Thus, the query pattern of GRECS for m queries consists of an $m \times m$ matrix where the entry for $q = (u, v)$ and $q' = (u', v')$ reveals whether $u = u'$, $u = v'$, $v = u'$ and $v = v'$. A similar leakage profile has been analysed by Liu et al. [LZWT14].

The second and new type of leakage consists of the pattern of common seeds across sketches. Once a vertex has been queried, the server can learn the hash of the seed identifiers in a sketch, as well as the size of each sketch. Assuming that each vertex has been queried, the server can then learn how frequent a seed identifier hash is in the sketches. Since the sketches have been constructed in such a way that they contain more close seeds, some information about the vertices could be inferred.

As far as we know, the sketch pattern has not been thorougly analysed yet. In the next chapter we explore what data about the distances in the sketches can be deduced from the sketch pattern.

Chapter 3

Distances Estimation from the Sketch Pattern Leakage

In this chapter we will analyse the sketch pattern leaked by the GRECS scheme [MKNK15] and try to recover useful information about the distance sketches. For now, we do not try to mount an attack to recover data about the graph itself or the queries a client is sending, but we only examine the structure of the sketch pattern and attempt to construct sketch estimates as close as possible to the original sketches.

We start by defining of what the sketch pattern exactly consists. We then consider sketches from the Das Sarma et al. sketching algorithm (see section 2.3.2) and try to reverse the process of building them to gain knowledge about the distance to each seed. Next, we adapt our technique to the Cohen et al. sketches and, lastly, we evaluate how our method performs on sketches of self-generated and real-life graphs.

3.1 Sketch Pattern Leakage

Definition 3.1 (Sketch pattern) *The sketch pattern of a set of sketches consists of sets of "pseudo-ids" of the seeds, that is*

$$\begin{aligned}\mathsf{SketchPattern}(\{\mathsf{Sketch}_v\}_{v\in V}) &:= \{\mathsf{SP}_v\}_{v\in V}\\ \mathsf{SP}_v &:= \{f(w) : (w,d) \in \mathsf{Sketch}_v\},\end{aligned}$$

where $f : V \to V$ is a random function.

The sketch pattern is leaked in the GRECS scheme from Meng et al. but not exclusively. The Connor scheme by Shen et al. [SMZ+17] extends to cost-constrained shortest path queries, but it is based on distance sketches as well and leaks the sketch pattern. Zhang et al. went futher in their PGAS

scheme [ZZX+20] by making the distance queries accurate, but they conserve the same leakage profile under the name *label pattern leakage*. Thus investigating the impact of the sketch pattern leakage on the privacy is relevant to multiple schemes.

In order to understand the relationship between the sketches and the sketch pattern, we start by examining in details the case of Das Sarma et al. sketches.

3.2 Das Sarma et al. Sketches

Remember from section 2.3.2 that the seeds come from the sets of seeds $S_0, S_1, ..., S_r$, where $r = \lfloor \log_2(n) \rfloor$, $|S_i| = 2^i$ and $S_i \subseteq V$ is sampled uniformly at random. To reflect the exploration approach adopted during the research phase of this thesis, we will decompose the recovery of distances in the sketches in two steps, as illustrated in Figure 3.1. In a first step, we are going to estimate from which seed set S_i a seed w of the sketch Sketch_v comes from. Then, we will estimate the distance between v and w knowing from which seed set it comes from.

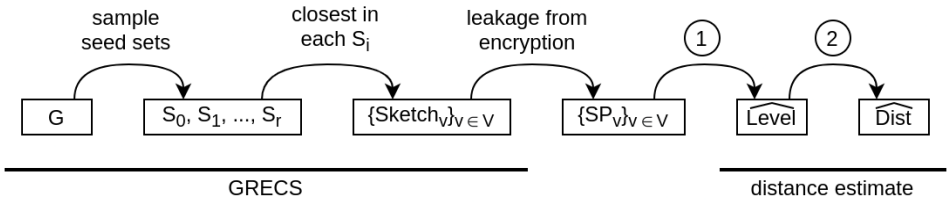


Figure 3.1: Overview of the Das Sarma et al. sketch construction and plan for the distance estimate from the sketch pattern leakage. Step 1 estimates from which seed set a seed comes from and step 2 estimates the distance in the sketch from the level of the seed.

3.2.1 Step 1: seed level recovery

Since a Das Saram sketch is the union of multiple minimal sketches (cf. section 2.3.2) which each sample their own seed sets, we will from now on only consider the *level* of the seed set.

Definition 3.2 (Level of a seed set) *For a seed set S_i, its level is $\log_2(|S_i|)$, namely i.*

We will also use the phrase *level of a seed w* to designate the level of the seed set from which a seed comes from.

We now make an observation linking the level of a seed to its *frequency* in a sketch pattern.

Definition 3.3 (Frequency of a seed) *For a seed w and a set of sketches $\{\mathsf{Sketch}_{\mathsf{v}v \in V}\}$, we denote by $\mathsf{Freq}(w)$ the number of sketches it appears in, that is*

$$\mathsf{Freq}(w) := |\{\mathsf{Sketch}_{\mathsf{v}} : (w, \cdot) \in \mathsf{Sketch}_{\mathsf{v}}\}|,$$

where the symbol $\cdot$ represents any value.

Since a sketch gets one seed from each seed set, there will be the same number of seeds from each seed set in the set of sketches. For now, we ignore overlaps between seed sets and assume $S_i \cap S_j = \emptyset \ \forall i, j \in \{0, ..., r\}$. Further, we note that the seed set S_{i+1} contains twice as many seeds as S_i by definition, thus a seed from S_{i+1} will in average appear in half as many sketches as a seed from S_i, as illustrated in Figure 3.2.

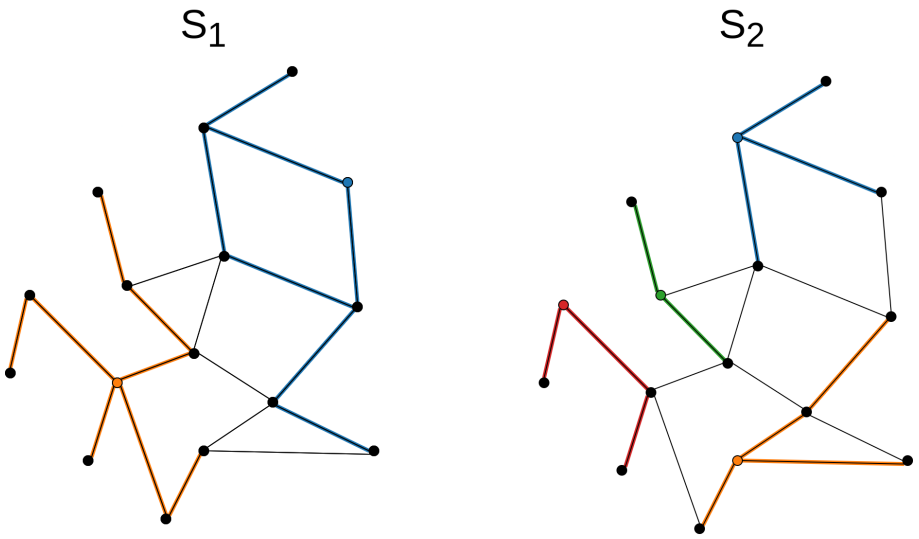


Figure 3.2: In this graph with 16 vertices, a seed of S_1 appears in average in 8 sketches and a seed of S_2 in average in 4 sketches.

This also means that if the frequency of a seed is lower than the one of another seed, it probably came from a higher level. Thus, we should be able to approximate from which seed set a seed comes from only by observing its frequency.

First attempt

Since $|S_i| = 2^i$, let's assume that each sketch has probability $\frac{1}{2^i}$ to contain a seed w of level i. Thus, the frequency of seed w is binomially distributed,

that is

$$\mathsf{Freq}(w) \sim \mathrm{B}\left(n, \frac{1}{2^i}\right). \tag{3.1}$$

Assuming that the frequency is close to its expected value $\frac{n}{2^i}$, we can formulate our first attempt to estimate the level of a seed given its frequency:

$$\widehat{\mathsf{Level}}(w) := \log_2\left(\frac{n}{\mathsf{Freq}(w)}\right). \tag{3.2}$$

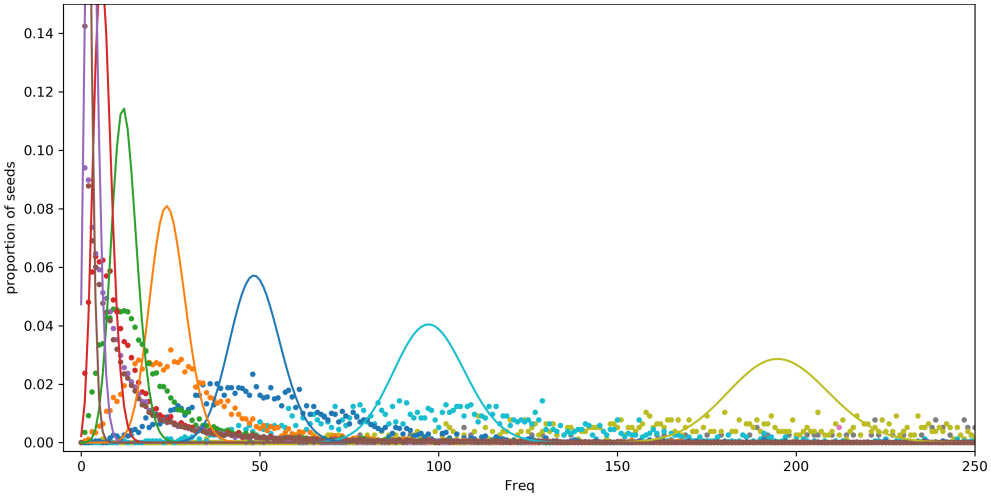
At this stage, we conduct a small experiment with a random graph to confirm whether we are on the right path. First, we generate the sketches and we plot in Figure 3.3a a histogram of the seed frequencies, using dots of a different color for each level. We then compare these distributions with our first hypothesis (3.1), represented with continuous lines in Figure3.3a. Although the distributions of the different levels overlap, we can distinguish that the actual distribution are relatively well aligned with the binomial ones. We note that the observed distributions are flatter, which probably means that the seeds are not as uniformly distributed as we assumed. In Figure 3.3b we display the average frequency of seeds for each level along with our first level estimate (3.2).

Taking the seed sets overlap into account

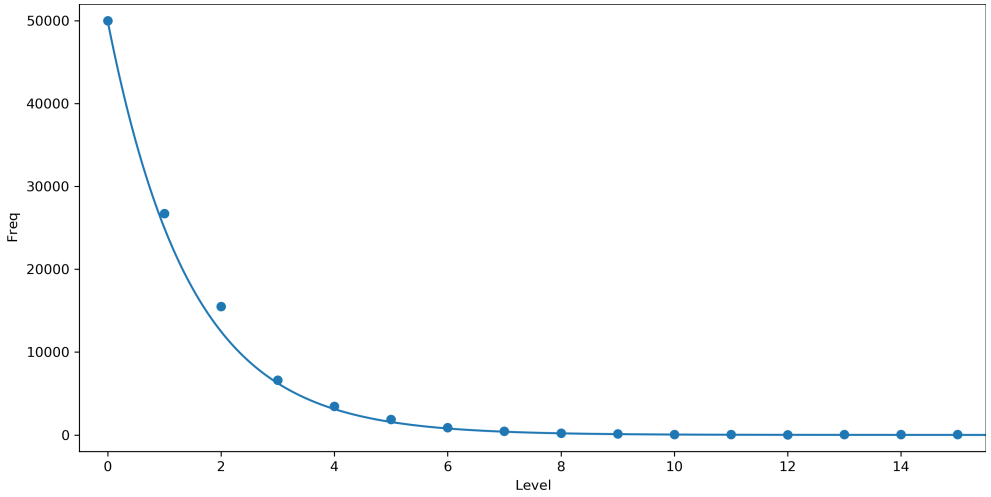
We mentioned earlier that we were ignoring the fact that two seed sets might overlap. To improve our level estimate, we have to take into account that a seed which was assigned to a sketch for a particular level might actually also appear in another seed set, which will increase its frequency. When observing the frequency of a seed of level i, we have to consider all the 2^r combinations of the r other levels in which the seed can be. Since the seed sets are sampled independently from each other, the probability p_J that a sketch contains a seed w appearing in a subset of a levels $J \subseteq \{0, ..., r\}$ is

$$\begin{aligned} p_J &= \Pr\left[(w, \cdot) \in \mathsf{Sketch}_v \mid w \in \bigcap_{j \in J} S_j\right] \\ &= \max\left(1, \sum_{j \in J} \frac{1}{2^j}\right), \end{aligned}$$

where the maximum is necessary because S_0 contains only one seed and makes the sum go over 1. The frequency of a seed appearing in all those levels is then binomially distributed with parameters n and p_J. Further, the probability q_J that seed w of level i also appears in the other levels



(a) Proportion of seeds having a given frequency. Each color represents a different level.



(b) Average frequency of seeds of a given level. The full line is the level estimate $\widehat{\text{Level}}$.

Figure 3.3: Seed level experiment on Das Sarma et al. sketches (with $\sigma = 3$) of a random graph $G_{n,p}$ with $n = 50000$ and $p = 0.002$.

$J \subseteq \{0, ..., r\} \setminus \{i\}$ is, again thanks to the independence of the seed sets,

$$\begin{aligned} q_J &= \Pr\left[w \in \bigcap_{j \in J} S_j\right] \\ &= \prod_{j \in J} \Pr[w \in S_j] \\ &= \frac{2^j}{n}. \end{aligned}$$

We can now formulate the probability mass function (PMF) of the event "w has frequency x given that it is in set S_i" as the weighted sum of the binomially distributed PMFs for each of the 2^r seed sets combination. That is

$$\Pr[\mathsf{Freq}(w) = x \mid (w, \cdot) \in S_i] = \sum_{J \subseteq \{0,...,r\} \setminus \{i\}} q_J \cdot f(x, n, p_{J \cup \{i\}}), \tag{3.3}$$

where $f(x, n, p) = \Pr[X = x]$ given that $X \sim \mathrm{B}(n, p)$.

Similarly to our first attempt, we check with a small visual experiment whether the frequency distribution fits (3.3). Figure 3.4 shows that our improved approximation of the frequency is closer to the actual values but still does not fit perfectly.

The observed gap could, for example, be caused by the fact that we only considered sketches with $\sigma = 1$ in our reasoning. We could explore possibilities to further improve the seed level recovery, but for now we move on to the next step, namely to approximate distances in a sketch with the help of our seed level estimate.

3.2.2 Step 2: distance recovery from the seed level

Now that we are able to approximate from which level a seed w in Sketch_v comes from, we can take the next step and use the level to recover the distance between v and w.

In order to picture how the distances in a sketch are related to the level of the seeds we will move our focus from a particular sketch with its seeds to the reverse mapping, i.e. a particular seed and all sketches it appears in.

Definition 3.4 (Subgraph of a seed) *For a graph $G = (V, E)$ and a set of sketches $\{\mathsf{Sketch}_v\}_{v \in V}$, the subgraph of a seed w is the subgraph $H \subseteq G$ induced by the subset of vertices V' for which the sketch contains w, that is*

$$V' := \{v \in V : (w, \cdot) \in \mathsf{Sketch}_v\}.$$

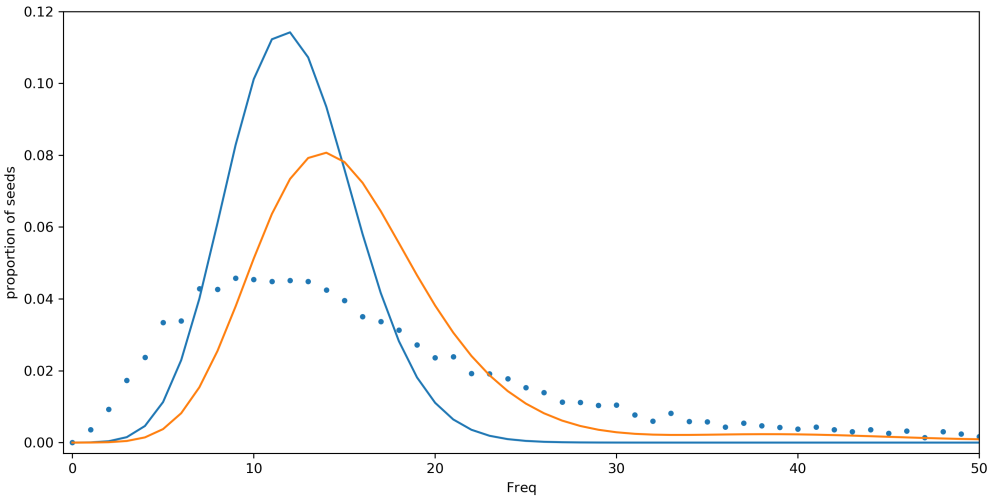


Figure 3.4: Frequency distribution for seeds of level 12, the dots are values of a random graph with $n = 50000$, the blue line is the basic binomial distribution and the orange one the improved PMF.

We make the following observation: if a seed appears in more sketches it will in average be further from the vertices in its subgraph. More abstractly, the bigger is a subgraph, the bigger is the average distance from one vertex to all other vertices of the subgraph. This can for example be observed in Figure 3.2: the seeds of the seed set S_1 will have a higher distance in the sketches than the seeds of S_2.

Definition 3.5 (Average path length of a vertex in a graph) *The average path length (or average distance) of a vertex u in graph G is the average length of paths between u and v for all $v \in V$. That is*

$$l_G(u) = \frac{1}{|V|} \sum_{v \in V} \mathsf{Dist}(u, v)$$

where $\mathsf{Dist}(u, v)$ is the length of the shortest path from u to v.

We assume that the average path length of a seed w in its subgraph is a good approximation for the distance between v and w in the sketch of v. Thus, we make an observation linking the average path length to the size of a subgraph.

For a vertex v in the subgraph H of a seed w, every vertex v' on the shortest path between w and v is in the subgraph H as well, thus the subgraph is "dense". This means that if we take a second subgraph H' of seed w' such that $|H'| > |H|$, the average path length $l_{H'}(w')$ is probably greater than $l_H(w)$.

How the distance relates to the level depends on the type of graph. We start by considering an intuitive approach and embed the graph in an Euclidian space with N dimensions. Later, we will reuse results from Fronczak et al. [FFH04] about the average path length in a random graph to formulate a second approach.

Graph embedded in an N-dimensional Euclidian space

Let us consider a graph embedded in an N-dimensional Euclidian space, where the shortest path length between two vertices corresponds to their Euclidian distance.

If we approximate the seed subgraphs as N-dimensional balls centered around their seed, the average path length of a seed in its subgraph can be deduced from the size of the subgraph. Indeed, since the volume of an N-ball of radius R is proportional to R^N and the average distance between the center and a point of the same N-ball is proportional to R (see Appendix A), we can approximate the average path length in the subgraph H of seed w by ignoring the constants with

$$\widehat{l_H}(w) = |V'|^{1/N},$$

where V' is the set of vertice of subgraph H.

As the graphs we consider in practice may have a high Euclidian dimension because of a high maximum degree [FKS20], we will use the average degree $\overline{k}$ of the graph to define the average dimension $D = \overline{k}/2$ — imagine a D-dimensional grid, each node has $2D$ neighbours. We now formulate our first estimate for the distance assigned to a seed w in a sketch:

$$\widehat{\mathsf{Dist}}_1(w) = \left(\frac{n}{2^{\widehat{\mathsf{Level}}(w)}}\right)^{1/D}$$

where we used the fact that the average subgraph size of a seed with level $i = \widehat{\mathsf{Level}}(w)$ is $\frac{n}{2^i}$.

After this quite geometrical approach, we move on to a method more adapted to the type of graphs for which the GRECS scheme was designed.

Random graph approach

As the GRECS experiments [MKNK15] only use real-life graphs with small-world properties and a small diameter (see section 3.2.4), we will consider a model which can have similar properties. The Erdős-Rényi model (see 1.4) is the simplest model of random graphs and the diameter of $G_{n,p}$ can be controlled by adjusting the parameters n and p.

Fronczak et al. [FFH04] established that the average path length in a random graph depends only on its size and average. That is

$$l_{G_{n,p}} = \frac{\ln n - \gamma}{\ln \bar{k}} + \frac{1}{2},$$

where $\gamma \simeq 0.5772$ is Euler's constant and $\bar{k}$ is the average vertex degree.

We want to apply this result to the seed subgraphs. As in the previous approach, we approximate the size of a subgraph with its level. To determine the average degree in a subgraph, we observe that the subgraphs are formed by taking vertices closest to a seed, thus only few edges will be lost in the "partitioning" process. This means that the average degree of a seed subgraph will be close to the average degree of the original graph.

This leads us to our second estimate for the distance assigned to a seed w of level i

$$\widehat{\mathsf{Dist}}_2(w) = \frac{\ln \frac{n}{2^i} - \gamma}{\ln \bar{k}} + \frac{1}{2},$$

where we assume a constant average degree $\bar{k}$.

3.2.3 Putting things together: distance recovery from the seed frequency

In step 1 we approximated the seed level from the seed frequency and in step 2 we used the seed level to estimate the distance assigned to the seeds in sketches. We now chain both steps and simplify the result.

We notice that in both approaches for the distance estimate we use the size of the subgraph, which we approximate by $\frac{n}{2^i}$ with its level i, but not the level only. Since the size of a subgraph is actually the frequency of its seed, we can simplify the distance estimates:

$$\widehat{\mathsf{Dist}}_1(w) = (\mathsf{Freq}(w))^{1/D}$$

and

$$\widehat{\mathsf{Dist}}_2(w) = \frac{\ln \mathsf{Freq}(w) - \gamma}{\ln \bar{k}} + \frac{1}{2}.$$

Equipped with those tools, we will evaluate how good they perform in the next section.

3.2.4 Experiments

Keeping in mind that we later want to approximate the result of an encrypted query with the sketch pattern only, we for now evaluate how well we can estimate the content of the sketches. First, we define some metrics we use to measure the quality of the estimates and then experiment on self-generated as well as real-life graphs.

Quality of the distance recovery

We will measure the quality of the distance estimate with two different metrics. The first one indicates how well the distance estimate approximates the distance in the unencrypted sketch and the second measures how well the distance estimate recovers the order of distances.

More formally, the distance recovery advantage for the sketches $\{\mathsf{Sketch}_v\}_{v\in V}$ is

$$\mathsf{Adv}_{\widehat{\mathsf{Dist}}} := \frac{l_{\mathsf{Sketch}} - \mathsf{RMSE}}{l_{\mathsf{Sketch}}},$$

where l_{Sketch} is the average distance in the sketches and RMSE is the *root-mean-square* error of the distance estimate of each seed in each sketch. The advantage will be 1 if the estimate is exact and 0 if it randomly chooses a distance in the sketches.

The ordering advantage of the distance estimate reflects whether the distance estimates are sorted like the actual distances:

$$\begin{aligned}\mathsf{Adv}_{\leq} := \mathbb{E}\Big(&\mathbb{1}(\widehat{\mathsf{Dist}}(v_i, w_i) \leq \widehat{\mathsf{Dist}}(v_j, w_j)) \\ &-\mathbb{1}(\mathsf{Dist}(v_i, w_i) \leq \mathsf{Dist}(v_j, w_j))\Big) \cdot 2 - 1,\end{aligned}$$

where the expectation goes over all vertex/seed pairs $((v_i, w_i), (v_j, w_j))$.

Datasets

Our experiments, in this chapter and the next, are run on some of the real-life datasets used by Meng et al. in [MKNK15]. Table 3.1 summarises their properties and Table 3.2 indicates their context. They are accessible from the Standford SNAP website [LK14]. Note that the diameter of the graphs is small compared to the number of vertices, which is a property observed in many real-life graphs [LKF05]. Note as well that we mostly use the largest connected component of the graph – which spans most of it, see Table 3.1 – to avoid undefined distances and simplify the implementation.

In this section we run our distance recovery algorithm on a random graph $G_{n,p}$ (with $n = 50000$ and $p = 0.0002$), the CondMat and the Enron datasets.

Setting

The Das Sarma et al. sketches are generated with $\sigma = 3$ as in the GRECS paper [MKNK15]. We approximate the distances with our two estimates, namely $\widehat{\mathsf{Dist}}_1$ for the Euclidian space approach and $\widehat{\mathsf{Dist}}_2$ for the random graph approach. The distances in all sketches are considered to evaluate the recovery advantage, but we only take in account a subset of 10'000 sketches for the ordering advantage since all pairs have to be considered for the latter and this results in an extended computing time.

Table 3.1: Datasets used in our experiments.

Name	Vertices	Edges	Average degree	Diameter	Largest CC
CondMat	23'133	93'439	8.08	14	21'363
Enron	36'692	183'831	10.02	11	33'696
Gowalla	196'591	950'327	9.67	14	196'591

Table 3.2: Context of the real-life datasets.

Name	Description
CondMat	Authors of articles about Condensed Matter research linked if they co-authored an article
Enron	Email adresses of the Enron Corporation linked when at least one email was exchanged
Gowalla	Users of the location-based social network Gowalla and their friendships

Discussion of the results

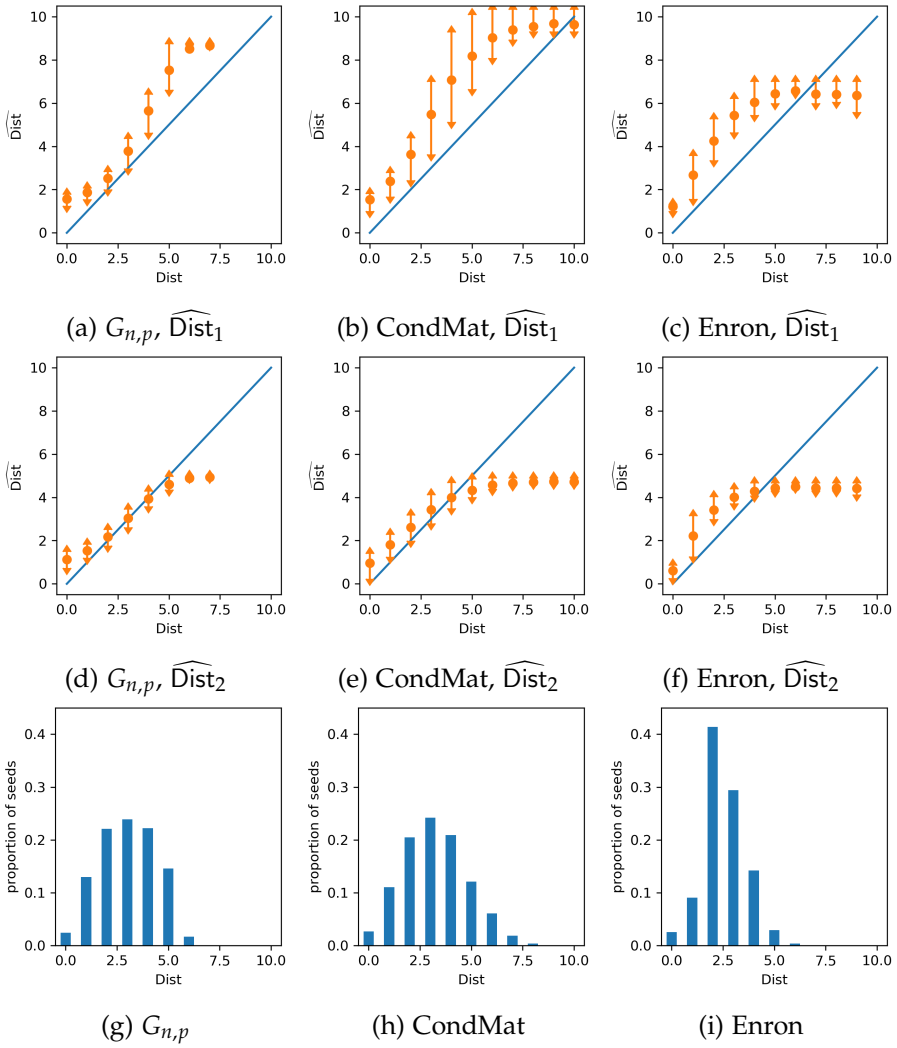
Figure 3.5 gives an overview of the performance of our estimates. Each row of subfigures corresponds to a dataset. The first row shows the results for the first estimate $\widehat{\mathsf{Dist}}_1$, the second for $\widehat{\mathsf{Dist}}_2$ and the third row reveals the distribution of distances in the sketches, which allows us to weight which range of distances are more important to recover accurately.

Table 3.3 shows the quantitative results of our experiments. We note that the recovery advantage of the first estimate $\mathsf{Adv}_{\widehat{\mathsf{Dist}}_1}$ is very low, especially for the real-life datasets. This can be linked with the relatively large error of the estimate $\widehat{\mathsf{Dist}}_1$ for frequent distances, as illustrated in Figures 3.5b and 3.5c. Nevertheless we do not discard $\widehat{\mathsf{Dist}}_1$ because it performs better in the actual attacks (see section 4.4.1). The recovery advantage of $\widehat{\mathsf{Dist}}_2$ as well as the ordering advantages are more promising and clearly show that the frequency of the seeds calculated from the sketch pattern allow us to recover partial information about the distances in the sketches.

3.3 Cohen et al. Sketches

We will now apply a procedure similar to the one used in the previous section to extract distance information from the sketch pattern resulting from the Cohen et al. sketching algorithm.

Remember from section 2.3.2 that in this setting the vertices are assigned a



(a) $G_{n,p}$, $\widehat{\text{Dist}}_1$ (b) CondMat, $\widehat{\text{Dist}}_1$ (c) Enron, $\widehat{\text{Dist}}_1$

(d) $G_{n,p}$, $\widehat{\text{Dist}}_2$ (e) CondMat, $\widehat{\text{Dist}}_2$ (f) Enron, $\widehat{\text{Dist}}_2$

(g) $G_{n,p}$ (h) CondMat (i) Enron

Figure 3.5: The first 6 figures show the median recovered distance from Das Sarma et al. sketches compared with the actual distance, with a range between the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles. The blue line is the identity function as reference. The last 3 figures show the distance distribution in the sketches.

Table 3.3: Measurement of the quality of sketch distance recovery on various datasets.

Data set	l_{Sketch}	$\text{Adv}_{\widehat{\text{Dist}}_1}$	$\text{Adv}_{\leq_1}$	$\text{Adv}_{\widehat{\text{Dist}}_2}$	$\text{Adv}_{\leq_2}$
$G_{n,p}$	2.99	0.413	0.687	0.787	0.684
CondMat	3.23	0.014	0.541	0.668	0.536
Enron	2.54	0.016	0.425	0.473	0.445

rank, and a vertex w is a seed in sketch Sketch_v if and only if its rank is lower than the ρ^{th} rank in the set of vertices closer to v than w. This means that a vertex with a low rank will probably be a seed in more sketches than a seed with a high rank, thus we could approximate the rank of a seed knowing its frequency as a first step and procede by estimating the distance of a seed from its rank.

But as we have seen with the distance estimates in the last section (see 3.2.3), these two steps partially cancel each other, therefore we will skip them and directly estimate the distances from the frequency of the seeds.

3.3.1 Distance recovery

Examining the algorithm to construct the sketches from section 2.3.2, we note that if a seed w is assigned to a vertex v, all vertices on the shortest paths between w and v also contain w in their sketches. Thus, the subgraph built by taking all vertices containing a particular seed is "dense" in the sense that it forms a group of vertices close to each other without "holes".

To approximate the distance from a seed to a vertex in a sketch we use the size of its subgraph, which is equivalent to the frequency of the seed over all sketches. Since this reasoning is exactly the same as in the case of the Das Sarma et al. sketches (see 3.2.3), we reuse the same distance estimates:

$$\widehat{\mathsf{Dist}}_1(w) = (\mathsf{Freq}(w))^{1/D},$$

where $D := \overline{k}/2$, and

$$\widehat{\mathsf{Dist}}_2(w) = \frac{\ln \mathsf{Freq}(w) - \gamma}{\ln \overline{k}} + \frac{1}{2}.$$

3.3.2 Experiments

Repeating the same experiments as for the Das Sarma et al. sketches, we use Cohen et al. sketches with a precision parameter $\rho = 4$ and estimate the distances with the average degree $\overline{k}$ as only prior knowledge about the graphs.

Discussion of the results

We observe in Figure 3.6 and in Table 3.4 that the experiments on Cohen et al. sketches yield results very similar to the ones on Das Sarma et al. sketches. This is expected since both sketching algorithms rely on a balanced distribution of close and farther seeds to give good distance approximations.

We mention here that it is impossible for estimates based on the sketch pattern to be exact since there is less information in the sketch pattern than

in the sketches themselves. This means that the accuracy of our estimates may be improved, but it is expected that it will never be perfect.

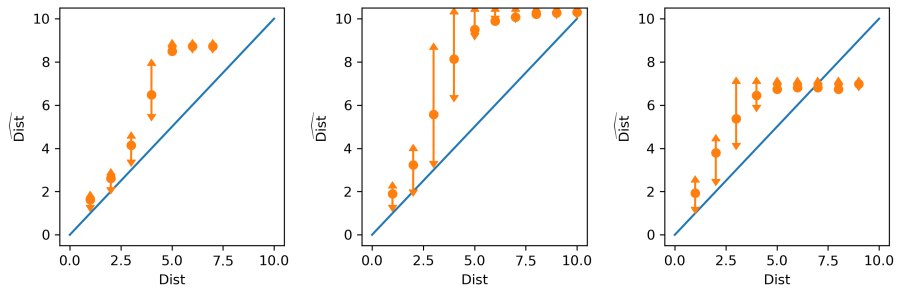
3.4 Chapter Summary

In this chapter we decomposed both the Das Sarma et al. and the Cohen et al. sketching algorithms to "reverse engineer" them and exploit the sketch pattern leakage. We used two different approches to estimate the distances in the sketches from the seed frequency and evaluated the performance of our estimates on self-generated and real-life graphs.

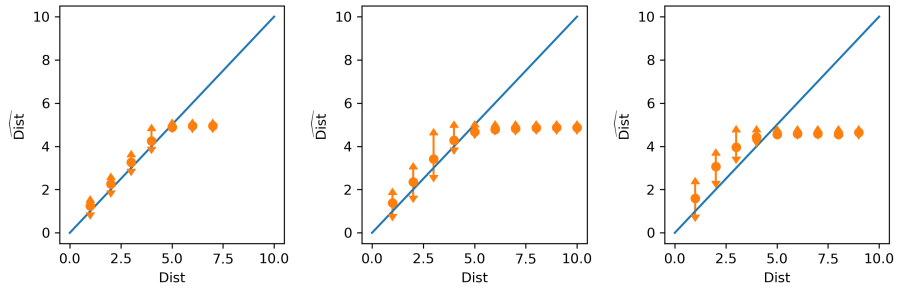
Even if the values and plots resulting from our experiments show that our estimates are not very accurate or precise, we still observed that a non-negligible quantity of information can be recovered from the sketch pattern leakage. This information can potentially be amplified and used by an attacker to gain knowledge about the encrypted graph or queries on it, as we will attempt to demonstrate in the next chapter.

Table 3.4: Meaurement of the quality of distance recovery in Cohen et al. sketches.

Data set	l_{Sketch}	$\mathsf{Adv}_{\widehat{\mathsf{Dist}}_1}$	$\mathsf{Adv}_{\leq_1}$	$\mathsf{Adv}_{\widehat{\mathsf{Dist}}_2}$	$\mathsf{Adv}_{\leq_2}$
$G_{n,p}$	3.01	0.279	0.768	0.817	0.768
CondMat	3.60	0.026	0.646	0.653	0.644
Enron	2.66	0.127	0.598	0.577	0.586



(a) $G_{n,p}$, $\widehat{\mathrm{Dist}}_1$ (b) CondMat, $\widehat{\mathrm{Dist}}_1$ (c) Enron, $\widehat{\mathrm{Dist}}_1$



(d) $G_{n,p}$, $\widehat{\mathrm{Dist}}_2$ (e) CondMat, $\widehat{\mathrm{Dist}}_2$ (f) Enron, $\widehat{\mathrm{Dist}}_2$

Figure 3.6: Median recovered distance from Cohen et al. sketches compared with the actual distance, with a range between the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles. The blue line is the identity function as reference.

Chapter 4

Leakage-based Attacks on GRECS

When a client sends an encrypted graph to a server which stores it and runs queries on it as in the GRECS scheme [MKNK15], the server learns among other things the sketch pattern. This leakage consists of the seed pseudo-identifiers in each of the sketches (see Definition 3.1), which allows the server to identify frequent and less frequent seeds across the sketches. After our preliminary exploration of the sketch pattern leakage in the last chapter we now attempt to recover information about the encrypted graph and queries.

In the following we expose our findings about ways to exploit the sketch pattern leakage. Not all the techniques lead to a feasible or useful attack in a real-life situation, but we describe them nevertheless as possible building blocks of more complex exploit chains.

We start by defining the assumptions we make about an attacker on the GRECS scheme. The first attack we then describe recovers some general information about the graph only requiring a *honest-but-curious* adversary. Assuming some more knowledge we are able to recover queried vertices. We further vary the settings of the query recovery attack to make it more accessible.

4.1 Attacker Model

The adversarial model we consider in this chapter is that of a server executing the protocol correctly but trying to gain knowledge private to the client. Indeed, the sketch pattern is revealed during the Query algorithm of the GRECS construction and cannot be inferred from the communication with the client. Therefore, a man-in-the-middle attacker having access to messages exchanged would not learn the sketch pattern. Hence we will only consider the server as potential adversary.

Honest-but-curious. In a first step we consider a *honest-but-curious* server. This means that the server executes the protocol as expected but attempts to deduce some knowledge about the content of the database or the queries. We make two reasonable assumptions. First, the server knows about the average degree $\bar{k}$ of the vertices of the graph, which is equivalent to knowing the type of data stored in the graph. Second, we assume that each vertex is queried at least once: this allows the server to reconstruct all the encrypted sketchs and learn the full sketch pattern. This is a legitimate assumption if the server is queried often by the client over a certain period of time.

With known queries. As we will see that little information is revealed by the sketch pattern only, we then assume that the server has access to a certain number of known queries between *candidate vertices* and *references vertices*. In the case of the partial distance matrix, the queries do not need to be chosen and could be the result of the attacker passively learning queries from another channel.

4.2 Building Blocks

Before we get to the attacks, we need to add two more items to our tool box. We will first adapt the sketch distance recovery to approximate the result of a query. Then, we will see how we can compare two vectors of distances and evaluate how similar they are.

4.2.1 Distance recovery

The subsequent attacks are based on the fact that the server can estimate the distance between two vertices with the sketch pattern. In the previous chapter we presented our method to estimate the distance d in each seed/distance pair in a sketch, i.e. $(w, d) \in \mathsf{Sketch}_v$, from the frequency of the seed $\mathsf{Freq}(w)$. We now use this estimate to replicate the sketch-based distance oracle quering method and approximate the result of the GRECS Query algorithm.

Algorithm 1 searches for common seed *pseudo-ids* (see Definition 3.1) in the sketch patterns of u and v and returns the distance estimate for the closest one. The estimate is multiplied by 2 since it has to account for the distance first from u to the seed and then from the seed to v.

Note that the RecoverDist function takes two lists of arguments. We use this *currying*-like notation to describe a modularisable function which can be instanciated with different parameters before being applied to the actual arguments. In this case, the distance recovery function can be evaluated independently of the underlying sketch distance estimate $\widehat{\mathsf{Dist}}$.

Algorithm 1 Distance query recovery

```
1: function RecoverDist(Dist^)(x, y)
2:     d ← ∞
3:     for h ∈ SP_x do
4:         for h' ∈ SP_y do
5:             if h = h' then
6:                 d' ← 2 · Dist^(h)
7:                 if d' < d then
8:                     d ← d'
9:     return d
```

4.2.2 Similarity metric

Later in the chapter we will need a metric to compare two vectors of distances.

Taking the distances from a vertex u in a graph to a set M of other vertices we can build a vector a. Similarly, with a vertex v we can build a vector b such that $b = (\mathsf{Dist}(v, y_1), \mathsf{Dist}(v, y_2), ...)$ for $M = (y_1, y_2, ...)$. A similarity metric Sim is a function of a and b which evaluates to 1 if they are identical, and thus probably $u = v$, and to another value $0 \leq \mathsf{Sim}(a, b) < 1$ if $a \neq b$. In our case, we will use a similarity metric to compare a vector of recovered distances with vectors of actual distances.

Among the popular similarity measures between two vectors we count the cosine similarity and the Euclidian distance. The so called cosine similarity is the cosine of the angle between the two vector and the Euclidian distance is the L2 norm[1] of the difference between the vectors. Since the recovered distance might have a scaling factor error depending on the quality of the distance estimate, the Euclidian distance would "forward" this error while the cosine similarity normalises the vectors and cancels this type of systematic error. Some preliminary experiences also showed that cosine similarity performs better, thus we choose this similarity metric.

Let θ be the angle between two vectors a and be b. Then we have

$$\cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|},$$

where $\cdot$ is the inner product. Since we are dealing with vectors of positive distances and since the coordinates which are more significant for the similarity are the smaller ones – corresponding to close vertices – we take the

[1] https://mathworld.wolfram.com/L2-Norm.html

element-wise inverse of a and b. Thus we define our similarity metric as

$$\mathsf{Sim}(a,b) = \frac{a^{-1} \cdot b^{-1}}{\|a^{-1}\|\|b^{-1}\|},$$

where $e^{-1} = (e_1^{-1}, e_2^{-1}, ...)$ for $e = (e_1, e_2, ...)$.

4.3 Graph Information Recovery

Now that we have a technique to approximate the distance between two vertices of the encrypted graph, we can construct an $n \times n$ matrix with the distance estimate of all pairs of vertices.

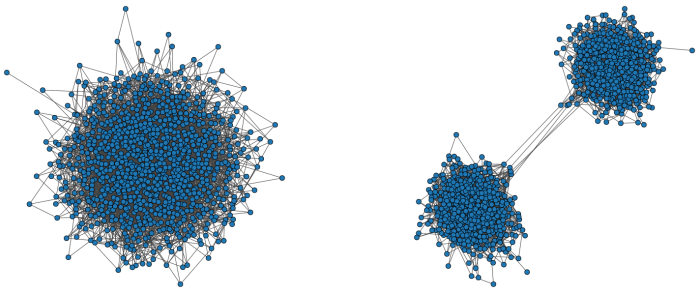
The distribution of the distance estimates allows us to approximate values like the density of the area around a vertex or the average path length in the graph. We will illustrate its usefulness by distinguishing the sketch patterns of two graphs with similar properties.

Distinguish two graphs

Let G_1 be a random graph $G_{n,p}$ with $n = 1000$ vertices and an edge probability $p = 0.008$ and let G_2 also have 1000 vertices but separated in two clusters. The edges between two vertices of a same cluster have a probability p as well, but inter-cluster edges have a probability $q = 0.00001$. Figure 4.1 shows possible instances for G_1 and G_2.

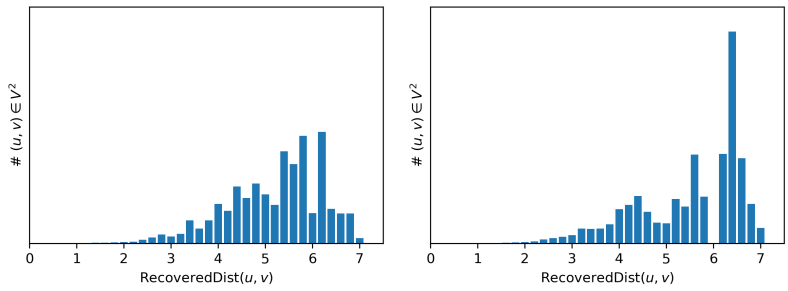
Using the distance estimate matrix, which can be computed knowing the sketch pattern and the average degree, an attacker can observe a difference in the approximation of the distance distribution. Figure 4.2 illustrates how, in the graph with two clusters, the inter-cluster distances form a seperate peak and allow the attacker to distinguish the distribution.

This kind of analysis could be performed on real-life graphs as well to recover information about the distance distribution. But we notice that we



(a) G_1 (b) G_2

Figure 4.1: Random graphs with $n = 1000$ vertices.



(a) G_1 (b) G_2

Figure 4.2: Distance estimate distribution in random graphs.

are not able to recover information about connectivity or centrality which could have lead to more significant weaknesses, thus we will now focus on an attacker which has access to more knowledge.

4.4 Query Recovery Attack

In this setting we consider an attacker trying to recover the vertices in an encrypted query. An query $q = (u, v)$ becomes $\tilde{q} = (x, y)$ once encrypted ($= P_{K_1}(u), P_{K_1}(v)$ in the case of GRECS). Thus, the attacker tries to guess which $u \in V$ corresponds to x and similarly for v and y.

We will break down the problem by first considering an attacker observing one encrypted vertex x and trying to recover to which of a set of *candidate vertices* L it corresponds. Later on we decrease the assumptions and observe how the initial attack performs.

4.4.1 Recover one vertex from l candidates

In order to distinguish the candidate vertices from each other, the attacker knows the distance from each of them to each of a set M of *reference vertices* in the form of a *distance matrix*. They can then use the distance estimate from the unknown vertex x to each of the reference vertices and build a distance estimate vector. This vector can then be compared to the rows of the distance matrix to find out to which it is most similar.

Algorithm

More formally, the attacker has

- l candidate vertices $L = (u_1, ..., u_l)$ and their unordered encryption $\widetilde{L} = \{x_1, ..., x_l\}$,
- m encrypted reference vertex identifiers $\widetilde{M} = (y_1, ..., y_m)$, and

- an $l \times m$ distance matrix A where $A_{i,j} = \mathsf{Dist}(u_i, v_j)$ for $u_i \in L$ and $v_j \in M$.

Note that the attacker does not need the actual reference vertices $M = (v_1, ..., v_m)$.

In Algorithm 2 the attacker uses the sketch pattern to build a distance estimate vector $D = (d_1, ..., d_m)$ such that $d_i = \textsc{RecoverDist}(x, y_i)$. The similarity of each row of A and the vector D is then computed. The highest similarity is the one corresponding to the best candidate, which is returned as the most probable vertex identifier for the encrypted vertex identifier x.

Algorithm 2 Vertex recovery attack

Require: $L, \widetilde{L}, \widetilde{M}, A$
1: **function** RecoverVertex(Sim)(x)
2: **for** $y_i \in \widetilde{M}$ **do**
3: $d_i \leftarrow \textsc{RecoverDist}(x, y_i)$
4: $D \leftarrow (d_1, ..., d_m)$
5: **for** $A_{i,*} \in A$ **do**
6: $s_i \leftarrow \mathsf{Sim}(A_{i,*}, D)$
7: $j \leftarrow \arg\max_i(s_1, ..., s_l)$
8: **return** L_j

Experiments

Setting. For our experiments we use the real-life datasets we introduced in Table 3.1 and sketches generated by our implementation of the Das Sarma et al. and Cohen et al. algorithms as in section 3.2.4. We run our experiments 1000 times with randomly sampled reference and candidate vertices to approximate the average performance of the recovery.

Distance estimate. In chapter 3 we developed two techniques to approximate the distances in a sketch, namely $\widehat{\mathsf{Dist}}_1$ and $\widehat{\mathsf{Dist}}_2$. After running some preliminary experiments we observe that the first estimate $\widehat{\mathsf{Dist}}_1$ performs better, as for example for the CondMat dataset in Figures 4.3a and 4.3b. This can be surprising given the results of our evaluation of the estimates (see section 3.2.4) but makes more sense considering that we only use the approximated distances as features that we compare with the similarity metric Sim and not as values close to the actual ones.

Recovery metric. Since a recovery from l candidates will in the worst case have a success rate of $\frac{1}{l}$, we substract this shift and scale it to define the *recovery advantage*.

Definition 4.1 (Recovery advantage) *The recovery advantage of a recovery attack of one among l candidates with success rate r is*

$$\mathsf{Adv}_{\mathsf{rec}} := \frac{r - \frac{1}{l}}{1 - \frac{1}{l}},$$

which is 0 when the recovery is uniform at random and 1 when it is always successful.

Results. The results of the experiments shown in Figure 4.3 indicate that from a reasonable number of reference vertices an attacker can distinguish an unknown vertex among candidates with a clear advantage. When the number l of candidate vertices increases, the probability that the unknown vertex is close to another candidate and cannot be distinguished from it becomes higher and the recovery advantage drops, as we can see in the plots.

We observe that the recovery performs better on the CondMat graph than the two others. This could be caused by the smaller size of this dataset or another property of the graph. Further investigation would be needed here.

4.4.2 Vertex recovery with restrictions

Since the knowledge required to execute the vertex recovery attack is quite extended, we investigate how the attack performs under different restrictions.

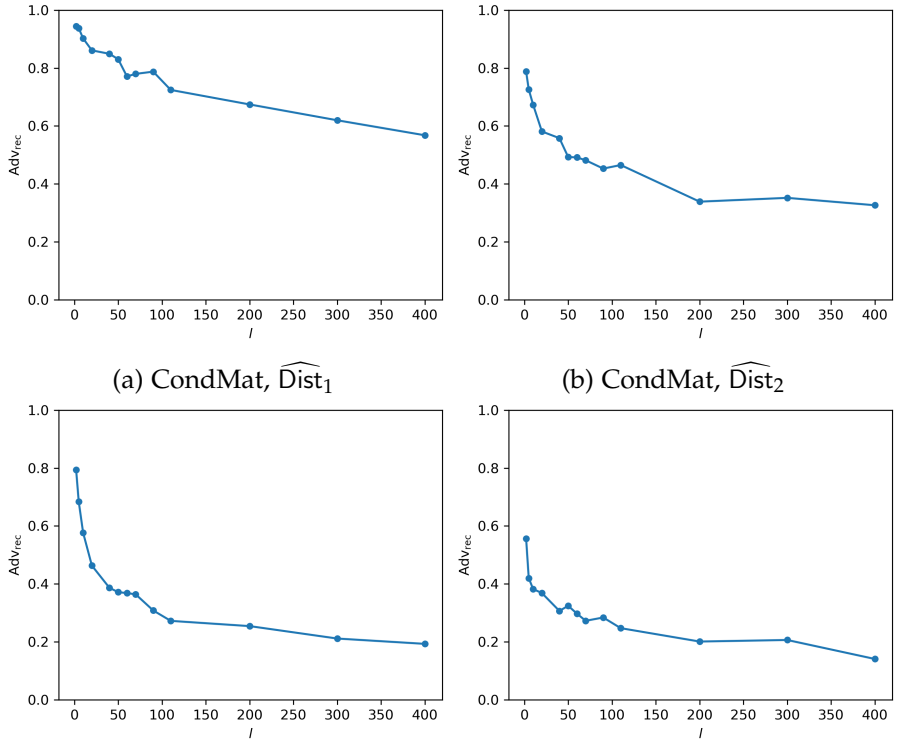
Partial distance matrix

First, we consider an attacker which cannot chose the reference vertices and only gets a certain number of query results revealed to them. This results in a distance matrix only partially filled, the rest being unknown.

As we can observe in Figure 4.4, the recovery advantage drops as expected. But we note from this experiment that not all the distances to the reference vertices are needed and more reference vertices can compensate a sparser distance matrix.

Reference vertices in different cluster

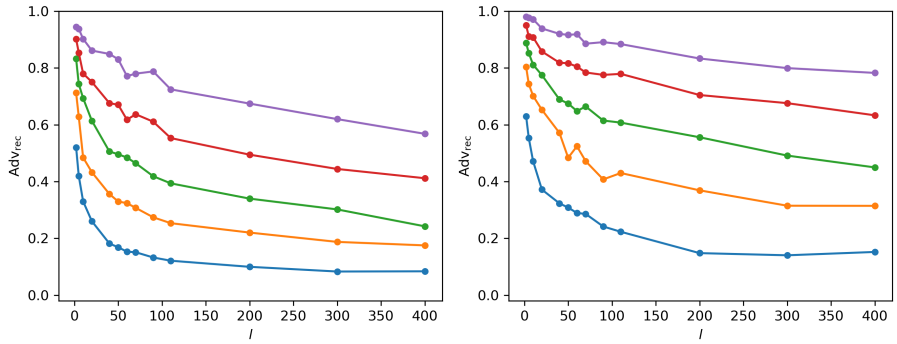
Until now we sampled the sets of vertices M and L uniformly at random. As a second restriction for the attacker we consider reference vertices in a different region of the graph than the candidate vertices. This would translate to an attacker learning the query results only from a localised set of reference vertices.



(a) CondMat, $\widehat{\text{Dist}}_1$ (b) CondMat, $\widehat{\text{Dist}}_2$

(c) Enron (d) Gowalla

Figure 4.3: Recovery of one vertex from l condidates with 50 references vertices using a Das Sarma et al. sketch. The average recovery advantage over 1000 trial is displayed.



(a) $m = 50$ (b) $m = 100$

Figure 4.4: Vertex recovery advantage with partial distance matrix. From the top: 100%, 80%, 60%, 40% and 20% of the distance matrix entries are kept.

For the experiment illustrated in Figure 4.5 we used the `community_fastgreedy` function of the `python-igraph` library implementing the method of [CNM04] to clusterise the vertices. The sets M and L are then chosen from two different clusters. We can observe that the advantage drops faster with l increasing than in the case of uniformly chosen candidate and reference vertices, even when increasing m. This means that the rows of the distance matrix are too similar and the attacker fails to distinguish the unknown vertex among too many of its neighbours. The reason why the recovery apparently performs worse with more reference vertices is unclear. A hypothesis is that the vertices making a difference in the similarity metric, i.e. probably the close ones, do not weigh enough in the similarity metric. Thus, a successful recovery would be rarer.

4.4.3 Permutation recovery

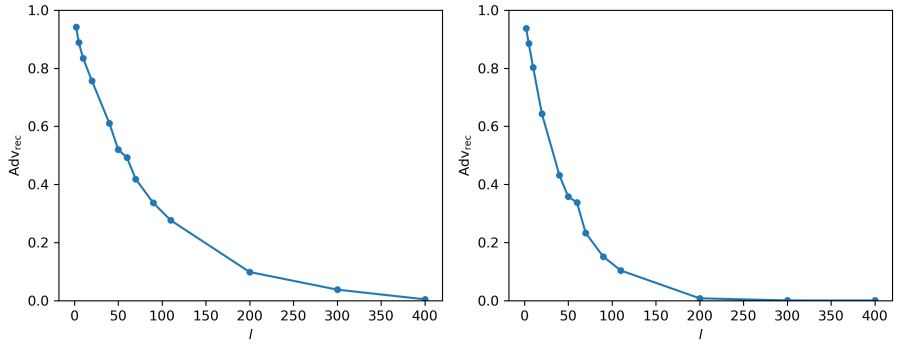
Until now the attacker was only attempting to recover one vertex among l. A more advanced setting is for the attacker to have an $l \times l$ distance matrix and no reference vertices. This means that they can compute an $l \times l$ permutated distance estimate matrix and attempt to recover the permutation.

This setting is comparable to the attack by Islam et al. in [IKK12], where they recover a queried keyword permutation from a keyword co-occurence matrix. Before trying to optimise the permuation recovery, we try a small experiment by maximising over all possible permutation with small values of l.

We can see that the results in Table 4.1 are not encouraging. The issue, in our case, might be that two vertices close to each other might have identical distances to most or all other candidate vertices and cannot be differentiated in the distance estimate matrix, even if the estimate is exact. This results in a poor recovery advantage even with small matrices.

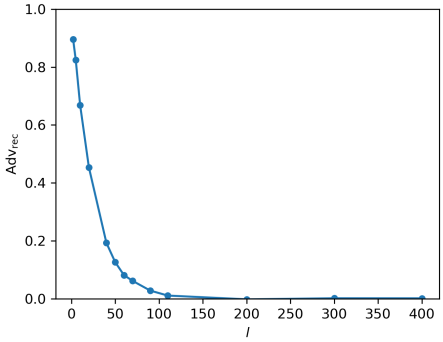
Table 4.1: Recovery rate r (proportion of the vertices recovered correctly) and corresponding recovery advantage of the permutation recovery of an $l \times l$ distance matrix. The values are averaged over 100 random subsets of the CondMat dataset.

l	r	$\mathsf{Adv}_{\mathsf{rec}}$
5	0.308	0.135
6	0.357	0.228
7	0.370	0.265
8	0.326	0.230
9	0.262	0.170



(a) $m = 50$

(b) $m = 100$



(c) $m = 200$

Figure 4.5: Recovery advantage with reference and candidate vertices in different clusters.

4.5 Countermeasures

The attacks we previously described are based on the sketch pattern leakage. In this section we will attempt to reduce the amount of information an attacker can learn with the sketch pattern to make such attacks inefficient.

4.5.1 Add fake seeds

The distance estimates we introduced in Chapter 3 is based on the frequency of a seed from Definition 3.3, which itself relies on the sketch pattern. If the seeds all had the same frequency, no information about the distance could be inferred, but then either the distance oracle would not give a good approximation anymore or the size of the sketches would be impractical. We have to find a compromise between the sketch size and the remaining leakage.

As Meng et al. mention in [MKNK15], adding fake seeds can improve the privacy of their construction. First, we are going to consider the Das Sarma et al. sketching algorithm and which seeds to add in this case.

Uniformise the seed frequency

We propose to modify the sketching process to uniformize the frequency of seeds above a certain level. Our level recovery method from section 3.2.1 should not be able to distinguish between the level of a large portion of the seeds and should thus have a lower success rate.

Algorithm 3 takes a parameter c and completes the seeds of the levels above $s = \lfloor \frac{r}{c} \rfloor$. The body of the if-statement at line 11 contains the modification to the original algorithm. k is the number of missing seed occurences to reach the same frequency as the seeds of level s. Thus, each seed w is added to the sketch of k random vertices.

Algorithm 3 Modified Das Sarma Setup including fake seeds

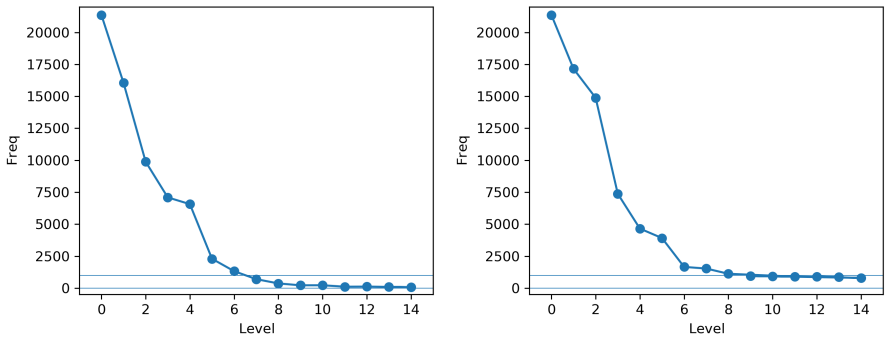
```
1:  function SETUP+(G, σ, c)
2:      for v ∈ V do
3:          Sketch_v^+ ← Map()
4:      r ← ⌊log |V|⌋
5:      s ← ⌊r/c⌋
6:      for j ∈ (1, ..., σ) do
7:          for i ∈ (0, ..., r) do
8:              S_i^j ←$ V^{2^i}
9:              for v ∈ V do
10:                 Sketch_v^+ ←add closest(v, S_i^j)
11:             if i > s then
12:                 k ← ⌊n/2^s − n/2^i⌋
13:                 for w ∈ S_i^j do
14:                     V' ←$ V^k
15:                     for v ∈ V' do
16:                         Sketch_v^+ ←add (w, Dist(v, w))
17:     return {Sketch_v^+}_{v∈V}
```

We can see in Figure 4.6b that the frequencies of the seeds have been uniformised above level 7. We will see in the next section how the attack performs on the modified sketch.

Impact on the sketch size

Without the countermeasure, each sketch has a size of $\mathcal{O}(\sigma \log n)$. We now calculate the size increase. Ignoring the σ factor, we have

$$\mathbb{E}\left(|\mathsf{Sketch}_v^+| - |\mathsf{Sketch}_v|\right) = \frac{1}{n} \sum_{i=s+1}^{r} 2^i \left(\frac{n}{2^s} - \frac{n}{2^i}\right)$$



(a) Without fake seeds. (b) With fake seeds ($c = 2$).

Figure 4.6: Average frequency of seeds for their level in the Das Sarma sketch of the CondMat dataset.

$$
\begin{aligned}
&= \mathcal{O}\left(\frac{1}{n}\sum_{i=s+1}^{r} n(2^{i-s}-1)\right) \\
&= \mathcal{O}\left(\sum_{i=1}^{r-s} 2^i\right) = \mathcal{O}\left(2^{r-s+1}-2\right) \\
&= \mathcal{O}\left(n^{\frac{c-1}{c}}\right),
\end{aligned}
$$

where the first equality is due to counting the average number of added entries per sketch in Algorithm 3 and the last one is due to $2^r \simeq n$ and $s \simeq \frac{r}{c}$.

This means for example with $c = \frac{4}{3}$ that the sketch size is now $\mathcal{O}(\sigma n^{1/4})$, which is a significant increase for a large n.

Adapting to the Cohen et al. sketches

As we uniformised the frequency of seeds above a certain level for the Das Sarma algorithm, we can modify the Cohen sketching method to uniformise the seed frequency above a certain rank. Since the procedure is very similar we leave it aside for the moment.

4.5.2 Experiments

To verify the effect of our countermeasure we generate Das Sarma sketches for some values of c and observe how the vertex recovery performs. Figure 4.7 shows the recovery advantage for values of c up to 2, $c = 1$ adding no seeds. We observe that the countermeasure effectively reduces the recovery advantage at the cost of increasing the sketch size as indicated in Table 4.2.

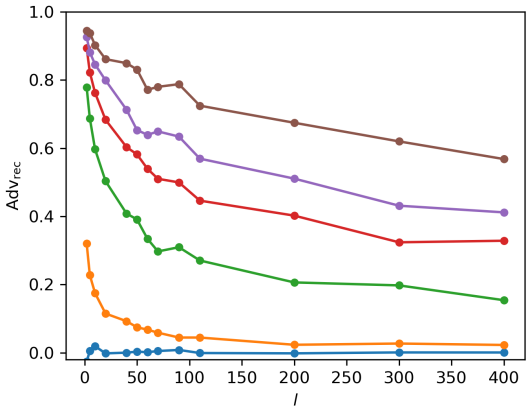


Figure 4.7: Vertex recovery with $m = 50$ on Das Sarma et al. sketches of the CondMat dataset generated with various values of the countermeasure parameter c. From the top, the values of c are: 1, $\frac{6}{5}$, $\frac{5}{4}$, $\frac{4}{3}$, $\frac{3}{2}$ and 2.

Table 4.2: Average sketch size of Das Sarma sketches ($\sigma = 3$) for the CondMat dataset.

c	Average sketch size
1	37.2
6/5	52.9
5/4	61.9
4/3	87.1
3/2	166.2
2	763.4

4.5.3 Possible improvements

Since a good performance of our countermeasure implies a large value of c and thus a high increase in the sketch size and generation time, there is room for improvement. One possibilitiy would be to randomise the seed level obfuscation and only add fake occurences for some seeds and without aiming at uniformisation. This could reduce the cost of the countermeasure while adding enough noise to the seed level to prevent it from ”betraying” the distances in the sketches.

4.6 Chapter Summary

In this chapter we described several attacks on encrypted sketch-based distance oracles which leak the sketch pattern. We started by investigating how much a honest-but-curious server can learn about the encrypted graph with

the sketch pattern only and observed that it only learns general information about the graph structure but nothing local like the importance of a node in a network. Thus, we moved on to a more powerful attacker and described an algorithm to recover queried vertices from some known queries in addition to the sketch pattern. We further showed that some contraints on the attacker, such as non-uniform or sparse known queries, reduce the efficiency of the attacks but can be worked around.

Our results show that an attacker can recover queries with a reasonable amount of additional knowledge. Although this kind of attack would be relatively hard to mount in a real-life situation, it shows that the security guarantees of schemes leaking the sketches pattern should be put in perspective of the knowledge an attack knowing some queries can gain.

In the last section we proposed a countermeasure to reduce the impact of the sketch pattern leakage by modifying the sketching algorithm to add fake seeds. This modification makes the sketch pattern ”noisy” at the cost of larger sketches and adds a paramater to control the security versus efficiency trade-off.

Chapter 5

Conclusion and Future Work

Most recently developed encrypted databases sacrifice some privacy to be efficiently queriable. Security definitions and the proofs that these schemes satisfy the definitions guarantee that only a controlled amount of information is leaked. However, there is no known technique to define exactly the impact of this leakage on the privacy of the data, thus we have to rely on cryptanalysis to investigate the importance of potential attacks exploiting the leaked information.

In this Master's thesis we tackled the analysis of the sketch pattern leakage resulting from encrypted sketch-based distance oracles. Focusing on the GRECS scheme for approximate shortest path queries from Meng et al. [MKNK15] we studied two sketching methods, namely the Das Sarma et al. and the Cohen et al. algorithms, to understand what information the sketch pattern reveals about the content of the sketches. Based on our observations we developed two distance estimates which approximate the distances in a sketch. Our experimental evaluation showed that the sketch pattern can give an advantage of about 60% on a guess using the distance distribution when estimating the distances in sketches of real-life graphs.

We further described an algorithm to recover the distance between two vertices using our distance estimate. Based on this algorithm we explored possible attacks from an honest-but-curious server on the encrypted graph database. After experimenting with different settings, we observed that an attacker with the sketch pattern only gains general information about the structure of the graph, but some additional knowledge allows the attacker to recover queried vertices. With some known queries, the server can distinguish among candidate vertices which one was queried, and this attack also works with non-uniform and sparse known queries. For example, considering the CondMat dataset, an attacker knowing 80% of the distances between 50 reference vertices and 90 candidate vertices can distinguish which of the 90 candidates was queried in more than 60% of the cases.

Our investigation of potential attacks showed that the sketch pattern, which is at first glance only a set of random vertices, can actually lead to a serious privacy breach and should not be overlooked. In response to this we proposed a countermeasure to reduce the impact of the sketch pattern leakage by modifying the underlying sketching algorithm.

Future work. Concerning future work we can mention two directions, namely improving the analysis of the sketch pattern and designing encrypted graph databases with more capabilities.

The distance estimate recovered from the sketch pattern leakage could probably be improved by taking in account topological properties of the targeted graph. Moreover, the performance of the query recovery attack could be increased by using a better method to measure the similarity between distance vectors. The attack could also be made more practical by reducing the amount of information required by the attacker.

Taking a step back, the research area of encrypted graph databases could be explored further. Schemes solving more than shortest path queries could be developed and existing schemes could be improved to support multiple users and to be dynamic.

Appendix A

Average distance in N-ball

Consider a ball B of radius R in an N-dimensional Euclidian space. Its volume and surface are, respectively,

$$V_N(R) = \frac{\pi^{\frac{1}{2}N}}{\Gamma\left(\frac{1}{2}N+1\right)} R^N,$$
$$S_N(R) = \frac{2\pi^{\frac{1}{2}N}}{\Gamma\left(\frac{1}{2}N\right)} R^{N-1}$$

[DLMF, Eq. 5.19(iii)] where $\Gamma(z)$ is Euler's integral [DLMF, Eq. 5.2(i)].

Let us denote the average distance between the center c of the N-ball B and a point sampled uniformly at random from B by $l_B(c)$. In order to compute the value of $l_B(c)$ we observe that each point on the N-sphere with radius r is at distance r from the center c, thus we integrate over the spherical layers of thickness dr forming B, each weighted with the value r, and divide the result by the volume of B. That is

$$\begin{aligned}
l_B(c) &= \frac{1}{V_N(R)} \int_0^R S_N(r) \cdot r \, dr \\
&= \frac{1}{V_N(R)} \int_0^R \frac{2\pi^{\frac{1}{2}N}}{\Gamma\left(\frac{1}{2}N\right)} r^N \, dr \\
&= \frac{1}{V_N(R)} \frac{2\pi^{\frac{1}{2}N}}{\Gamma\left(\frac{1}{2}N\right)} \left[\frac{1}{N+1} r^{N+1} + C\right]_0^R \\
&= \frac{\Gamma\left(\frac{1}{2}N+1\right)}{\pi^{\frac{1}{2}N}} R^{-N} \frac{2\pi^{\frac{1}{2}N}}{\Gamma\left(\frac{1}{2}N\right)} \frac{1}{N+1} R^{N+1} \\
&= \frac{2}{N+1} \frac{\Gamma\left(\frac{1}{2}N+1\right)}{\Gamma\left(\frac{1}{2}N\right)} R \\
&= \frac{2}{N+1} \frac{N}{2} R
\end{aligned}$$

$$= \frac{N}{N+1} R,$$

where C is a constant and where we used $\Gamma(z+1) = z\Gamma(z)$.

We can conclude that the average distance from the center of an N-ball is proportional to its radius R, and is actually close to R in high dimensional spaces.

Bibliography

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 205–222, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 563–574, New York, NY, USA, 2004. Association for Computing Machinery.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 535–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 224–241, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 506–522, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BKOS07] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith. Public key encryption that allows pir queries. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 50–67, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*, pages 253–273, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography*, pages 535–554, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[CDF+13] Edith Cohen, Daniel Delling, Fabian Fuchs, Andrew V Goldberg, Moises Goldszmidt, and Renato F Werneck. Scalable similarity estimation in social networks: Closeness, node labels, and random edge lengths. In *Proceedings of the first ACM conference on Online social networks*, pages 131–142, 2013.

[CGKO11] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 668–679, 2015.

[CJJ+13] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 353–373, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[CJJ+14] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: data structures and implementation. In *NDSS*, volume 14, pages 23–26. Citeseer, 2014.

[CK10] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT 2010*, pages 577–594. Springer Berlin Heidelberg, 2010.

[CM05] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 442–455, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[CNM04] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[CYW+11] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou. Privacy-preserving query over encrypted graph-structured data in cloud computing. In *2011 31st International Conference on Distributed Computing Systems*, pages 393–402, June 2011.

[DLMF] Nist digital library of mathematical functions. http://dlmf.nist.gov/, Release 1.0.28 of 2020-09-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

[DSGNP10] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. A sketch-based distance oracle for web-scale graphs. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 401–410, 2010.

[EHF16] Benjamin Edwards, Steven Hofmeyr, and Stephanie Forrest. Hype and heavy tails: A closer look at data breaches. *Journal of Cybersecurity*, 2(1):3–14, 12 2016.

[FFH04] Agata Fronczak, Piotr Fronczak, and Janusz A Hołyst. Average path length in random networks. *Physical Review E*, 70(5):056110, 2004.

[FJK+15] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In Günther Pernul, Peter Y A Ryan, and Edgar Weippl, editors, *Computer Security – ESORICS 2015*, pages 123–145, Cham, 2015. Springer International Publishing.

[FKS20] Nóra Frankl, Andrey Kupavskii, and Konrad J. Swanepoel. Embedding graphs in euclidean space. *Journal of Combinatorial Theory, Series A*, 171:105146, 2020.

[G+03] Eu-Jin Goh et al. Secure indexes. *IACR Cryptol. ePrint Arch.*, 2003:216, 2003.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 506–522, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[GJW19] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. Encrypted databases: New volume attacks against range queries. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 361–378, New York, NY, USA, 2019. Association for Computing Machinery.

[GLMP18] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 315–331, 2018.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Ndss*, volume 20, page 12. Citeseer, 2012.

[JA20] Ahmad H Juma'h and Yazan Alnsour. The effect of data breaches on company performance. *International Journal of Accounting & Information Management*, 2020.

[KKNO16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1329–1340, 2016.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[KM18] Seny Kamara and Tarik Moataz. Sql on structurally-encrypted databases. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 149–180, Cham, 2018. Springer International Publishing.

[KM19] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 183–213. Springer, 2019.

[KMO18] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. Structured encryption and leakage suppression. In *Annual International Cryptology Conference*, pages 339–370. Springer, 2018.

[KO12] Kaoru Kurosawa and Yasuhiro Ohtaki. Uc-secure searchable symmetric encryption. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security*, pages 285–298, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[KP13] Seny Kamara and Charalampos Papamanthou. Parallel and dynamic searchable symmetric encryption. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 258–274, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 965–976, New York, NY, USA, 2012. Association for Computing Machinery.

[KT16] Amira Kharroubi and Jamel Touir. *The Geocentric Model of the Earth: Physics and Astronomy Arguments*. PhD thesis, Department of Geography, National Engineering School of Sfax, Sfax University, 2016.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[LKF05] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible

explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[LMP18] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 297–314. IEEE, 2018.

[LW16] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1167–1178, 2016.

[LZWT14] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-an Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Inf. Sci.*, 265:176–188, 2014.

[MKNK15] Xianrui Meng, Seny Kamara, Kobbi Nissim, and George Kollios. Grecs: Graph encryption for approximate shortest distance queries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 504–517, 2015.

[PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 375–391, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[SMZ+17] Meng Shen, Baoli Ma, Liehuang Zhu, Rashid Mijumbi, Xiaojiang Du, and Jiankun Hu. Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. *IEEE Transactions on Information Forensics and Security*, 13(4):940–953, 2017.

[SPS13] E. Stefanov, Charalampos Papamanthou, and E. Shi. Practical dynamic searchable encryption with small leakage. *IACR Cryptol. ePrint Arch.*, 2013:832, 2013.

[SWP00] D Song, D Wagner, and A Perrig. Practical techniques for searching on encrypted data, s & p 2000, 2000.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, January 2005.

[vLSD+10] Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker. Computationally efficient searchable symmetric encryption. In Willem Jonker and Milan Petković, editors, *Secure Data Management*, pages 87–100, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 707–720, 2016.

[ZZX+20] Can Zhang, Liehuang Zhu, Chang Xu, Kashif Sharif, Chuan Zhang, and Ximeng Liu. Pgas: Privacy-preserving graph encryption for accurate constrained shortest distance queries. *Information Sciences*, 506:325–345, 2020.