Searchable Encryption for Sensitive Data in Social Network

A Confirmation Report submitted in fulfilment of the requirements for the candidature of

Doctor of Philosophy

By

Shangqi Lai

Supervisor: Dr. Joseph Liu

Co-supervisor: Dr. Ron Steinfeld, Dr. Dongxi Liu

Faculty of Information Technology

Monash University

20/11/2017

Abstract

Social networks are growing in leaps and bounds for many years, and more and more people use social network as their daily tool for communication and interaction with others. Using the social network service generates a huge amount of data relating to users' behaviour, preference and so on. As a result, social network providers started to make those data searchable by introducing "Graph Search" to improve the search experience of their services. However, such data always contain users' sensitive data like their name, address, photo, etc., making those data searchable leads to increasing privacy concern. This research aims to make contributions for protecting user?s privacy in the context of Graph Search. It introduces a new model to enable searchable encryption on Graph Encryption, and private search functions will be proposed and analysed to support rich search functions in above model. It also implements about search functions into privacy preserving scheme for Graph Search, and evaluation the schemes on a cloud scenario. This research improves users' privacy when using social network service and practical to deploy in the large scale distributed system..

Contents

1 Research Problem 1

1.1 Background of Problem 1

1.2 Research Questions 3

1.3 Research Scope 3

1.4 Contributions 5

2 Literature Review 6

2.1 Privacy Preserving Scheme for OSN 6

2.1.1 Decentralisation-based Solutions 6

2.1.2 Cryptographic Based Solutions 7

2.2 Graph Search 9

2.3 Searchable Encryption 10

2.3.1 Searchable Symmetric Encryption 10

2.3.2 Public Key Searchable Encryption 12

2.4 Field of Research 12

3 Objectives and Methodology 13

4 Model Construction 14

4.1 Social Graph and Modelling 14

4.2 Searchable Encryption on Graph and its index 15

4.3 Boolean Queries in Graph Encryption 17

4.4 Future Works 20

5 Progress 21

5.1 Research 21

5.2 Courseworks 23

6 Timeline to Completion 25

References 26

Chapter 1

Research Problem

1.1 Background of Problem

Online Social Network (OSN) is highly popular for many years, and more and more Internet users start to use OSN as their medium of communication: The monthly active user of Facebook is reached 2 billion in 2017, which means over one half Internet users are using the services from Facebook [1, 2], and this is just an epitome of the today's OSN dependency.

A result of the huge growth of OSN is more and more sensitive personal data is revealed while OSN users enjoy using it. In 2005, two researchers from Carnegie Mellon University conducted a study in the online behaviour of more than 4,000 students in the university who had signed up in Facebook [3]. In their study, they found that the majority of Facebook users at CMU provided astonishing amount of sensitive information in their Facebook profiles: it may contain their real names (over 90%), date of birth (87%), personal images (80%) and so on; actually, privacy preference is provided to OSN user for controlling the visibility and access privilege of their profiles, but it is sparingly used, which makes most of the users searchable and identifiable to the vast majority of anybody else in the network.

Due to the richness and variety of sensitive information in OSN, the privacy of these data is of critical importance to OSN users, because the permissive and unconcerned uses of sensitive information will put themselves at risk for many attacks from the cyberspace or even the real world: The precise personal information collecting from OSN helps potential adversaries to construct more deceitful fraud messages (e.g. Context-Aware Spam [4]); it also can be used to re-identify some anonymous datasets like hospital medical information by matching the common attributes [5]; additionally, OSN users may be easily stalked as they revealed their location or timetable when they shared their activities with their friends.

Although the main purposes of OSN are to help its members to communicate with others and maintain social relations in a more convenient way, OSN service providers also design diverse Social Network Services (SNS) for its user interaction virtually: it includes updating activities and location information, sharing multimedia (e.g. photo, video) during some events, getting updates and comments on activities by friends, etc.. All of these services retain a huge amount of data about its users: Until 2013, there were 1.11 billion monthly active users and they put 4.75 billion shared items in Facebook, these contents received over 4.5 billion tags such as 'Likes' from their friends [6].

Making these data searchable to further enhance the capabilities of OSN had become a prevailing trend in recent years. In 2013, Facebook introduced their social data search engine – Graph Search [7]. This search engine aims to return the information based on the content from user's social network of friends and connections (their social circle): For example, users may search "The restaurants visited by friends live in Melbourne" to get restaurants recommendation from their friends in Melbourne. The search results are more personally relevant and satisfactory because the results are generated and refined by users' social circle where the people within always have some similar or the same attributes (e.g. geographical location, hobbies, education background etc.) according to "Homophily" theory [8]. The concept of Graph Search is also introduced by other OSN service provider: LinkedIn has Job Search Engine which is based on users' location, skill as well as the information from their colleague and alumni; WeChat also can search user-specific content from Moments and Official Accounts.

In a nutshell, Graph Search-like search engines are powerful tools from the aspect of search because the search results are from a more reliable source (i.e. social circle) and more user-centric. However, they make user data searchable and then provide an extremely simple way to unearth user sensitive data: For instance, potential adversaries now can easily construct a query like "People in Melbourne working/traveling outsides" to find break-in crime targets. As a result, these new social data search engines ("Graph Search" is used to represent them) cause increasing privacy concerns about OSN to the public, and, to protect user's privacy in the context of Graph Search becomes the main motivation of this research.

1.2 Research Questions

In general, this research is going to answer following question:

How can we design privacy preserving algorithm to protect user's privacy in Graph Search?

in the context of today's OSN dependency and the inevitable trend of Graph Search wide deployment.

1.3 Research Scope

After providing the research question, the scope will be defined in this section. Indeed, there are multiple solutions to secure users' privacy in OSN, since the limited time and resource for this research, it will focus on solving privacy issue of Graph Search by addressing the following key technical points:

- **Cryptographic Model for Graph Data.** Cryptographic solutions are preferred in outsourced cloud service likes OSN because it can stop the privacy invasion not only from malicious adversaries, but also from untrusted service providers. However, most of cryptographic technique makes data unsearchable which contradicts to the searchable requirement of Graph Search. This research first attempts to design an privacy preserving Graph data model which can protect users' privacy by using cryptography without compromising the ability to search.

- **Search with Secure Index.** Search algorithms usually use a precomputed index. This allows keyword searches to be performed in essentially constant time with respect to the size of the database. In our privacy preserving Graph Search model, Secure index is the necessary part for the security of users' data as well as the efficiency of search. This research will seek the solution for efficient and secure index building upon unencrypted index (e.g. forwarded index, inverted index) to index encrypted graph data.
- **Search on Encrypted Graph.** The main block of this research is about performing search on Encrypted Graph. This is a special case of structured search [9]. However, comparing with the most well-studied class of structured encryption – searchable symmetric encryption (SSE), the studies of Graph Encryption and its search functions are limited.
- **Large Scale Deployment.** Deploying the privacy preserving search functions into a extensive graph to enable private Graph Search on it is the final objective of this research, a formal security analysis will be presented in advance to verify the performance of proposed algorithm in security. And the search functions will implement as privacy preserving schemes based on the search algorithms for Graph Encryption in Hadoop (Facebook use it to build their Graph Search backend [10]). Finally, the schemes will be deployed upon extensive graph data to simulate the scenario of Graph Search and an evaluation in large dataset will be presented to verify its performance in query delay, system throughput and cryptographic primitives overhead over Big Data.

As mentioned above, this research aims to design privacy preserving algorithm of Graph Search by using cryptography, it will not design novel cryptographic primitives.

1.4 Contributions

This research focuses on the research questions about the privacy issues on Graph Search and solve it by introducing cryptography based privacy preserving schemes for Graph Search with rich search functions. To facilitate cryptography under the searchable requirement of Graph Search, a new model will be designed to enable efficient search over cryptographic protected graph. Several search functions will be adapted on the basis of the model to extend the search ability on encrypted graph. A formal security analysis will be given to verify the security performance of this new model. This analysis plays an important role to justify that users' privacy in graph can be protected under this model and the search functions which are based on it. Once the new model is fully defined, it will be implemented as several privacy preserving schemes and deployed on large scale graph for evaluation in different query use-case. The evaluation serves as a efficiency test by showing the overhead from cryptography is affordable, additionally, it also shows the practicality of the schemes as it supports rich search functions.

Chapter 2

Literature Review

2.1 Privacy Preserving Scheme for OSN

Numerous works have proposed to preserve the sensitive data privacy in OSN, and two strategies are mainly applied in these works: Data Decentralisation and Cryptography.

2.1.1 Decentralisation-based Solutions

Decentralised solutions attempt to provide social services through a collection of independent nodes and users can control their data in a flexible way:

Decentralised OSN in Peer-to-Peer overlays: PeerSoN [11] and Safebook [12] build a decentralised social network in Peer-to-Peer (P2P) overlays. Users can store their data in their local device and on the devices of their trusted friends, they consist of two parts: a distributed hash table (DHT) is used provide lookup service to find users and the data they store; and the P2P network offers direct data exchange between users' devices. In these social network, users are granted full privilege to control their data as well as the communication channel over P2P network: they can choose to either establish communication via their real-life trust [12], or the encrypted channel [11].

Decentralised OSN in outsourced storage: Diaspora [13] and Confidant [14] create the decentralised social network by decoupling users's data with the SNS. Users are asked for keeping their data in some trusted server, and then, the scheme can generate some data descriptors and use them to request the SNS from existing OSN. Others can read and retrieve the data from trusted server via a valid descriptor and an appropriate access privilege. In comparison, Persona [15] allows its user keeping their data in untrusted server by using cryptographic techniques, and Cachet [16] uses the idea from P2P network to further address the performance issue in decentralised system by introducing social cache over DHT.

However, decentralised solutions without cryptography are considered as insufficient approaches for solving the privacy issue of OSN. The reason for it is because of the main purpose of these decentralised solutions is to provide access control by users' preference, and this functionality is overlap with OSN access control settings [17]. In addition, existing OSN service providers have ability to enforce the access control policy in a better way: For example, it is easier for a centralised OSN service provider to stop crawlers and makes their users' data invisible to search engines, when users set the access policy to "private". Also, some OSN give "nudge" when users intent to reveal their sensitive data [18], but a decentralised system may not work properly in above cases, because its implementation only needs to fit the protocol but does not consider such details.

In addition, We argue that Graph Search can not be deployed in decentralised system as the relations in decentralised system are distributed in different network, the quality of Graph Search service is impaired as all searches, in this case, needs to be processed by all independent nodes and client undertakes heavily aggregation after the results return to client from multiple nodes.

2.1.2 Cryptographic Based Solutions

Cryptographic solutions protect the sensitive data via encryption, these solutions are preferred in public social network because it achieve access control by making data unreadable instead of enforcing some protocols, which is considered as more reliable way to stop unauthorised access.

Index Obfuscation: NOYB [19] stores users' data with untrusted service providers, but the index of these data is obfuscated by cryptographic technique. In this system, authorised users possess a part of secret can recover the real index and personal information associated with specific users while unauthorised users only get a mixture of arbitrary personal data from the crowd. NOYB is able to protect the privacy of user profile but it doesn't work for user relations and interactions.

Attribute Based Encryption: Persona [15] uses cryptographic primitive (i.e. Attribute Based Encryption (ABE)) to enforce access control over a group of users in OSN. The ability to associate the attributes of user with a key provides a convenient way for group manager to grant different access privileges to different groups. Users within those groups use the key to access the secret key of group manager as well as his/her sensitive data, so each group's access to the sensitive data is properly restricted, which guarantees the privacy of group manager. In Persona, key revocation is not efficient because it needs to re-issue a new key to all remaining users in the group. To address this issue, Sun et al. [20] and Jahid et al. [21] achieve efficient key revocation by using broadcast encryption.

Private Social Relationship: Lockr [22] separates social relationship from OSN service providers by providing Social Attestation mechanism and Social Access Control Lists. Furthermore, Social Attestation mechanism makes it easier for key revocation as it has a explicit expire date. Another benefit for using Lockr applies proof of knowledge protocol to ensures the non-transferability of sensitive data – third parties sites cannot abuse these sensitive data because they will not receive the actual identifier of users under the protocol. Some other scheme also proposed private social relationship based on privacy homomorphisms [23] or Blind Signature [24].

Above cryptographic solutions protect users' privacy generally following two steps:

1. Encrypt data and put it into some public accessible storages.
2. Authorised users use their key to access, decrypt and use it.

However, this procedure infers the dilemma when we want to integrate OSN services with the schemes: Most of the OSN services are deployed in server side, if the data in server side is encrypted, it can't be used to support the OSN services.

2.2 Graph Search

Graph Search [7] is a typical OSN service to enrich the users' experience of using OSN. Nonetheless, it significantly improves users' search ability over user-generated data in OSN. To reach this goal, Facebook conducted a research about the social graph based on the large amount of the users' data they possessed, and they concluded that although there are billions of nodes representing the entities (e.g. users and user-generated contents like photos, pages) in social graph, the graph is very sparse because a typical node only have less than a thousand edges describing relations (e.g. friends, likes, tags) between nodes. Intuitively, it is better to represent the graph as a set of adjacency lists of users' id, and an index it by edges [10]. Hence, Graph Search service is built upon the backend index service for edges, and achieve a very low query delay (11 ms in average for finishing a query with 6 million results) for querying the entities such as friends, places, pages, etc.. Furthermore, Graph Search allows more complex queries such as graph traversal to find entities which are more than one edge away from source and it only takes 2000 ms at most for a 100 thousand results query.

Graph Search highly depends on users' data especially their relations to build its index system. Prior cryptographic solutions also have met the same dilemma that the system can not build index to support Graph Search with encrypted data.

2.3 Searchable Encryption

Graph Search makes use of users' sensitive data storing on their server to generate its indexes, Searchable Encryption (SE) enables the ability to search over encrypted data, so it can offer a potential better solution for privacy issue when users' sensitive data are used for searching in outsourced cloud.

2.3.1 Searchable Symmetric Encryption

Searchable Symmetric Encryption (SSE) is the most well-studied category of searchable encryption, most of the state-of-art SSE schemes provide secure index to guarantee its query efficiency, and have a sounded security definition.

SSE with Index: The first SSE scheme is proposed by Song et al. [25], but his construction doesn't have an index to accelerate keyword search. However, index based SSE protocols have been proposed to support secure index, each more efficient than its predecessor. The first secure encrypted index was proposed in [26], based on form of forward index, storing for each document a Bloom filter containing all the document's keywords. This allows a single document to be searched in a constant time but requires each document to be checked in turn which gains a search complexity proportional to the number of documents.

Curtmola et al. [27] are the first to propose to use inverted-index data structure, storing in a hash table for each keyword, the encrypted IDs of the documents that contain it (while hiding the number of documents matching each keyword), resulting in complexity proportional to the number of matching results, even for searching the whole document collection. However, [27] does not support multiple keyword conjunctive queries efficiently; it has complexity proportional to the number of documents matching the most frequent queried keyword. Later, [28] presented the OXT protocol, extending [27] by adding a ‘Cross-Tag Set’ (XSet) data structure, which lists hashed pairs of keywords and IDs of documents containing them, and reducing search complexity to be proportional to the number of results matching the least frequent queried keyword.

Security Definition and Leakage of SSE: The studies of SSE security definition started from [26], their definition can make sure that adversaries can not deduct document’s content from its index, but it doesn’t consider the security of search tokens (trapdoors for searching) However, Curtmola et al. [27] pointed out that the security of indexes and the security of trapdoors are inherently linked. In his definition, a SSE is secure if it reveals any information about the documents and the indexes besides the outcome and the pattern of the searches. Chase and Kamara [9] extended their definition to more general structured encryption, as SSE is only a special case of structured encryption search.

Oblivious RAM (ORAM) allows SSE with no leakage [29]. However, it requires communication rounds and search complexity poly-logarithmic in the database size.To balance the efficiency and security of SSE, almost all of the practical schemes leak information about documents based on current security definition.

The notion of Graph Encryption was introduced in [9] as a special case of structured encryption. But not like SSE, which is also a sub-category of structured encryption search, the researches about Searchable Graph Encryption are very limited. In [9], several constructions were proposed to answer Adjacency Queries (give two nodes, do they have a common edge?), Neighbour Queries (get all nodes will a common edge) and Labeled Graph Queries (search graph with some labels in nodes) in encrypted graph. It is obvious that above search functions can not enable the rich functionality on Graph Encryption, and then not suit for Graph Search service.

2.3.2 Public Key Searchable Encryption

SE protocols have also been studied extensively in the public key setting. Such protocols allow any user with the public key to insert data but only allow the user with the private key to search. The use of public key cryptography makes the protocols less efficient than SSE, but allows more powerful functionality and/or better security properties. The first such protocol was proposed by [30] as a generalisation of anonymous Identity-Based Encryption (IBE), and supporting equality queries. It was significantly further generalised in [31] to HVE, applied to conjunctive, subset and range searchable encryption queries. However, public key setting is not suit for Graph Search, because the graph is generated by a sole information source (OSN provider), and the source is untrusted, it is meaningless to let an untrusted party to do encryption and broadcast the data.

2.4 Field of Research

As a conclusion of literature review, The privacy issue in OSN and its solutions are studied for many years, but none of them fit the context of Graph Search due to its searchable requirement. Since the lack of work about Searchable Encryption on Graph, this research will try to apply the concept of Searchable Encryption to define a Searchable Graph Encryption model. The new privacy preserving scheme for Graph Search will be designed on the basis of above model, and its performance of security will be further investigated as well as the efficient on large scale graph in distributed environment.

Chapter 3

Objectives and Methodology

As mentioned in the literature review, this research plans to contribute in the field of Searchable Graph Encryption by proposing new Graph Encryption scheme with rich search functions, and protect OSN users' privacy in Graph Search by using the above scheme. The detailed objectives and the methodologies are summarised as follows:

1. **Searchable Graph Encryption Model:**
    - Abstract the graph model from social graph
    - Induct cryptography into the graph model
    - Apply secure index on encrypted graph structure
2. **Search Functions for Graph Encryption:**
    - Support boolean queries in Graph Encryption
    - Support complex operators in Graph Search
    - Security notation and analysis of search functions
3. **Deployment on Large Scale Graph**
    - Implement the search functions as privacy preserving schemes
    - Deploy scheme into large scale graph to simulate Graph Search
    - Evaluate the efficiency and usability of simulated Graph Search via our scheme

The methodologies of this research are basically the abstraction and modelling from real world, inducting and developing new concepts and protocols based on the model, implementing and evaluating the model in the real world cases.

Chapter 4

Model Construction

This chapter introduces a model based on Social Graph, which is the underlying structure to support Graph Search, the searchable encryption approaches and secure index for this model will be further discussed. In the last part of this chapter, a boolean query function will be presented on this model.

4.1 Social Graph and Modelling

As mentioned in [10], Social graph is a directed graph consists of nodes which are describing the entities (users and user-generated contents like photos, pages) and edges as the relationships in social network. Social graph is a huge graph which contains many billions of nodes, however it is also a sparse graph, because most of the nodes only have no more than a thousand links between other nodes, while only a tiny fraction of nodes can attract tens of millions of links from others. As a result, in Unicorn [10], the social graph can be represented by adjacent lists, and it stores as key-value pairs for query.

In addition, there are various relationships in a social graph, the most common relationship is "friends", and a typical user may have up to a hundred friends. The interaction between nodes generates more relationships: For example, users can give a positive feedback to a restaurant by using "likes" on their page, it creates the "likes" relationship between user and page. In conclusion, the edges in social graph are labeled like a weighted graph, but a descriptor of relationship is used instead of a numeric weight. The relation label on edge can help to further reduce the size of adjacent lists. Moreover, the adjacent lists can be indexed not only by the node identifier but also the relationship type between its neighbours, such adjacent lists enable the content search in user's social network.

The generalised social graph model is defined on the basis of above facts. Formally, social graph is a edge-labeled and directed graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ and $E = \{e_1, e_2, ..., e_m\}$, the total number of nodes is $n = |V|$, and the number of edges is $m = |E|$. There are several relationships R associated with edges in social graph, all edges in G can be represented as a triad $e = (u, v, r)$ which consists of its egress, ingress nodes (u and v) plus a relationship $r \in R$ as its label. After introducing the notation of graph, the database based on it can be presented as follows: The social graph database $\mathbf{DB} = (ind_i, V_{ind_i})_{i=0}^{d}$ is an adjacent lists indexed by $ind = (u, r)$ which consists of egress nodes u and a relationship r, the size of the database is proportional to the number of nodes $d \propto n$. A query takes an index ind as input and returns a node list $V_{ind} \subseteq V$ subjects to the conditions $\{v \in V_{ind} | \exists e \in E \wedge e = (u, v, r)\}$.The complexity of such a query can be considered as a constant because $\mathbf{DB}$ is based on adjacent lists. The boolean queries on this database can take multiple indexes $ind_1, ind_2, ..., ind_t$ as input and get conjunctive result by $\mathbf{DB}(ind_1) \cap \mathbf{DB}(ind_) \cap ... \cap \mathbf{DB}(ind_t)$, the disjunctive queries can be done in the same way.

4.2 Searchable Encryption on Graph and its index

The searchable encryption notion can be applied to the Social Graph model by introducing the concept upon the $\mathbf{DB}$. Let λ be the security parameter, the searchable encryption scheme Π consists of two algorithms:

- $\mathbf{EDBSetup}(1^\lambda, \mathbf{DB}) \rightarrow (mk, param, \mathbf{EDB})$: Run in client side, takes $1^\lambda$and $\mathbf{DB}$ as inputs and returns the system master key mk, public parameters $param$ and encrypted database $\mathbf{EDB}$. mk is kept by client. $param$ is publicly known and $\mathbf{EDB}$ is stored in the server.
- $\mathbf{EDBSearch}(mk, param, ind, \mathbf{EDB}) \rightarrow (\mathbf{DB}(ind))$: $\mathbf{EDBSearch}$ runs

between client and server as a protocol. Client?s inputs are mk, $param$, and query over ind, while server's inputs are $param$ and $\mathbf{EDB}$. At the end of the protocol, client outputs node lists matching query over ind, and server outputs nothing.

An overview of Search Encryption System on Graph is shown in Fig. 4.1. The system involves two parties: the user and the server in untrusted cloud. During the $\mathbf{EDBSetup}$ phase, User can establish relationships with other users and upload these relationships into cloud server in an encrypted form. And in $\mathbf{EDBSearch}$ phase, user generates a search token related to the index $ind = (u, r)$ and send it to the cloud server, the server runs $\mathbf{EDBSearch}$ to retrieve requested results from encrypted Graph data without decrypt it and send it back to client. User can decrypt it on his/her local device.

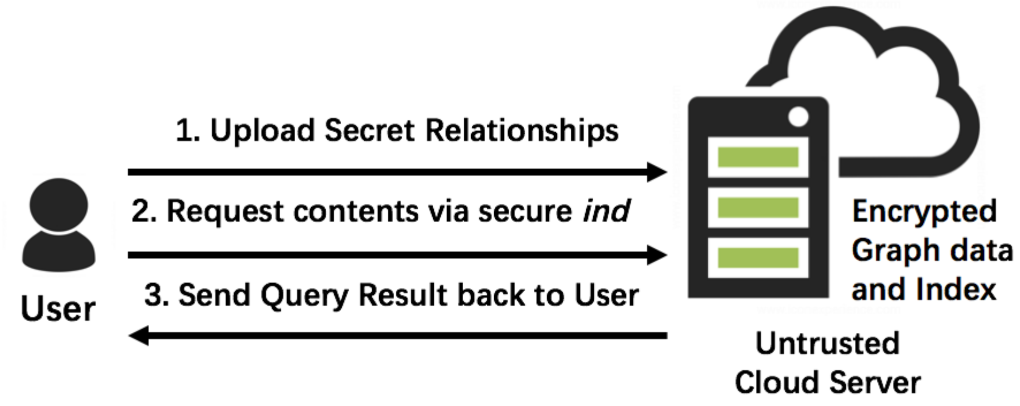
1. Upload Secret Relationships
2. Request contents via secure *ind*
3. Send Query Result back to User
User
Encrypted Graph data and Index
Untrusted Cloud Server

Figure 4.1: System Architecture Overview

Because of the index structure of $\mathbf{DB}$ is a typical inverted index. T-set can be used for efficient index on it. A T-set is an expanded inverted-index data structure [28] used for efficient SSE. It is a cryptographic data structure that associates a list of fixed-length data tuples to each keyword in a database. Later it enables the owner to issue corresponding tokens to retrieve these lists related to the queried keywords. A syntax, a correctness definition, a security model, and an instantiation of such a hash table is given in [28]. In the context of graph database, ind can be used as the keyword of this database, and a T-set can be set to provide secure index for single keyword search in encrypted graph database.

4.3 Boolean Queries in Graph Encryption

Boolean queries are a fundamental functions for a database. In Graph database, boolean queries are used to get nodes with two or more relationships, and such queries are ubiquitous in our daily lives, for example, to get restaurants from friends, the query "The restaurants visited by friends live in Melbourne" is actually the composition of query "Friends live in Melbourne" and "The restaurants visited by friends". Furthermore, boolean queries also serve as the basis of many other rich type queries. Therefore, it is also necessary to support boolean queries in Graph Encryption because it is widely use and its importance to other queries.

A naive solution is to perform single keyword search on encrypted database multiple times, and let server or client to do intersections by theirselves. [28] pointed out that this solution is not efficient and has a significant leakage that leaks the sets of documents matching each keyword. There are some SSE constructions that only leaks the matching documents [27] but it is not efficient in large database, because the conjunctive queries require to do $O(d)$ work on server side, where d is the number of documents or records in the database (many billions in social graph).

To address the efficient and security issues, Cash et al. proposed 'OXT' protocol, which also can be applied in graph encryption. The protocol is based on T-set and the concept of "Oblivious tag". In particular, the client sends the server a 'search token' (called a $\mathtt{stag}$) related to the index ind_1 (called the 's-term' and denoted by $\mathbf{sterm}$), which allows the server to retrieve from the $\mathbf{TSet}$, the set $\mathbf{DB}(ind_1)$ of all nodes. In addition, the client sends 'intersection tokens' (called $\mathbf{xtrap}$) related to the $n-1$ index *pairs*

$(index_1, index_i)$ consisting of the 's-term' paired with each of the remaining query index ind_i, $2 \leq i \leq n$ (called 'x-terms'). The intersection tokens allow the server to *filter* the set $\mathbf{DB}(ind_1)$ to determine the $n-1$ subsets of nodes $\mathbf{DB}(ind_1) \cap \mathbf{DB}(ind_i)$ that contains the pairs (ind_1, ind_i), returning only those nodes that contain all $\{ind_i\}_{1 \leq i \leq n}$. The intersection subsets $\mathbf{DB}(ind_1) \cap \mathbf{DB}(ind_i)$ are efficiently computed by the server using the '$\mathbf{XSet}$' data structure; the '$\mathbf{XSet}$' is in essence a list of hashed pairs $h(ind, \mathbf{id_v})$, over all database node identities $\mathbf{id_v}$ and index ind contained in $\mathbf{id_v}$, where h is a certain (public) cryptographic hash function. To filter $\mathbf{DB}(ind_1)$ to compute $\mathbf{DB}(ind_1) \cap \mathbf{DB}(ind_i)$, the server runs through each node identifier $\mathbf{id_v}$ in $\mathbf{DB}(ind_1)$ and checks, using the $\mathbf{xtrap}$ token for (ind_1, ind_i) whether $h(ind_i, \mathbf{id_v})$ is in the $\mathbf{XSet}$. Therefore, the server computation time is dominated by $|\mathbf{DB}(ind_1)|(n-1)$ evaluations of h, which is proportional to just the number of nodes containing the *least frequent* 's-term' ind_1, even if other 'xterm' indexes are much more common.

The ideal searchable encryption only leaks Whole Result Pattern (WRP) in this result pattern (i.e. the number (and possibly also, identities) of documents matching all query keywords). However, in OXT protocol has leakages other than WRP in boolean query, because it has Keyword-Pair Result Pattern (KPRP): for an n index conjunction query $ind_1 \wedge \cdots \wedge ind_n$, with ind_1 designated as the 's-term' index, the KPRP reveals to the server the set $\mathbf{DB}(ind_1) \cap \mathbf{DB}(ind_i)$ of documents containing every *pair* of query keywords of the form (ind_1, ind_i), $2 \leq i \leq n$.

The new construction from this research named HXT eliminates KPRP leakage in state-of-art OXT protocol while leaves other query and result pattern leakage components in existing SSE protocols unchanged. Thus, in terms of security, our protocol offers strictly better guarantees than OXT protocol. Significantly, this improved security is obtained while preserving, within moderate constant factors, the practical high performance, in terms of server computation complexity, and storage, offered by OXT protocol.

To remove the KPRP leakage of OXT, a main idea of our protocol HXT is the use of HVE to implement a type of *ciphertext policy* searchable encryption supporting subset membership predicates. Our ciphertext policy protocol is a kind of 'dual' to the HVE-based *key policy* searchable encryption protocol for subset membership predicates presented in [31]. In our HVE-based scheme, a dataset owner can encrypt a set $S \subseteq T = \{1, \ldots, n\}$, for some positive integer n, into a ciphertext c_S, which specifies a 'policy'. Using a master secret key msk, the owner can generate a search token $tk_{S'}$ for any subset $S' = \{s'_1, \ldots, s'_\ell\}$ of T. Using the token $tk_{S'}$ for S' and the ciphertext c_S for S, anyone can efficiently check whether $S' \subseteq S$ or not, without leaking any partial information if $S' \nsubseteq S$, e.g. whether any particular element s'_i of S' is in S or not (note that in the scheme of [31], the set S is used to generate the token, while the set S' is encrypted in the ciphertext). Given this subset membership searchable encryption protocol, a natural idea to apply it to eliminate KPRP leakage in OXT would be to use it to *encrypt the* '$\mathbf{XSet}$' during set up: we let $S \subseteq T$ denote the $\mathbf{XSet}$ list of hashed pairs $h(ind, \mathbf{id}_\mathrm{v})$, over all node identifiers $\mathbf{id}_\mathrm{v}$ and indexes ind contained in $\mathbf{id}_\mathrm{v}$, and we encrypt S into a ciphertext c_S stored on the server using our HVE-based subset searchable encryption scheme. In the search phase with query $ind_1 \wedge \cdots \wedge ind_n$, the client issues the server a HVE search token $tk_{S'}$ for $S' = \{h(ind_i, \mathbf{id}_\mathrm{v})\}_{i=2}^n$, $\mathbf{id}_\mathrm{v} \in \mathbf{DB}(ind_1)$. This allows the client to check whether $S' \subseteq S$, i.e., whether $\mathbf{id}_\mathrm{v}$ contains all n indexes $\{w_i\}_{1 \leq i \leq n}$, without revealing the KPRP information on whether $\mathbf{id}_\mathrm{v}$ contains any particular pair (w_1, w_i).

There are two main issues with the above basic idea: the first one is the storage overhead. HXT introduces a small (but negligible) false positive probability P_e to use the classical notion of a Bloom filter [32] reduce the storage size of $\mathbf{XSet}$ to $m \approx 2\log_2(1/P_e)N$, m is the number of bits in

Bloom Filter, and N is number of items in the filter. Another issue is the performance of HVE, because most of the construction of HVE has exponentiation and pairings during instantiation, which are not efficient when msk are built over billions possible attributes (it happened in Bloom Filter). we proposed a new symmetric HVE without pairings to address this issue.

4.4 Future Works

In this chapter, a general model for social graph and the searchable encryption notation for the proposed model are established. Based on the model, some existing definitions and tools from SSE can be adapted to support the secure index on graph encryption. By applying these definitions and tools, this research designed a secure and efficient boolean query protocol for graph encryption. So far, I have finished the modelling of Searchable Graph Encryption and fundamental queries on it, this research will focus on more complex search functions that support "apply" and "extract" queries, which are important components for Graph Search service, and the formal security defines and proof in future.

Chapter 5

Progress

5.1 Research

In the first year of my PhD candidature, I mainly focus on the a SSE scheme that supporting private and efficient boolean query. I and my supervising team design this new SSE scheme named HXT together. The new scheme is based on the 'OXT' protocol [28], but its security is further improved by removing a result pattern leakage that leaks the matching pairs of queried keywords. To achieve this, we design a new Hidden Vector Encryption (HVE) scheme without pairing to improve the performance of HVE significantly, and we embedded the new HVE with OXT to hide the result pattern leakage.

A more detailed but concise technical description of our HXT is presented here:

The HXT SSE protocol uses the following primitives (for security parameter λ):

- A cyclic group $\mathbf{G}$ with prime order p and generator g, for which the **DDH** assumption holds.
- A Hidden Vector Encryption scheme **HVE**.
- A symmetric key encryption scheme **Sym** with key space 1^λ.
- A Bloom filter **BF** with length m and k functions.
- Psuedorandom functions (PRFs) F with range 1^λ, F_p with range $\mathbf{Z}_p^*$ and F_j with range $[m]$ for $j = 1, \ldots, k$.

Likes other SSE schemes, HXT protocol consists of two algorithms:

$\mathbf{EDBSetup}$ to create encrypted database and $\mathbf{EDBSearch}$ to query the result from EDB.

The setup algorithm $\mathbf{EDBSetup}$ gets the security parameter λ and returns mk, $param$, and $\mathbf{EDB}$. The encrypted database $\mathbf{EDB}$ has two components: $\mathbf{EDB}(1)$ is $\mathbf{HVE}$ generated exactly as in OXT, and $\mathbf{EDB}(2)$ is an $\mathbf{XSet}$ encryption of a carefully designed Bloom filter $\mathbf{BF}$ set up for $\mathbf{XSet}$ elements of the form $h(\mathbf{id}, w) = g^{F_p(\kappa_X, w)\cdot \mathbf{xid}}$, for encrypted identifiers $\mathbf{xid} = F_p(\kappa_I, \mathbf{id})$ over all $\mathbf{id} \in \mathbf{DB}(w)$. In particular, this algorithm writes 1's into $\mathbf{BF}$ in positions in set

$$S = \{h_j(\kappa_j, h(\mathbf{id}, w))\}_{1\leq j\leq k},$$

over all $(\mathbf{id}, w)$ pairs with $\mathbf{id} \in \mathbf{DB}(w)$, and then encrypts $\mathbf{BF}$ with $\mathbf{HVE}$.

Similar to OXT, The search protocol generates $\mathbf{stag}$ and $\mathbf{xtoken}$ first. The $\mathbf{XSet}$ membership test for conjunctions in OXT is replaced by a $\mathbf{DB}$ token generation and query. Namely, the $\mathbf{HVE}$ token $\mathbf{token}_c$ is generated for a predicate ($\mathbf{BF}$) vector $\mathbf{v}_c$ with 1's in positions in set

$$S' = \{h_j(\kappa_j, h(\mathbf{id}_c, w_i))\}_{1\leq j\leq k}$$

over all $\mathbf{id}_c \in \mathbf{DB}(w_1)$ and $\mathbf{xterms}$ w_i for $2 \leq i \leq n$, and wild-cards in other positions. Consequently (as the message encrypted by $\mathbf{HVE}$ was set to 'True' in $\mathbf{EDBSetup}$) the $\mathbf{HVE}.\mathbf{Query}$ returns 'True' if $S' \subseteq S$, i.e. if *all* $n-1$ $\mathbf{xterms}$ w_i are in the document $\mathbf{id}_c$. Otherwise, $\mathbf{HVE}.\mathbf{Query}$ returns $\perp$, without revealing $\mathbf{KPRP}$ information on which w_i are in $\mathbf{id}_c$. Importantly, the $\mathbf{HVE}.\mathbf{Query}$ algorithm only uses components of $\mathbf{c}$ in the non wild-card positions of $\mathbf{v}_c$ and $\mathbf{token}_c$, so search run-time is only proportional to $|\mathbf{DB}(w_1)| \cdot n \cdot k$ (similar to OXT), and not to the size m of the $\mathbf{BF}$.

Currently the implementation of this scheme in Hadoop is finished and a comparable dataset with OXT is tested in HXT. The result shows that by adopting the in-memory distributed computing framework and distributed database (Spark [33] and HBase [34]), the new scheme can perform efficient queries over encrypted database and the overhead of our HVE is almost negligible because it is only use symmetric encryption primitives.

The paper of HXT is writing in progress, we will test the HXT protocol in a larger dataset as a supplementary result for its efficiency in large scale database. We plan to submit this paper to IEEE S&P by 1st Dec.

HXT is a more secure SSE protocol that support boolean query functions, and it is also indexed by inverted index. Hence, HXT can serve as a building block of my Searchable Graph Encryption by providing secure and efficient boolean query functions.

Course Code	Course Name	Result
FIT5143	**IT Research Methods**	**Exempted**
FIT6021	**Advanced Research Methods**	**C**
	Argument Analysis	HD
	Problem Solving and Proof	D
	Scientific Method	HD
	Modelling and Computer Simulation	Attended but Missing
	Design Research	D
FIT5144	**Research strategies and skills**	**99.5 hours**
FIT5144	MGE Monash Graduate Research Student Induction	4 hours
FIT5144	MGE Research Integrity	12 hours
FIT5144	Faculty of IT Induction	6 hours
FIT5144	Communicating Research	30 hours
FIT5144	Ethical Research and Publications	6 hours
FIT5144	Data61 Reading Group	20 hours
FIT5144	GSAS Workshops	17 hours
FIT5144	HDR Student Consultation Forum with Dean of IT	1.5 hours
FIT5144	Staff Appointment Seminar	3 hours

Table 5.1: Coursework Activities and Claimed Hours

5.2 Courseworks

To meet the coursework requirements, I've get exemption for one compulsory module and attend another one as well as the compulsory events and workshops in FIT 5144 in the first year. However, there are some issues with my grades in FIT6021 module “Modelling and Computer Simulation”. Table 5.1 shows the detailed information about my coursework activities.

Apart from the compulsory 58 hours’ events and workshops listed above, I also attend GSAS Oral Presentations Workshops, Preparing for Confirmation Workshops; The Data61 reading group; Dean’s Seminar Series; Staff Appointment Seminar in FIT as elective activities to fulfil the requirement of 120 hours activities attending record for FIT5144.

Chapter 6

Timeline to Completion

Title of Activity	Timelines																																				
	2016		2017												2018												2019										
	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11
Literature Review	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																								
HVE Scheme					▨	▨	▨	▨	▨	▨	▨	▨																									
System Modeling							▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																				
Conference Paper													▨	▨	▨																						
Journal Article																▨	▨	▨																			
Rich search function																		▨	▨	▨	▨	▨	▨	▨	▨	▨											
Conference Paper																						▨	▨	▨													
Large Scale Test																									▨	▨	▨	▨	▨	▨	▨	▨	▨				
Data Collection																									▨												
Function Analysis																													▨	▨	▨	▨	▨				
Conference Paper																																	▨	▨	▨		
Thesis																									▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

References

[1] Statista. Number of monthly active facebook users worldwide as of 3rd quarter 2017.
`https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/` [online], 2017.

[2] Internet World Stats. World internet usage and population statistics.
`http://www.internetworldstats.com/stats.htm` [online], 2017.

[3] Ralph Gross and Alessandro Acquisti. Information Revelation and Privacy in Online Social Networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005.

[4] Garrett Brown, Travis Howe, Micheal Ihbe, Atul Prakash, and Kevin Borders. Social Networks and Context-Aware Spam. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 403–412. ACM, 2008.

[5] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[6] Facebook. Facebook's Growth In The Past Year.
`https://www.facebook.com/media/set/?set=a.10151908376636729.1073741825.20531316728&type=3theater` [online], 2013.

[7] Facebook. Facebook Graph Search.
`https://www.facebook.com/graphsearcher/` [online], 2013.

[8] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of A Feather: Homophily in Social Networks. *Annual review of sociology*, 27(1):415–444, 2001.

[9] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer, 2010.

[10] Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar, et al. Unicorn: A System for Searching the Social Graph. *Proceedings of the VLDB Endowment*, 6(11):1150–1161, 2013.

[11] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52. ACM, 2009.

[12] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: a Privacy Preserving Online Social Network Leveraging on Real-Life Trust. *IEEE Communications Magazine*, 47(12), 2009.

[13] Diaspora Project. diaspora*.
`https://diasporafoundation.org` [online].

[14] Dongtao Liu, Amre Shakimov, Ramón Cáceres, Alexander Varshavsky, and Landon P Cox. Confidant: Protecting OSN Data without Locking It

Up. In *Proceedings of the 12th International Middleware Conference*, pages 60–79. International Federation for Information Processing, 2011.

[15] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: An Online Social Network with User-Defined Privacy. In *ACM SIGCOMM Computer Communication Review*, pages 135–146. ACM, 2009.

[16] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 337–348. ACM, 2012.

[17] Arvind Narayanan, Vincent Toubiana, Solon Barocas, Helen Nissenbaum, and Dan Boneh. A Critical Look at Decentralized Personal Data Architectures. *arXiv preprint arXiv:1202.4503*, 2012.

[18] Dyian Mori. Privacy nudges protect information. `http://thetartan.org/2010/3/22/scitech/privacynudges` [online], May 2010.

[19] Saikat Guha, Kevin Tang, and Paul Francis. NOYB: Privacy in Online Social Networks. In *Proceedings of the first workshop on Online social networks*, pages 49–54. ACM, 2008.

[20] Jinyuan Sun, Xiaoyan Zhu, and Yuguang Fang. A Privacy-Preserving Scheme for Online Social Networks with Efficient Revocation. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[21] Sonia Jahid, Prateek Mittal, and Nikita Borisov. EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 411–415. ACM, 2011.

[22] Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: Better Privacy for Social Networks. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pages 169–180. ACM, 2009.

[23] Josep Domingo-Ferrer, Alexandre Viejo, Francesc Sebé, and Úrsula González-Nicolás. Privacy homomorphisms for social networks with private relationships. *Computer Networks*, 52(15):3007–3016, 2008.

[24] Emiliano De Cristofaro, Claudio Soriente, Gene Tsudik, and Andrew Williams. Hummingbird: Privacy at the time of twitter. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 285–299. IEEE, 2012.

[25] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *Security and Privacy, 2000. S&amp;P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[26] Eu-Jin Goh et al. Secure Indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[27] 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.

[28] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-Scalable Searchable

Symmetric Encryption with Support for Boolean Queries. In *Advances in cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.

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

[30] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In *Eurocrypt*, volume 3027, pages 506–522. Springer, 2004.

[31] Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. *Theory of cryptography*, pages 535–554, 2007.

[32] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[33] Apache. Apache Spark^{TM} - Lightning-Fast Cluster Computing. `https://spark.apache.org` [online], 2017.

[34] Apache. Apache hbase – apache hbase^{TM} home. `https://hbase.apache.org` [online], 2017.