Verifiable Social Data Outsourcing

Xin Yao*†, Rui Zhang‡, Yanchao Zhang†, Yaping Lin*
*Hunan University, Changsha, Hunan, China
†Arizona State University, Tempe, AZ, USA
‡University of Delaware, Newark, DE, USA
Email: xinyao@hnu.edu.cn, ruizhang@udel.edu, yczhang@asu.edu, yplin@hnu.edu.cn

***Abstract*—*Social data outsourcing* is an emerging paradigm for effective and efficient access to the social data. In such a system, a third-party Social Data Provider (SDP) purchases complete social datasets from Online Social Network (OSN) operators and then resells them to data consumers who can be any individuals or entities desiring the complete social data satisfying some criteria. The SDP cannot be fully trusted and may return wrong query results to data consumers by adding fake data and deleting/modifying true data in favor of the businesses willing to pay. In this paper, we initiate the study on *verifiable social data outsourcing* whereby a data consumer can verify the trustworthiness of the social data returned by the SDP. We propose three schemes for verifiable queries over outsourced social data. The three schemes all require the OSN provider to generate some cryptographic auxiliary information, based on which the SDP can construct a verification object for the data consumer to verify the query-result trustworthiness. They differ in how the auxiliary information is generated and how the verification object is constructed and verified. Extensive experiments based on a real Twitter dataset confirm the high efficacy and efficiency of our schemes.**

I. Introduction

Online social networks (OSNs) are pervasive. As three exemplary popular OSNs, Twitter, Facebook, and Sina Weibo have 310 million, 1.65 billion, and 261 million monthly active users as of the first quarter of 2016, respectively. OSN users produce information at an unprecedented rate and scale. For example, there are about 3 million posts per minute on Facebook and 500 million tweets on Twitter per day. Besides facilitating social interactions, OSNs are increasingly used in massive information campaigns, public relations, political campaigns, pandemic and crisis situations, marketing, and many other public/private contexts. For instance, over 76% of businesses used social media to achieve their marketing objectives in 2014; business retailers have seen 133% increases in their revenues from social media marketing; about 71% of the consumers respond to the feedbacks and recommendations of social users regarding a particular product [1].

The traditional way to access the social data is via the public APIs provided by each OSN itself, but the data obtained in this way are often incomplete, biased, and even incorrect. For example, Twitter provides the Filter API and Sample API to access real-time tweets as well as the Search API to retrieve historical tweets. These public APIs, however, often have very limited functionalities. For example, the Filter API and Sample API both return at most 1% random samples of all the data satisfying the query condition [2]; the data returned by the Filter API have been found strongly biased [2]; and the crawling process of the Sample API can be easily manipulated by attackers wishing to promote their social content [3]. In addition, the Search API searches against a sampling of recent tweets published in the past 7 days. There are also rate limits imposed on these public APIs. For instance, no more than 180 calls per user and 450 calls per application for the Search API can be made every 15 minutes. While the Twitter Firehose API yields 100% of all public tweets, it incurs a prohibitive monetary cost and very high server requirements to host and process the real-time tweets. The public APIs of other OSNs such as Facebook have similar constraints as well.

Social data outsourcing is an emerging paradigm for effective and efficient access to social data. A system built on this paradigm consists of a third-party Social Data Provider (SDP), many OSN operators, and numerous data consumers. The SDP purchases complete data from OSN operators and offers paid data services to data consumers who can be any individuals or entities requiring the complete social data satisfying some criteria. Some popular SDPs include DataSift, Gnip, NTT Data, Brandwatch, Twazzup, CrowdEye, etc. For example, DataSift has 20 data sources until now such as Facebook, Instagram, Youtube, Tumblr, and Google+.

How much trust could a data consumer have in the data offered by these SDPs? There have been too many stories revealing the dark side of the web industry. For instance, Yelp has "always had a complicated relationship with small businesses" according to BusinessWeek [4], and there have been wide allegations that Yelp has manipulated the online reviews based on the participation in its advertising programs [5]. As another famous example, the "death of Wei Zexi" has resulted in the official investigation of Baidu—the biggest search engine in China—which has been notoriously providing highly skewed search results due to paid placement that are unaware by the Internet users. In our context, a dishonest SDP can return wrong query results to data consumers by adding fake data and deleting/modifying true data in favor of the businesses who would like to pay. An honest SDP may also be hacked to return wrong query results.

In this paper, we coin the concept of *verifiable social data outsourcing* whereby a data consumer can verify the trustworthiness of the social data returned by the SDP before making any critical business or personal decision. The data are trustworthy if and only if two conditions are met. First, the data are *correct* in the sense that they satisfy the query criteria and were indeed generated by the purported OSN users. Second,

the data are *complete* in the sense that all the data satisfying the query criteria have been returned.

As the first work along this line, this paper tackles the following specific problem and leaves the investigations of other variations to future work. We view any specific OSN as an undirected social graph, in which each node corresponds to a unique OSN user and has a set of searchable attributes (e.g., location, age, gender, and education level). An edge exists between two nodes if the two users are mutual friends; e.g., they are following each other on Twitter. We focus on single-attribute queries on any node attribute, each of which can be an equality query (e.g., age="30"), a range query (e.g., age="[20, 30]"), or a subset query (e.g., age="20, 25, 30"). The query result corresponds to a subgraph of the original social graph. It is *complete* if containing all the nodes satisfying the query and the associated edges among them, and it is *correct* if none of the nodes and edges therein do not satisfy the query or were illegally added by the SDP.

We propose three solutions to verifiable data queries over outsourced social data. The basic solution requires each OSN provider to generate some cryptographic auxiliary information for its dataset, based on which the SDP can construct a verification object that can be used by the data consumer to verify the query-result trustworthiness. The enhanced solution significantly reduces the computation, storage, and communication overhead of the basic solution by generating the auxiliary information for grouped nodes with identical attribute values. The advanced solution explores a memory-efficient Bloom filter to further reduce the overhead.

We thoroughly evaluate the three solutions on a real Twitter dataset with 1.5 million nodes and 50 million edges. Our experiments show that our basic, enhanced, and advanced schemes can generate the auxiliary information with the size 38.85%, 3.23%, and 0.14% the original social network data in 2973.96s and 204.33s and 211.7s, respectively. In addition, a data consumer can verify a query result returned by the SDP in 631.8ms and 25.3ms and 29.4ms with our basic, enhanced, and advanced schemes, respectively. Our experiment results confirm the efficacy and efficiency of our schemes.

The rest of this paper is outlined as follows. Section II introduces the problem formulation. Section III details our three schemes. Section IV-B analyzes the performance of the three schemes. Section V evaluates the three schemes on a real Twitter dataset. Section VI outlines additional related work. Section VII concludes this paper.

II. Problem Formulation

The system for social data outsourcing comprises a SDP, many OSN operators, and numerous data consumers. The SDP acquires the complete social data from each OSN operator and profits by answering the data queries from data consumers. For example, DataSift has the data from 20 OSNs such as Facebook, Youtube, Tumblr, and Google+. For convenience only, our subsequent illustrations focus on a single OSN operator, but similar operations should be independently performed on the data of each OSN operator.

We represent the social data of the OSN operator as an undirected graph $\mathcal{G}$ with n nodes denoted by $V=\{1,\ldots,n\}$, where each node corresponds to a unique OSN user. We will use nodes and users interchangeably hereafter. An undirected edge $e_{i,j}$ (or equivalently $e_{j,i}$) exists between nodes i and j if and only if the two nodes (users) are mutual friends. Such mutual friendships are natural in Facebook-like OSNs where a friend request needs to be approved. In microblogging systems such as Twitter, anyone can follow arbitrary users (e.g., a pop star), and two users are mutual friends if they follow each other. For the later case, we do not consider unidirectional followings which are too arbitrary/random and much less meaningful than mutual followings. How to extend our work to directed social graphs can be further explored.

Each node has a profile corresponding to w universal attributes specific to each OSN operator, such as location, age, education level, and hobbies. The profile of node i is denoted by $(b_{i,1}, b_{i,2}, \ldots, b_{i,w})$, where $b_{i,j}$ denote the j-th attribute value for any $j \in [1, w]$. An attribute value can be specified by the user him/herself and can also be inferred by the OSN operator based on the user's posts and online interactions [6], [7]. Some attributes such as age have a numeric value by nature, while others such as location and hobbies may have non-numeric values. For the latter case, we assume that the OSN operator has the rules for converting non-numeric attribute values into numeric ones. For example, a location can be converted into a unique sequence of digits like a telephone number, and those in the same metropolitan area can have the same prefix. Such conversions are done by the OSN operator in the background and totally unaware to the user. For simplicity, we assume that each attribute value $b_{i,j}$ in the social dataset represents a unique numeric value (possibly after conversion) in the range of the particular attribute. Each node is also affiliated with the data content s/he has ever generated (e.g., original posts, replies, and comments).

We consider single-attribute queries on any node attribute, each of which can be an equality query (e.g., age="30"), a range query (e.g., age="[20, 30]"), or a subset query (e.g., age="20, 25, 30"). The extension of our work to other query types such as multi-attribute queries is left as future work. If the queried attribute (e.g., location) has a non-numeric value, the same rules used by the OSN operator are applied to generate the corresponding numeric value. The data consumer submits a query to the SDP, which specifies the query condition and the interested OSN as well. An appropriate payment may also need to be submitted simultaneously or later.

The SDP processes the query on the specified OSN dataset. The query result includes a subgraph $\mathcal{G}'$ of $\mathcal{G}$, consisting of all the nodes whose attribute values satisfy the query and also the edges among the nodes in $\mathcal{G}'$. If needed, the posts for each node in $\mathcal{G}'$ need to be returned as well.

We adopt the following **trust and adversary** models. The OSN operator is fully trusted and outsources the authentic dataset to the SDP. In contrast, the SDP is untrusted and may modify the query result by adding, deleting, or modifying data for various malicious motives, e.g., in favor of the

businesses who are willing to pay. We also assume that the communications between the OSN operator and SDP and those between the SDP and the data consumer are secured using traditional mechanisms such as TLS (Transport Layer Security). Therefore, a wrong query result can only be attributed to the misbehavior of the SDP. Also note that the OSN operator is highly motivated to help identify malicious SDPs, as data consumers who make critical decisions based on manipulated query results may eventually blame the OSN operator.

Under the above trust and adversary models, a query result is said to be trustworthy if the following requirements are met.

- *Social-graph correctness*: All the nodes in $\mathcal{G}'$ are indeed in $\mathcal{G}$ and satisfy the query. In addition, all the edges in $\mathcal{G}'$ belong to $\mathcal{G}$. Finally, all the attributes values of each node in $\mathcal{G}'$ are intact in contrast to $\mathcal{G}$.
- *Social-graph completeness*: $\mathcal{G}'$ contains all the nodes in $\mathcal{G}$ that satisfy the query and also all the edges in $\mathcal{G}$ that connect any two nodes in $\mathcal{G}'$.
- *Content authenticity*: The data content returned for each node in $\mathcal{G}'$ is the same as that in $\mathcal{G}$.

Content authenticity can be easily satisfied by letting the query result include the OSN operator's digital signature for each node's data content, which can then be verified by the data consumer. So we subsequently focus on achieving social-graph correctness and completeness.

III. Verifiable Social Data Outsourcing

In this section, we illustrate how to achieve verifiable social data outsourcing by enabling social-graph correctness and completeness verifications. A naïve solution is for the OSN operator to digitally sign the entire dataset, in which case the SDP has to return the entire dataset and the OSN operator's digital signature for each data consumer to verify. This is impractical because most data consumers are only interested in a tiny fraction of the huge dataset. We propose three schemes to enable efficient query processing and also the verifiability of the query result. The three schemes all require the OSN provider to generate some cryptographic auxiliary information, based on which the SDP can construct a verification object for the data consumer to verify the query-result trustworthiness. They differ in how the auxiliary information is generated and how the verification object is constructed and verified.

A. The Basic Scheme

Recall that each node in $\mathcal{G}$ has a profile of w attributes, each having a numeric value after possible conversion. The basic scheme uses cryptographic methods to chain all the nodes in $\mathcal{G}$ and also tie each node with its neighbors (friends) in an ascending order of the attribute values. As long as cryptographic primitives are non-breakable, any manipulated query result can definitely be detected by the data consumer.

1) Generating auxiliary information: Before outsourcing the dataset to the SDP, the OSN operator generates some auxiliary information for each attribute and also for each node to enable the trustworthiness verification of query results.

For each attribute $k \in [1, w]$, the OSN operator creates an array $\Psi_k = \{\psi_{\min}, \psi_1, \ldots, \psi_{\max}\}$, in which each element consists of a prefix (denoted by .pre) and a suffix (denoted by .suf). Each prefix corresponds to a unique attribute value, while each suffix equals the XORs of hashed node IDs with the corresponding attribute value. Specifically, let ID_i ($\forall i \in [1, n]$) denote the ID of node i, which can be either a real identifier or a pseudonym assigned by the OSN operator for privacy concerns. Ψ_k initially contains two elements $\psi_{\min}$ and $\psi_{\max}$, where $\psi_{\min}$.pre and $\psi_{\max}$.pre equal the minimum possible attribute value minus one and the maximum possible attribute value plus one, respectively. In contrast, $\psi_{\min}$.suf and $\psi_{\max}$.suf are both set to zero. Then the OSN operator checks Ψ_k for the attribute value $b_{i,k}$ of each node $i \in [1, n]$. If there is an element, say ψ_x with ψ_x.pre $= b_{i,k}$, the OSN operator updates ψ_x.suf $:= \psi_x$.suf $\bigoplus h(ID_i)$, where $h(\cdot)$ denotes a cryptographic one-way hash function such as SHA. If $b_{i,k}$ does not exist in Ψ_k, a new element, say ψ_x with ψ_x.pre $= b_{i,k}$ and ψ_x.suf $= h(ID_i)$, is inserted into Ψ_k, of which the preceding and subsequent elements have the prefix value smaller and larger than $b_{i,k}$, respectively.

The next step is to build a binary Merkle Hash Tree (MHT) [8] over the elements in Ψ_k. The MHT is a commonly used cryptographic data structure that supports very efficient and secure verification of large-scale datasets. The OSN operator first computes each leaf node as the hash of the concatenation of the prefix and the suffix of a unique element in Ψ_k, and the order of the leaf nodes corresponds to the element order in Ψ_k. Each internal node of the MHT is then derived as the hash of the concatenation of its two children nodes.[1] Finally, the OSN operator uses its private key to digitally sign the root of the MHT. The auxiliary information for attribute $k \in [1, w]$, denoted by $\mathcal{AU}_{\Psi_k}$, consists of all the internal nodes of the MHT and the signature as well. Later we will see that $\mathcal{AU}_{\Psi_k}$ enables verifiable social-graph correctness and completeness pertaining to each individual node.

The OSN operator also generates the auxiliary information, denoted by $\mathcal{AU}_i$, for each node $i \in [1, n]$ to ensure the correctness and completeness verifications of the edges (i.e., social links) in the query result. $\mathcal{AU}_i$ contains $w + 1$ elements, denoted by $\mathcal{AU}_{i,k}$ for $k \in [0, w]$. The first element $\mathcal{AU}_{i,0}$ is simply the OSN operator's digital signature over $h(b_{i,1} \parallel b_{i,2} \parallel \cdots \parallel b_{i,w} \parallel ID_i)$, i.e., the hash of the concatenation of node i's w attribute values and ID. Each other element $\mathcal{AU}_{i,k}$ ($\forall k \in [1, w]$) corresponds to the k-th attribute. The OSN operator derives $\mathcal{AU}_{i,k}$ in the same way as computing $\mathcal{AU}_{\Psi_k}$ with the exception that the operations involve only node i and all its neighbors (friends) in $\mathcal{G}$.

Finally, the OSN operator sends to the SDP its entire social dataset and all the auxiliary information, i.e., $\{\mathcal{AU}_{\Psi_k} | k \in [1, w]\}$ and $\{\mathcal{AU}_{i,k} | i \in [1, n], k \in [0, w]\}$.

2) Query Processing: We now illustrate how the SDP processes a query. A subset query can be decomposed into multiple equality queries, each of which can be considered a special range query with just one queried attribute value.

[1]Note that if the number of leaf nodes is not power of two, some dummy leaf nodes (e.g., $\psi_{\max}$) need be introduced for constructing the MHT.

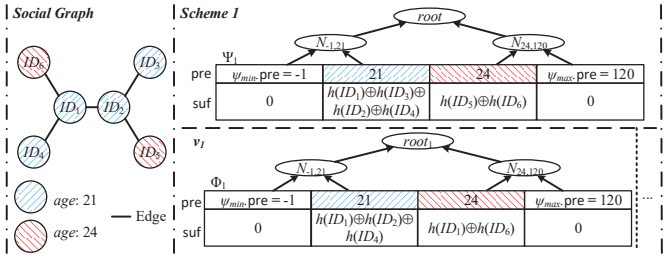


Fig. 1: Illustration of the basic scheme.

Our subsequent discussions thus focus on a range query $[\delta_{\min}, \delta_{\max}]$ over an arbitrary attribute $k \in [1, w]$, where $\delta_{\min}$ and $\delta_{\max}$ denote the minimum and maximum attribute values of interest, respectively.

After receiving a data request $\langle k, \delta_{\min}, \delta_{\max} \rangle$ for a specific OSN dataset, the SDP searches the corresponding social graph $\mathcal{G}$ to locate all the nodes whose k-th attribute value belongs to $[\delta_{\min}, \delta_{\max}]$ and then constructs a subgraph $\mathcal{G}'$ which consists of all these qualified nodes and the edges among them in $\mathcal{G}$. The query result includes $\mathcal{G}'$ as well as a verification object the SDP constructs according to $\{\mathcal{AU}_{\Psi_k} | k \in [1, w]\}$ and $\{\mathcal{AU}_{i,k} | i \in [1, n], k \in [0, w]\}$ as follows.

First, the SDP locates the maximum k-th attribute value just below $\delta_{\min}$ (denoted by $\delta^-_{\min}$) and also the minimum k-th attribute value just above $\delta_{\max}$ (denoted by $\delta^+_{\max}$). We refer to $\delta^-_{\min}$ and $\delta^+_{\max}$ as boundary values. The SDP then determines the unique attribute values falling into $[\delta_{\min}, \delta_{\max}]$, as multiple qualified nodes may have the same attribute value.

Second, the SDP checks $\mathcal{AU}_{\Psi_k}$ to obtain the auxiliary authentication information needed to reconstruct the MHT root for the aforementioned array $\Psi_k = \{\psi_{\min}, \psi_1, \ldots, \psi_{\max}\}$ (see Section III-A1). Specifically, the qualified attribute values and two boundary values each correspond to the prefix of a unique element in Ψ_k, so each hash value of the concatenation of the prefix and its corresponding suffix is a leaf node of the MHT. The auxiliary authentication information for each such leaf node includes the siblings of itself, its parent, its parent's parent, etc. Since these leaf nodes are adjacent in the MHT, their auxiliary authentication information should be combined to reduce the likely redundancy. The verification object includes all the auxiliary authentication information and related root signature.

Third, the SDP does the similar operations as above for each qualified node i based on $\mathcal{AU}_{i,k}$. These operations involve node i and its neighbors only (see Section III-A1). The verification object additionally contains all the related auxiliary authentication information and signatures as well, which includes $\mathcal{AU}_{i,0}$, i.e., the OSN operator's digital signature over $h(b_{i,1} \parallel b_{i,2} \parallel \cdots \parallel b_{i,w} \parallel ID_i)$ for each qualified node i.

3) Correctness and completeness verification: The data consumer verifies the correctness and completeness of the query result through the following operations in sequel.

First, the data consumer verifies that each node i in $\mathcal{G}'$ has the k-th attribute value in $[\delta_{\min}, \delta_{\max}]$, and the signature $\mathcal{AU}_{i,0}$ is correct. This step ensures that $\mathcal{G}'$ only contains qualified nodes whose attribute values are authentic.

Second, the data consumer reconstructs the MHT root of Ψ_k based on $\mathcal{G}'$ and the auxiliary authentication information in the verification object. The reconstructed MHT root should match the signed one in the verification object. This step ensures that $\mathcal{G}'$ includes all the qualified nodes.

Finally, the data consumer uses the auxiliary authentication information in the verification object to reconstruct the root of the MHT for each node i and all its neighbors in $\mathcal{G}'$. The reconstructed MHT root should match the signed one in the verification object. This step ensures that $\mathcal{G}'$ contains all the edges between each qualified node i and its qualified neighbors in the original graph $\mathcal{G}$ but not any fake edge.

The query result is considered complete and correct if all the verifications above succeed. The security of this basic scheme relies on the unanimously assumed security of the cryptographic hash function $h(\cdot)$ and the digital signature scheme. In particular, the SDP cannot fabricate a query result that can lead to valid MHT roots with correct signatures.

4) A working example: To better illustrate the basic scheme, we show an example in Fig. 1 with $n = 6$ nodes for a single attribute *age*, where the top-right and top-bottom subfigures correspond to the MHTs for the entire graph and for node 1 and its neighbors, respectively. Suppose that the data consumer queries the nodes age 21 and the edges among these nodes. The qualified nodes are {1, 2, 3, 4}, and the edges are $\{e_{1,2}, e_{1,4}, e_{2,3}\}$, respectively. Node 1 has two neighbors (2 and 4) with age 21, and the boundary values $\delta^-_{\min}$ and $\delta^+_{\max}$ are -1 and 24, respectively. The verification object thus contains the following information for node 1, $\{-1||0,\ 24||h(ID_1) \bigoplus h(ID_6),\ 120||0,\ \mathcal{S}(root_1)\}$, where $\mathcal{S}(\cdot)$ denotes the OSN operator's digital signature operation. Similarly, we have $\{-1||0,\ 24||h(ID_2) \bigoplus h(ID_5),\ 120||0,\ \mathcal{S}(root_2)\}$, $\{-1||0,\ 120||0,\ \mathcal{S}(root_3)\}$ and $\{-1||0,\ 120||0,\ \mathcal{S}(root_4)\}$ in the verification object for nodes 2, 3 and 4, respectively. Besides, the verification object contains $\{-1||0,\ 24||h(ID_5) \bigoplus h(ID_6),\ 120||0,\ \mathcal{S}(root)\}$ for the array Ψ_1 covering all the nodes.

After receiving the query result, the data consumer derives $N_{-1,21}$ and $N_{24,120}$ whereby to compute the MHT root. If the root hash equals the signed one in the verification object, the query result is determined to contain all the qualified nodes. Similarly, the data consumer derives candidate $root_1$, $root_2$, $root_3$, and $root_4$. If they all match the signed counterpart in the verification object, the query result contains all the qualified edges. If the query result passes all the verifications, the data consumer considers it correct and also complete.

B. The Enhanced Scheme

The basic scheme achieves deterministic detection of fake query results at the cost of huge overhead. Consider the generation of the auxiliary information $\{\mathcal{AU}_{i,k} | i \in [1, n], k \in [1, w]\}$ as an example. The basic scheme requires the OSN operator to construct a distinct MHT per attribute per node, leading to nw MHTs in total. Each MHT involves the OSN operator signing the root. Since n is extremely large in practice, the computational overhead for signature generation can be daunting especially given that the dataset needs to be updated

from time to time. Motivated by the observation that many nodes have the same attribute value, we propose the enhanced scheme to build an MHT for each unique attribute value instead of each unique node. The OSN operator performs the following steps in sequel to generate the auxiliary information for its dataset.

First, the OSN operator computes $\mathcal{AU}_{i,0}$ as $h(b_{i,1} \parallel b_{i,2} \parallel \cdots \parallel b_{i,w} \parallel ID_i)$ for each node $i \in [1,n]$, i.e., the hash of the concatenation of node i's w attribute values and ID. This is the same as in the basic scheme except without signing.

Second, for each attribute $k \in [1,w]$, the OSN operator creates an array $\Psi'_k = \{\psi'_{\min}, \psi'_1, \ldots, \psi'_{\max}\}$, in which each element consists of a prefix (denoted by .pre), an infix (denoted by .inf), and a suffix (denoted by .suf). As in the basic scheme, Ψ'_k initially contains two elements $\psi'_{\min}$ and $\psi'_{\max}$, where $\psi'_{\min}$.pre and $\psi'_{\max}$.pre equal the minimum possible attribute value minus one and the maximum possible attribute value plus one, respectively. In contrast, $\psi'_{\min}$.inf, $\psi'_{\min}$.suf, $\psi'_{\max}$.inf, and $\psi'_{\max}$.suf are all set to zero. Then the OSN checks Ψ'_k for the attribute value $b_{i,k}$ of each node $i \in [1,n]$. If there is an element, say ψ'_x with $\psi'_x.\mathsf{pre} = b_{i,k}$, the OSN operator updates $\psi'_x.\mathsf{inf} := \psi'_x.\mathsf{inf} \bigoplus \mathcal{AU}_{i,0}$ and $\psi'_x.\mathsf{suf} := \psi'_x.\mathsf{suf} \bigoplus h(ID_i)$. If $b_{i,k}$ does not exist in Ψ'_k, a new element, say ψ'_x with $\psi'_x.\mathsf{pre} = b_{i,k}$, $\psi'_x.\mathsf{inf} = \mathcal{AU}_{i,0}$, and $\psi'_x.\mathsf{suf} = h(ID_i)$, is inserted into Ψ'_k, of which the preceding and subsequent elements have the prefix value smaller and larger than $b_{i,k}$, respectively. Subsequently, the OSN operator derives an MHT based on the concatenation of the prefix, the infix and the suffix of each element in Ψ'_k, and it also signs the MHT root.

Third, the OSN operator creates an array $\Psi_{i,k}$ ($\forall k \in [1,w]$) for each node $i \in [1,n]$ and its neighbors in the graph $\mathcal{G}$. Each element in $\Psi_{i,k}$ consists of a prefix equal to a unique value in attribute k and also a suffix equal to the concatenation of the ID hashes of the nodes with the corresponding attribute value. Each $\Psi_{i,k}$ is the same as that in the basic scheme, where the prefix values are ranked in the ascending order.

Fourth, the OSN operator constructs another array for each unique value in attribute k ($\forall k \in [1,w]$), in which each element is composed of a prefix (denoted by .pre) and a suffix (denoted by .suf). Let m_k denote the number of unique values in attribute k and $\Phi_{j,k}$ denote the j-th array ($\forall j \in [1,m_k]$). $\Phi_{j,k}$ is also initialized with two dummy elements corresponding to the minimum attribute value minus one and the maximum attribute value plus one, respectively.

Fifth, for each node $i \in [1,n]$, the OSN operator finds the corresponding array corresponding to $b_{i,k}$, say $\Phi_{x,k}$. Each element in the array $\Psi_{i,k}$ is then traversed. If its prefix value can be found in $\Phi_{x,k}$, its suffix is appended to that of the corresponding element in $\Phi_{x,k}$; otherwise, it is inserted into $\Phi_{x,k}$ in the ascending order of the attribute value.

Finally, the OSN operator generates an MHT for each $\Phi_{j,k}$ ($\forall j \in [1,m_k], k \in [1,w]$) based on its concatenation values in the similar way as before, and also signs the root of each MHT tree.

As in the basic scheme, the auxiliary information includes the internal nodes of all the MHT trees and also all the signatures. It is sent along with the dataset to the SDP. In addition, the enhanced and basic schemes use almost the same processes to process a query and verify a query result. The only exception is that the verification object should contain the auxiliary authentication information necessary to reconstruct and verify the roots of the MHTs whose corresponding attribute values satisfy the query. We do not repeat the redundant operations due to space constraints.

We continue with the example in Fig. 1. The data consumer also queries the nodes age 21 and their edges. The verification information for the attribute value 21 includes $\{-1||0, 24||h(ID_2)\bigoplus h(ID_5)||h(ID_1)\bigoplus h(ID_6), 120||0, \mathcal{S}(root_{21})\}$. Based on this verification information, the data consumer can ensure the completeness of edges among qualified nodes in $\mathcal{G}'$. Moreover, the verification information from the array Ψ'_k is $\{-1||0||0, 24||h(24||ID_5)\bigoplus h(24||ID_6)||h(ID_5)\bigoplus h(ID_6), 120||0||0, \mathcal{S}(root)\}$. Based on this verification information, the data consumer can guarantee the completeness of attribute values and nodes.

C. The Advanced Scheme

The advanced scheme further reduces the signature operations by constructing a Bloom filter instead of an MHT for each $\Phi_{j,k}$ ($\forall j \in [1,m_k], k \in [1,w]$), where m_k again denotes the number of unique values in attribute k.

A Bloom Filter [9] is a space-efficient probabilistic data structure for set-membership testing and many other applications [10], [11]. Assume that we want to use an α-bit Bloom filter for a data set $\{d_i\}_{i=1}^{\beta}$, which has every bit initialized to bit-0. Let $\{h_j(\cdot)\}_{j=1}^{\tau}$ denote τ different hash functions, each with output in $[1,\alpha]$. Every element d_i is added into the Bloom filter by setting all bits at positions $\{h_j(d_i)\}_{j=1}^{\tau}$ to bit-1. To check the membership of an arbitrary element e in the given data set, we can simply verify whether all the bits at positions $\{h_j(e)\}_{j=1}^{\tau}$ have been set. If not, e is certainly not in the data set; otherwise, it is in the data set with some probability jointly determined by α, β, and τ.

In the advanced scheme, the OSN operator generates $\Phi_{j,k}$ ($\forall j \in [1,m_k], k \in [1,w]$) as in the enhanced scheme and then initializes the corresponding Bloom filter $\mathcal{BF}_{j,k}$. The prefix and suffix of each element in $\Phi_{j,k}$ (except the dummy elements) are both inserted into $\mathcal{BF}_{j,k}$. Finally, the OSN operator signs $\mathcal{BF}_{j,k}$.

The auxiliary information is now composed of all the internal nodes of the MHT for Ψ'_k, the Bloom filters $\{\mathcal{BF}_{j,k} | j \in [1,m_k], k \in [1,w]\}$, and all the related signatures. In addition, the advanced scheme uses the almost identical processes for query processing and query-result verification to those in the basic and enhanced schemes. The only exception is that the verification object should contain the signed Bloom filters whose corresponding attribute values satisfy the query. Therefore, the data consumer can construct an array like $\Phi_{j,k}$ for each unique attribute value in the query result and check whether all the array elements are in the corresponding signed Bloom filter. If not, the query result is untrustworthy. Other operations are omitted here for lack of space.

Let us continue the example in Fig. 1. Given the same query for age 21, the verification information from the array Ψ'_k is identical with that in the enhanced scheme. However, the verification information for the attribute value 21 changes to $\{\mathcal{BF}_{21,1},\ \mathcal{S}(\mathcal{BF}_{21,1})\}$. The data consumer first matches the Bloom filter $\mathcal{BF}_{21,1}$ with the OSN operator's signature $\mathcal{S}(\mathcal{BF}_{21,1})$. Subsequently, it verifies whether the two values, $h(21)$ and $h(h(ID_1)\bigoplus h(ID_2)\bigoplus h(ID_4)||h(ID_2)\bigoplus h(ID_1)\bigoplus h(ID_3)||h(ID_3)\bigoplus h(ID_2)||h(ID_4)\bigoplus h(ID_1))$, are both in the Bloom filter $\mathcal{BF}_{21,1}$. If the query result passes these verification, the data consumer considers it correct and complete.

IV. Security and Overhead Analysis

In this section, we briefly analyze the security and overhead of the proposed schemes.

A. Security Analysis

The basic and enhanced schemes both enable a data consumer to detect an incorrect and/or incomplete query result in a deterministic fashion. The reason is that the auxiliary information amounts to chaining the authentic nodes, attribute values, and edges with cryptographic methods. As long as the hash function and digital signature scheme used for constructing the MHTs are secure, the SDP cannot modify the authentic query result without failing the signature verification.

In contrast, the advanced scheme detects an incorrect and/or incomplete query result with overwhelming probability. On the one hand, the signed MHT can effectively prevent the SDP from inserting/delecting nodes or modifying any attribute value in the query result while escaping the detection. On the other hand, the signed Bloom filters may not reveal all the illegitimate edge insertions or deletions by the SDP. In particular, assume that the SDP deletes (or adds) one edge between nodes i and l from the query result for attribute k. In the query-result verification phase, the data consumer derives many array elements and then check whether each of them is indeed in the corresponding Bloom filter. An edge deletion (or addition) results in two new array elements related to nodes i and l, respectively. If both of them are found in the corresponding Bloom filter (i.e., two false positives), the SDP escapes the detection. The false-positive probability of a Bloom filter with parameters $\langle \alpha, \beta, \tau \rangle$ can be easily estimated as $0.6185^{\alpha/\beta}$, for which τ is set as $\frac{\alpha}{\beta} \times \ln 2$ to minimize the false-positive rate [9]. So the SDP can escape the detection with probability $(0.6185^{\alpha/\beta})^{2\chi}$ for adding (or deleting) $\chi \geq 1$ edges from the query result.

B. Overhead Analysis

Now we analyze the computation, communication, and storage overhead our schemes incur to enable the correctness and completeness verifications of the query results. The overhead that exists with or without our schemes is ignored here (e.g., the time to search for qualified nodes).

1) Computation overhead: All our schemes involve digital signature generations, verifications, and hash operations. The computation overhead is dominated by signature generations and verifications (especially, the former), so we can safely ignore the hash operations for simplicity.

First, we estimate the computation overhead at the OSN operator for generating the auxiliary information. The OSN operator needs $2{\cdot}n{+}1$ signature operations for attribute k in the basic scheme, thus the complexity of signature operations is $\mathcal{O}(n)$. The OSN operator takes $m_k{+}1$ signature operations for attribute k in the enhanced scheme. Hence, the complexity of signature operations in the enhanced scheme is $\mathcal{O}(m_k)$. In the advanced scheme, the OSN operator also takes $m_k{+}1$ signature operations, so the complexity is $\mathcal{O}(m_k)$.

Next, we discuss the computation overhead at the data consumer for verifying a query result. Suppose that the number of nodes in the query result is z. In the basic scheme, the data consumer performs up to $2{\cdot}z{+}1$ signature verifications, leading to the complexity of $\mathcal{O}(z)$. Assume that the number of unique attribute values in the query result is m'_k. In the enhanced scheme, the data consumer performs up to $m'_k{+}1$ signature verifications, resulting in the complexity of $\mathcal{O}(m'_k)$. As well, the advanced scheme requires the data consumer to verify no more than $m'_k{+}1$ signatures, meaning the complexity of $\mathcal{O}(m'_k)$.

2) Communication and storage Overhead: We first discuss the communication overhead for transmitting the auxiliary information from the OSN operator to the SDP, which is the same as the storage overhead at the SDP for storing the auxiliary information. For convenience, only a single attribute k is considered, and the overall overhead can simply be scaled by a factor of w for all w attributes. Let l_{hash} and l_{sig} denote the lengths of a hash value and a digital signature, respectively. Assume that node i and its neighbors have $\theta_{i,k}$ unique values for attribute k. In the basic scheme, the communication overhead includes $(2{\cdot}n{+}1){\cdot}l_{\text{sig}}$ and $(\sum_{i=1}^{n}(2^{\log_2\lceil\theta_{i,k}\rceil}{-}2){+}2^{\log_2\lceil m_k\rceil}{-}2){\cdot}l_{\text{hash}}$. Since $\theta_{i,k}$ is no more than m_k, the communication complexity of the basic scheme is $\mathcal{O}(n\cdot 2^{\log_2\lceil m_k\rceil})$ hash values and $\mathcal{O}(n)$ signatures. Let $\rho_{j,k}$ denote the number of unique attribute values of neighbors for nodes with each unique value in attribute k. In the enhanced scheme, the communication overhead consists of $(m_k{+}1){\cdot}l_{\text{sig}}$ and $(\sum_{i=1}^{m_k}(2^{\log_2\lceil\rho_{j,k}\rceil}{-}2){+}2^{\log_2\lceil m_k\rceil}{-}2){\cdot}l_{\text{hash}}$. Likewise, the worst case for $\rho_{j,k}$ is also m_k, the enhanced scheme incurs the communication complexity of $\mathcal{O}(m_k\cdot 2^{\log_2\lceil m_k\rceil})$ hash values and $\mathcal{O}(m_k)$ signatures. Finally, the communication overhead of the advanced scheme is composed of $(m_k{+}1){\cdot}l_{\text{sig}}$, $(m_k{-}2){\cdot}l_{\text{hash}}$, and $\sum_{j=1}^{m_k} l_{\mathcal{BF}_{j,k}}$, where $l_{\mathcal{BF}_{j,k}}$ denotes the length of the Bloom filter $\mathcal{BF}_{j,k}$. So the advanced scheme incurs the communication complexity of $\mathcal{O}(m_k)$ signatures, $\mathcal{O}(m_k)$ hash values, and $\mathcal{O}(m_k)$ Bloom filters.

The three schemes also incur the communication overhead for transmitting the verification object from the SDP to the data consumer, which depends on the particular query that determines how many nodes and edges are in the query result. Since it is difficult to give a meaningful, generic

TABLE I: Computation overhead

	Basic			Enhanced			Advanced		
	hash	*XOR*	*sign*	*hash*	*XOR*	*sign*	*hash*	*XOR*	*sign*
S-100K	4.92M	3.47M	192.28K	3.71M	3.13M	1.2K	3.38M	3.13M	1.2K
S-1M	50.5M	35.5M	1.98M	35.9M	34.2M	3.49K	34.5M	34.2M	3.49K
S-1.5M	76.0M	53.5M	2.98M	53.74M	51.87M	4.07K	52.06M	51.87M	4.07K

estimation, we omit it here which is considerably smaller than the communication overhead for transmitting the auxiliary information from the OSN operator to the SDP.

V. Experimental Results

In this section, we experimentally evaluate the three schemes using experiments with real datasets. We implement all three schemes using Python 2.7 with totally 1800+ lines of codes. All the experiments are carried out on a commodity PC, with 3.4 GHz Intel-i7 3770 CPU, 16 GB memory, a 7200 RPM hard disk, and Windows 10 OS. The false positive probability of the Bloom filter in the advanced scheme is set to 0.001 unless mentioned otherwise.

A. Datasets

We use a real-world Twitter dataset that we collected in March 2016. Specifically, we randomly selected 100,000 Twitter users as seeds. For each seed user, we crawled his attribute, followers and friends, and obtained nearly 4M users' attributes and the corresponding social graph. Our subsequent experiments focus on location attribute. For our purpose, we extracted location with city-level labels in the form of "cityName,stateName" or "cityName,stateAbbreviation", where we considered all cities in the "List of Valid U.S. cities" [2]. Among all the crawled users, we found 1.6M users with valid location attribute and more than 50M edges among them.

To evaluate the performance of our schemes for datasets with different sizes, we randomly sampled three groups of users (*S*-100K, *S*-1M, *S*-1.5M) from the above dataset, i.e., 100K, 1M, 1.5M users and their corresponding social graphs. The sizes of three sampled datasets *S*-100K, *S*-1M and *S*-1.5M are 315.58MB (10.95MB users' attribute, 304.63MB social network), 3.13GB (109.57MB users' attribute, 3.02GB social network), 4.71GB (164.36MB users' attribute, 4.54GB social network), respectively.

B. Generating Auxiliary Information

We now evaluate the computation and storage overhead incurred by generating auxiliary information in the basic, enhanced, and advanced schemes.

Computation overhead. We compare the three schemes in terms of the number of *hash*, *XOR*, and *signature* operations, and computation time. Table I lists the numbers of *hash*, *XOR*, and *sign* operations for all three schemes and all three datasets. We can see from Table I that (1) The basic scheme requires the highest number of *hash* operations, followed by the enhanced scheme and the advanced scheme; (2) all three schemes require similar numbers of *XOR* operations; and (3) the number of signature operations in the advanced scheme and enhanced schemes are significantly lower than that of the basic scheme. For example, and the number of signature operations in the advanced and enhanced schemes are only 0.13% of that in the basic scheme for *S*-1.5M, respectively. This is expected, as the complexity of signature operations in our three schemes are $\mathcal{O}(n)$, $\mathcal{O}(m_k)$ and $\mathcal{O}(m_k)$, respectively. Finally, Table II shows the total computation time of generating auxiliary information for three datasets. We can see the advanced scheme and the enhanced scheme almost take the same time for generating auxiliary information, and shorter time than the basic scheme. For example, the computation time in the basic scheme is 14.04× of that in the advanced scheme for *S*-1.5M.

TABLE II: Computation time.

	Basic	Enhanced	Advanced
S-100K	192.69s	16.33s	16.88s
S-1M	2111.46s	139.91s	143.23s
S-1.5M	2973.96s	204.33s	211.7s

TABLE III: Storage overhead.

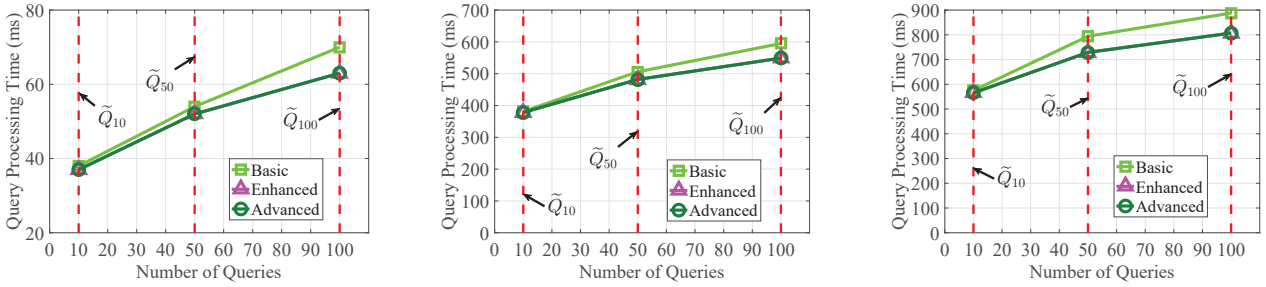
	Basic	Enhanced	Advanced
S-100K	121.22MB(38.41%)	31.16MB(9.87%)	1.44MB(0.45%)
S-1M	1.22GB(38.98%)	125.42MB(3.91%)	5.49MB(0.17%)
S-1.5M	1.83GB(38.85%)	155.8MB(3.23%)	6.76MB(0.14%)

Storage overhead. Table III shows the storage overhead of auxiliary information, i.e., the total size of signatures, internal nodes of MHTs, and Bloom filters, in the basic, enhanced, and advanced schemes in bits. We can see from Table III that the basic scheme incurs the highest storage overhead, followed by the enhanced scheme and the advanced scheme. Consider *S*-1.5M as an example, the storage overhead of the basic, enhanced or advanced scheme are 1.83GB, 155.8MB or 6.76MB, respectively. This is of no surprise, as the complexities of storage overhead for signatures and hash values in the basic and enhanced schemes are $\langle \mathcal{O}(n), \mathcal{O}(n \cdot 2^{\log_2 \lceil m_k \rceil}) \rangle$ and $\langle \mathcal{O}(m_k), \mathcal{O}(m_k \cdot 2^{\log_2 \lceil m_k \rceil}) \rangle$, and the advanced scheme utilizes space-efficient data structure (Bloom filter) to further reduce the storage overhead. Table III also shows the ratio between the the size of the auxiliary information and that of original user data. It is clear that the advanced scheme incurs very small additional storage overhead for auxiliary information.

C. Query Processing

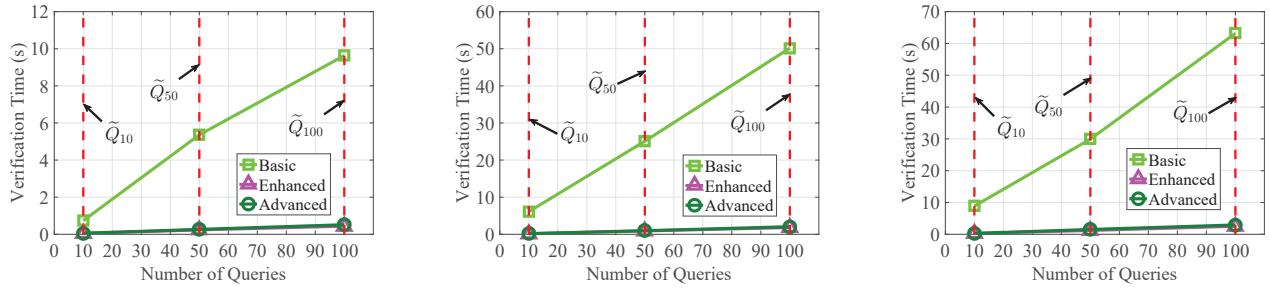
To evaluate the computation overhead incurred by query processing, we generate three types of queries: $\widetilde{Q}_{10}$, $\widetilde{Q}_{50}$, and $\widetilde{Q}_{100}$, where the query $\widetilde{Q}_x$ means randomly choosing x cities as the query condition for $x = 10, 50$ and 100. Fig. 3a, 3b and 3c show the query processing times of three schemes for the three types of queries for all three datasets, where each point represents the average of 100 runs, each with a random seed. We can see that the query processing time of the basic

[2] https://www.whitehouse.gov/sites/default/files/omb/assets/procurement_fair/usps_city_state_list.xls



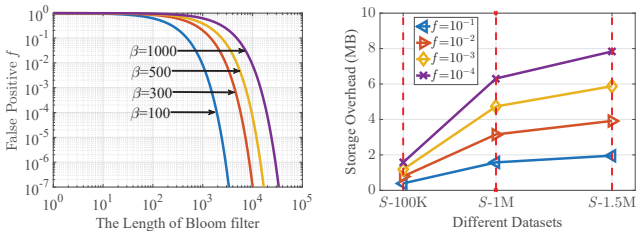
(a) *S*-100K (b) *S*-1M (c) *S*-1.5M

Fig. 2: Comparison of query processing time.



(a) *S*-100K (b) *S*-1M (c) *S*-1.5M

Fig. 3: Comparison of query-result verification time.



(a) Bloom filter size (b) Different f

Fig. 4: Impact of Bloom filter size on false positive probability and storage overhead.

scheme is the longest and that of the enhanced and advanced schemes are similar. For *S*-1.5M, it takes the SDP only 807ms to process 100 queries under the advanced scheme.

D. Query-Result Verification

Figs. 3a to 3c compare the time for query-result verification under three proposed schemes for three datasets. We can see that the verification time in the advanced and enhanced schemes are almost the same, followed by the basic schemes. This is expected, as the complexities of signature verification in three schemes are $\mathcal{O}(z \cdot m'_k)$, $\mathcal{O}(m'_k)$ and $\mathcal{O}(m'_k)$, respectively. Moreover, the average verification time in the advanced scheme is 29.4ms for the dataset *S*-1.5M, which clearly shows its high efficiency for real application.

E. Impact of Bloom Filter on the Advanced Scheme

Fig. 4a shows the false-positive probability of a Bloom filter varying with its length, where the number of elements β=100, 300, 500, and 1,000. We can see that the lower the false positive probability we desire, the larger the Bloom filter needs to be. In addition, the more elements inserted into the Bloom filter, the higher the false positive probability, and vice versa. These results coincide with the property of Bloom filter. Fig. 4b shows the storage overhead incurred by all the Bloom filters for *S*-100K, *S*-1M and *S*-1.5M with false positive probability varying from 10^{-1} to 10^{-4}. We can see that if we reduce false positive probability from 10^{-3} to 10^{-4}, the storage overhead incurred by all the Bloom filters for *S*-1.5M increases by only 2MB. Recall that Table III that the total storage overhead of the advanced scheme is 6.76MB for *S*-1.5M when f=10^{-3}, so the total storage overhead is 8.76MB for *S*-1.5M when f=10^{-4}, which is still significant lower than that of the basic and the enhanced schemes.

VI. Related Work

Our work is mostly close to data outsourcing [12] paradigm, in which the data owner outsources its dataset to a third party service provider, which in turn answers data queries from either the data owner or other users.

A major security challenge in data outsourcing is to ensure the integrity of the query results [13]. A common solution is to let the data owner outsource both its dataset and also some auxiliary information over the data to the service provider which returns both the query result and a verification object computed from the auxiliary information for the querying user

to verify query integrity. Many techniques based on signature chaining were proposed for auxiliary information and $\mathcal{VO}$ generations. Narasimha *et al.* [14] proposed an approach *DSAC* based on signature chain to verify the integrity of dynamic databases. Pang *et al.* [15] proposed a novel signature caching scheme *SigCache* to reduce the overhead of *DSAC*. In [16], the authors proposed efficient authentication schemes for single- and multi-attribute range aggregate queries. There are also some schemes based on Merkle Hash Tree (MHT) [17] or its variants proposed for authenticating aggregation queries [18], kNN queries [19], [20], top-k spatial keyword queries [21], [22], and location-based skyline queries [23]. However, none of these schemes consider query over graph data, so they cannot be applied to our problem.

Another line of research has been devoted to ensure the integrity of outsourced query over graph data. Goodrich *et al.* [24] proposed a scheme to verify whether two nodes are connected in the graph. Yiu *et al.* [25] proposed a landmark-based verification method to verify shortest-path query results. Moreover, Fan *et al.* [26] proposed a technique to authenticate subgraph query. Nevertheless, none of these schemes can be applied to verifiable range queries over social graph.

Authenticating outsourced query processing has also been studied in other contexts. Zhang *et al.* [22] presented several techniques to enable efficient verification of location-based top-k query results returned by untrusted location-based service providers. Besides, the techniques in [27]–[30] allow a sensor network owner to verify range-query results over encrypted data stored inside the network. These schemes target different scenarios and are orthogonal to our work here.

VII. Conclusions

In this paper, we initiated the study of verifiable social data outsourcing to allow a data consumer to verify the trustworthiness of the social data returned by the SDP. Specifically, we have proposed three solutions to allow the data consumer to verify the social-graph correctness, social-graph completeness, and content authenticity of any query result returned by an untrusted SDP. The efficacy and efficiency of our solutions have been confirmed by extensive experiments based on real Twitter dataset.

Acknowledgment

This work was partially supported by US Army Research Office through grant W911NF-15-1-0328, US National Science Foundation through grants CNS-1700032 and CNS-1700039, and National Natural Science Foundation of China through grants 61472125 and 61402161.

References

[1] B. Jason, "The impact of social media marketing trends on digital marketing," Mar. 2014. [Online]. Available: http://www.socialmediatoday.com/content/impact-social-media-marketing-trends-digital-marketing

[2] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose," in *ICWSM'13*, Boston, MA, Jul. 2013.

[3] F. Morstatter, H. Dani, J. Sampson, and H. Liu, "Can One Tamper with the Sample API?: Toward Neutralizing Bias from Spam and Bot Content," in *WWW'16*, Montreal, Canada, Apr. 2016.

[4] C. Patrick, "Yelp's newest weapon against fake reviews: Lawsuits," Jul. 2014. [Online]. Available: http://www.bloomberg.com/news/articles/2013-09-09/yelps-newest-weapon-against-fake-reviews-lawsuits

[5] A. Chang, "Tempers flare at yelp's town hall for small business owners in l.a." Aug. 2013. [Online]. Available: http://articles.latimes.com/2013/aug/21/business/la-fi-tn-yelp-town-hall-reviews-20130820

[6] J. Zhang, J. Sun, R. Zhang, and Y. Zhang, "Your age is no secret: Inferring microbloggers' ages via content and interaction analysis," in *IEEE CNS'15*, Florence, Italy, Sept. 2015.

[7] J. Zhang, X. Hu, Y. Zhang, and H. Liu, "Your age is no secret: Inferring microbloggers' ages via content and interaction analysis," in *ICWSM'16*, Cologne, Germany, May 2016.

[8] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine, "Authentic data publication over the internet," *Journal of Computer Security*, vol. 11, no. 3, pp. 291 – 314, Jan. 2003.

[9] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[10] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," pp. 636–646, 2002.

[11] Y. Zhao and J. Wu, "B-SUB: A practical bloom-filter-based publish-subscribe system for human networks," in *ICDCS'10*, Genoa, Italy, June 2010.

[12] H. Hacigümüs, B. Iyer, and S. Mehrotra, "Providing database as a service," in *ICDE'02*, San Jose, CA, Feb. – Mar. 2002.

[13] S. Ku, L. Hu, C. Shahabi, and H. Wang, "Query integrity assurance of location-based services accessing outsourced spatial databases," in *SSTD'09*, Aalborg, Denmark, Jul. 2009.

[14] M. Narasimha and G. Tsudik, "Authentication of outsourced databases using signature aggregation and chaining," in *DASFAA'06*, Singapore, Apr. 2006.

[15] H. Pang, L. Zhang, and K. Mouratidis, "Scalable verification for outsourced dynamic databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 802 – 813, Aug. 2009.

[16] H. Pang and L. Tan, "Verifying completeness of relational query answers from online servers," *ACM Transactions on Information and System Security*, vol. 11, no. 2, pp. 5:1 – 5:50, May 2008.

[17] C. Merkle, "A certified digital signature," in *CRYPTO'89*, Santa Barbara, CA, Aug. 1989.

[18] Y. Liu, P. Ning, and H. Dai, "Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures," in *S&P'10*, Washington, DC, May 2010.

[19] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with voronoi neighbors," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 863 – 876, Apr. 2013.

[20] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving knn queries," in *ICDE'11*, Hannover, Germany, Apr. 2011.

[21] D. Wu, B. Choi, J. Xu, and C. S. Jensen, "Authentication of moving top-k spatial keyword queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 922 – 935, May 2015.

[22] R. Zhang, Y. Zhang, and C. Zhang, "Secure top-k query processing via untrusted location-based service providers," in *INFOCOM'12*, Orlando, FL, Mar. 2012.

[23] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, "Truth finding on the deep web: Is the problem solved?" in *VLDB'13*, Riva del Garda, Trento, Aug. 2013.

[24] T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen, "Authenticated data structures for graph and geometric searching," in *CT-RSA'03*, San Francisco, CA, Apr. 2003.

[25] L. Yiu, M. Lin, and K. Mouratidis, "Efficient verification of shortest path search via authenticated hints," in *ICDE'10*, Long Beach, CA, Mar. 2010.

[26] Z. Fan, Y. Peng, B. Choi, J. Xu, and S. S. Bhowmick, "Towards efficient authenticated subgraph query service in outsourced graph databases," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 696 – 713, Oct. 2014.

[27] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *INFOCOM'08*, Phoenix, AZ, Apr. 2008.

[28] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," in *INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.

[29] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," in *MobiHoc'09*, New Orleans, LA, May 2009.

[30] F. Chen and A. X. Liu, "SafeQ: Secure and efficient query processing in sensor networks," in *INFOCOM'10*, San Diego, CA, Mar. 2010.