A Generalized Framework for Preserving Both Privacy and Utility in Data Outsourcing

Shangyu Xie, Meisam Mohammady, Han Wang, Lingyu Wang, *Member, IEEE,* Jaideep Vaidya, *Fellow, IEEE,* and Yuan Hong, *Senior Member, IEEE*

Abstract—Property preserving encryption techniques have significantly advanced the utility of encrypted data in various data outsourcing settings (e.g., the cloud). However, while preserving certain properties (e.g., the prefixes or order of the data) in the encrypted data, such encryption schemes are typically limited to specific data types (e.g., prefix-preserved IP addresses) or applications (e.g., range queries over order-preserved data), and highly vulnerable to the emerging inference attacks which may greatly limit their applications in practice. In this paper, to the best of our knowledge, we make the first attempt to generalize the prefix preserving encryption via *prefix-aware* encoding that is not only applicable to more general data types (e.g., geo-locations, market basket data, DNA sequences, numerical data and timestamps) but also secure against the inference attacks. Furthermore, we present a generalized *multi-view outsourcing* framework that generates multiple *indistinguishable* data views in which one view fully preserves the utility for data analysis, and its accurate analysis result can be obliviously retrieved. Given any specified privacy leakage bound, the computation and communication overheads are minimized to effectively defend against different inference attacks. We empirically evaluate the performance of our outsourcing framework against two common inference attacks on two different real datasets: the check-in location dataset and network traffic dataset, respectively. The experimental results demonstrate that our proposed framework preserves both privacy (with bounded leakage and indistinguishability of data views) and utility (with 100% analysis accuracy).

Index Terms—Privacy, Prefix Preserving, Utility, Outsourcing

1 Introduction

With the significant development of cloud computing, an increasing number of data-related services such as data analysis and storage have been prevalently outsourced to the cloud [1]. In practice, outsourcing data analyses to service providers (e.g., the cloud) would request the data owner to share their original data. This may result in immense privacy concerns of the data owners with severe consequences for the enterprises [2]. As data leaking incidents become even more severe, a GDPR article [3] states that enterprises cannot share their sensitive data without sufficient protection, while acquiring the third-party services.

To date, different types of encryption algorithms may be applied to protect the outsourced data. First, encrypting the datasets using a traditional algorithm like AES [4] or 3DES [5] may prevent the external service providers from conducting useful data analysis, whereas Homomorphic encryption (including fully) [6], [7], [8] might be too expensive and inflexible for different analyses. The recent property preserving encryption schemes [9], [10], [11] have enabled service providers to perform efficient and accurate data analyses on the encrypted data, which are deterministic ciphertexts to retain a certain property of their plaintexts. Examples include hashing where the ciphertexts reveal the equality of messages, order preserving encryption (OPE) [11], [12], where the ciphertexts retain the ordering of data, and prefix preserving encryption (PrefixPE) [10], where the ciphertexts share the same length of prefixes as shared between the plaintexts. However, most existing property preserving encryption schemes [9], [10], [11] as mentioned above have the following two major limitations.

First, property preserving encryption is typically limited to specific data or applications. For instance, OPE is mostly applied in range queries based analysis on numerical data. While achieving more prefix-based utility (e.g., network trace analysis [13]), PrefixPE (e.g., CryptoPAn [10]) is only applicable to IP addresses, which is an important limitation since PrefixPE may potentially benefit a wide variety of outsourced data analyses on different datasets. As discussed in Section 2.2, besides IP addresses, some other data types (e.g., geo-location data, DNA sequences, market basket items, and timestamps) can be encoded with meaningful prefixes to retain very high utility in the encrypted data. For instance, in a dataset collected from location based services (LBS), two places which are close in the plaintexts (e.g., central park and the empire state building in New York) can be converted to ciphertexts (sharing the same length of prefix to maintain the same spatial distance) as two places in another city. Thus, it is highly desirable for a service provider to analyze the prefix preserving encrypted data.

Second, it is well known that most property preserving encryption techniques are vulnerable to various forms of inference attacks [14], [15], [16], [17], which attempt to link

- *S. Xie, H. Wang and Y. Hong are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA. Corresponding Author: Yuan Hong, E-mail: yuan.hong@iit.edu*
- *M. Mohammady is with Data61, CSIRO, Australia, affiliated to Cyber Security CRC*
- *L. Wang is with Concordia University, Montreal, QC, Canada.*
- *J. Vaidya is with the MSIS Department, Rutgers University, USA.*

Manuscript received February 12, 2020; revised March 2, 2021; accepted April 26, 2021. Date of publication xx; date of current version xx. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. xx

the encrypted data to original data with background knowledge or auxiliary data. As discussed in Section 3.1, due to the deterministic ciphertexts, PrefixPE (e.g., CryptoPAn [10]) is also vulnerable to the emerging inference attacks [14]. We have also conducted experiments on the inferences attacks in Section 7.1 to validate this limitation.

In this paper, we first propose a novel scheme to encode a variety of data types into bit strings (*prefix-aware encoding*), and then propose a framework for outsourcing different types of prefix preserving encrypted datasets by generalizing a multi-view approach [13] to significantly reduce the information leakage against inference attacks. Specifically, the proposed prefix-aware encoding converts various types of data into the prefix-aware data (viz. bit strings with prefix-based utility), and then the prefix preserving encryption, i.e., CryptoPAn [10] (originally on 32-bit IPv4 addresses or 128-bit IPv6 addresses) can be generalized to encrypt *any type of data that can be encoded into the bit strings with utility resulted from the preserved prefixes. Essentially, if any data is naturally hierarchical (e.g., IP addresses as bit strings) or can be indexed by a prefix-aware tree (e.g., location data, and market basket data. See Section* 2.2), the prefixes in the encoded bit strings could be preserved to ensure utility when directly analyzing the outsourced data. For instance, the distance between any two locations can be fully preserved in the outsourced data.

Furthermore, to address the inference attacks, the multi-view outsourcing framework will generate multiple *indistinguishable* data views in which one real data view fully preserves the utility (ensuring 100% accuracy while performing data analyses on the prefix preserved data), and its corresponding analysis result can also be privately retrieved.

Contributions. The primary contributions focus on the generalization of the CryptoPAn to broader data types and applications from two aspects. First, we revise the CryptoPAn by extending the input size of block cipher function (cryptographic building block) to encrypt any length of bit strings rather than fixed 32-bit IPv4 addresses or 128-bit IPv6 addresses. Second, we generalize the multi-view outsourcing framework [13] with the following major improvements: (1) encrypting multiple types of data with the new prefix-aware encoding scheme and the above generalized CryptoPAn (for any length of data), (2) incorporating more and stronger inference attacks [14], [18], [19], [20] into the threat model, and (3) generating a minimum number of data views given formally defined privacy leakage bound (say Γ-Leakage). Moreover, other contributions are summarized as below.

- To our best knowledge, we propose the first general purpose prefix encryption scheme, which can potentially be applied to a wide variety of data types and applications such as geo-locations, DNA sequences, market basket datasets, and timestamps.
- The generalized multi-view outsourcing framework provides an additional layer of protection that can make the vulnerable PrefixPE scheme (i.e., CryptoPAn) sufficiently secure against various inference attacks (e.g., [14], [18], [19], [20]).
- Besides privacy and utility guarantees, our approach offers negligible communication overheads, and the computational costs can be easily adjusted based on any bounded leakage w.r.t. inference attacks.
- We empirically evaluate the performance of numerous inference attacks [14], [18], [19], [20] on the PrefixPE encrypted data using real datasets (the *check-in location* dataset and the *network traffic* dataset) and our generalized multi-view outsourcing framework. The experimental results demonstrate that our proposed framework preserves both privacy (with bounded leakage and indistinguishability of data views) and utility (with 100% analysis accuracy).

The rest of this paper is organized as follows. Section 2 presents the prefix-aware encoding scheme for different data types and the corresponding data analysis. Section 3 illustrates the inference attacks on PrefixPE (i.e., CryptoPAn), the threat model for our outsourcing framework. Section 4 overviews the generalized multi-view outsourcing framework and the privacy properties. Section 5 gives the details of the generalized outsourcing framework and theoretical analyses. Section 6 discusses relevant issues. Section 7 presents the experimental results. Section 8 reviews the related work. Finally, Section 9 concludes the paper.

2 PrefixPE and Prefix-aware Encoding

2.1 Generalized CryptoPAn

As a PrefixPE scheme, CryptoPAn [10] was originally designed to generate deterministic ciphertexts for IP addresses (32-bit ciphertexts for IPv4 and 128-bit ciphertexts for IPv6). We first generalize CryptoPAn for any n-bit data as below:

Definition 2.1 (Generalized Prefix Preserving Encryption [10])**.** *Given two n-bit strings $a = a_1a_2a_3...a_n$ and $b = b_1b_2b_3...b_n$, if a and b share a k-bit ($0 \leq k \leq n$) prefix, we have $a_1a_2...a_k = b_1b_2...b_k$ and $a_{k+1} \neq b_{k+1}$. An encryption function $f(\cdot) : \{0,1\}^n \rightarrow \{0,1\}^n$ is said to be prefix-preserving, if $f(a)$ and $f(b)$ also share a k-bit prefix.*

To encrypt each bit, CryptoPAn applies a cryptographic function (including padding, a block cipher function such as Rijndael [4] with a 256/128-bit key K, and the least significant bit function) to the bits, and then XOR with the current bit to ensure the prefix preserving property [10].

Theorem 2.1. *CryptoPAn has the following two properties [10]: (1) associative property: given an n-bit string $a \in \{0,1\}^n$,*

$$1 \leq i, j \leq n : f^j(f^i(a, K)) = f^{(i+j)}(a, K) \tag{1}$$

and (2) inverse property: given two n-bit strings a and b,

$$\text{if } f(a, K) = b, f^{-1}(b, K) = a \tag{2}$$

Similar to $f(\cdot)$, for each bit in b, the inverse CryptoPAn $f^{-1}(\cdot)$ applies the same cryptographic function to the bit's corresponding prefix in a (which was computed while applying $f^{-1}(\cdot)$ to the previous bits) and XOR with the current bit.

Theorem 2.1 can be directly proven with the extension from 32-bit IP addresses in [13] to generalized CryptoPAn.

2.2 Prefix-aware Encoding

We can directly apply CryptoPAn to encrypt the sensitive IP addresses [10], [13] for outsourcing the network traffic since IP addresses automatically hold the prefix property

(32-bit IPv4 addresses or 128-bit IPv6 address
of preserving prefixes in the encrypted IP
be realized for many analyses since the c
preserve all the subnet structure of the origina
a prefix in the original IP addresses also res
the same length of prefix in the encrypted IP

Motivated by such prefix properties, ma
of datasets can also be encoded into prefix-a
as the geo-location data [21], DNA sequence
market baskets [23], numerical data [24], ar
[25] (ensuring utility for performing differe
the encrypted data) using a prefix-aware tree

Definition 2.2 (Prefix-aware Tree)**.** *Given the*
generate a balanced tree in which nodes (from
signify a sequence of prefixes, all the sibling node
prefix. Then, each value in the domain can be re
bits concatenated from the top (except the root) to

Example 2.1 (Prefix-aware Tree for IPv4)**.** *IPv*
to form the 2^{32} *addresses, thus its prefix-aware tre*
tree with 33 levels (including the root node), where each IP address
is formed by concatenating the bits from the top to each leaf node.

In the following, we will discuss the prefix-aware encoding for some representative data types, and illustrate the corresponding prefix preserving data analysis.

2.2.1 Locations in Location-based Services (LBS)

The location data usually include the 2-dimensional latitude and longitude coordinates of different places, which are highly precise float numbers (up to 8 decimal digits) for representing the locations in map applications, e.g., Google Map and Bing Map. In the *Bing Map Tiles System* [26], the map is recursively divided into four tiles equally to reach the required resolution for users to quick map zoom in/out. Specifically, given the resolution, the system can map the longitude and latitude coordinates to bit strings called *quad key*, which is uniquely represented as the index of tile for the coordinates (can be used for map image retrieval).

Motivated by such hierarchical structure, we encode the coordinates into bit strings by concatenating the index of each level for one specific location. As shown in Figure 1, there is a root node at the top. At each level, the four children of each node can be encoded using two bits $00, 01, 10, 11$ to represent four tiles, respectively. Thus, after concatenating the bits from the first level, every location can be encoded by a leaf node, and all the locations can be encoded with the same length of bits if the coordinates use the same precision. Location precision can be increased with additional levels and longer bit strings. Our experimental location data utilizes a length of 46 bits (23 levels) with a ground resolution 4.78m$\times$4.78m (tile) at the finest level, which is sufficiently accurate for precise location coordinates.

Prefix Preserving Data Analysis for LBS. As discussed above, for the encoded bit strings of coordinates, utility can be fully preserved for data analysis since prefixes can be preserved in the encrypted locations (while preserving the privacy). For instance, "central park" and "the empire state building" in New York share a prefix, and the encrypted data for these two locations should also share the same

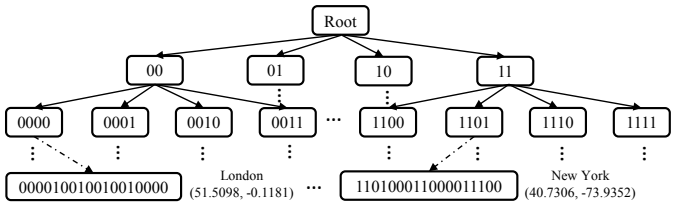


Fig. 1. A Prefix-aware Tree for Location Data

length of prefix (e.g., might be two other places in London with the same proximity). Thus, the structure of the locations and the distance between such locations (besides other features such as frequency) can be preserved in the outsourced data. Besides the basic queries (e.g., counting), the location data analysis which uses distance-based metrics can achieve the same utility while analyzing the encrypted prefix preserving data, e.g., user mobility prediction [21].

Thus, the PrefixPE on the encoded locations can fully preserve the utility in the general data analyses for LBS. We have conducted experiments to validate such utility with prefix preserving encrypted location data in Section 7.3.

2.2.2 Market Basket Dataset [23]

This dataset includes the purchased items by the customers. With the item generalization hierarchy [23], we can encode the items into the bit strings with the same length. Similar to the location hierarchy (assigning 2 bits to each level since it is partitioned into 4 blocks), in each level of the item hierarchy, we assign $\log(n)$ bits to the n generalized items. The bit string of each item can be encoded by concatenating all the bits assigned in each level. The PrefixPE can fully preserve the item generalization path (e.g., the encrypted values of "apples" and "pears" also share a long prefix to make them close). With such prefix-aware encoding, data mining applications (e.g., frequent itemset mining [27]) can be performed on the prefix preserving encrypted data. Thus, PrefixPE on the encoded data/items can fully preserve the utility of the market basket data in the related data mining.

2.2.3 Numerical Data [24] and Timestamps [25]

Numerical data and timestamps are commonly included in a wide variety of datasets, e.g., network traffic [13], transaction data [28], and search logs [29]. Such values in different datasets can be simply converted into bit strings with meaningful prefix (aka. the converted binary numbers). Encrypting such bit strings with PrefixPE can also fully preserve the prefixes in numerical values and daytime formats (e.g., "yyyy-mm-dd hh:mm:ss[.nnnnnnnnn]") for outsourcing. The proximity between any two values of these two data types can be preserved in the ciphtertexts (but not preserving the order of them). For instance, the ciphertexts of two close numerical values (or timestamps) also possess the same degree of proximity. Since the order of two numerical values and timestamps cannot be fully preserved, PrefixPE can preserve the partial utility in the encoded data of numerical and timestamp data.

2.2.4 Genome Dataset [30]

Genomic features (e.g., DNA sequences) of different organisms are studied to significantly advance the biological and

Data Type	IP Data	Locatio
Encoding	Default (IPv4/6)	4 Tiles (00,
PrefixPE Utility	Fully	Ful

medical research. For instance, DNA sequen
order of adenine (A), guanine (G), cytosine (C
(T). Thus, each of A, G, C, T can be enco
bits, and the sequence can be concatenate
bit in order. In the encrypted DNA sequenc
e.g., Private Edit Distance [22], PrefixPE on
data can preserve the prefixes (order of nu
shared prefixes), but cannot preserve the ful
Hamming distance) between two random se
PrefixPE using such simple encoding can pa
the utility on the genome data. We will als
encoding schemes in conjunction with Pr
preserve the utility of the DNA sequences in the future.

2.2.5 Summary

Table 1 summarizes the prefix-aware encoding for some representative data and the utility preservation in the generic analyses. Note that the prefix-aware encoding can be tailored with different prefix definitions if necessary.

3 Security Models

3.1 Inference Attacks

The inference attacks on the property preserving encryption schemes [13], [14], [31], [32], [33], [34] have been extensively studied recently. We then introduce two typical inference attacks on the PrefixPE encrypted data, and briefly discuss other inference attacks [32], [33], [34].

Inference Attacks using Frequency and ℓ_p-Optimization. Frequency analysis [18] is the most well known inference attack with the auxiliary background knowledge. Extended from frequency analysis, ℓ_p-optimization [14] is further utilized to form a family of attacks that maximally infer the original data from property preserving encrypted data (e.g., order preserving encryption (OPE) and deterministic encryption (DTE)). Such attacks find the most matches from ciphertexts to plaintexts by minimizing the ℓ_p distance between the histograms of the encrypted dataset and an auxiliary dataset (as the background knowledge). The auxiliary dataset may be obtained by the adversary from some public statistics or prior versions of the original dataset. We explain such attacks on the encrypted location data as below:

Example 3.1 (Frequency and ℓ_p-Optimization based Inference Attacks [14], [18])**.** *As shown in Figure 2, the geo-locations can be encoded to bits (fully preserving the utility with the preserved prefixes, see Section 2.2) which are represented in hex format (for shortening the notations). The adversary may simply get some auxiliary information from publicly known data sources, e.g., top 50 most popular spots in New York. Then, the adversary can match the auxiliary location list (sorted by frequency) with the ciphertext of the locations also sorted by frequency [18].*

Inference Attacks using Fingerprinting. Adversaries can perform strong inference attacks by injecting data into the original data collection and fingerprinting the records in the

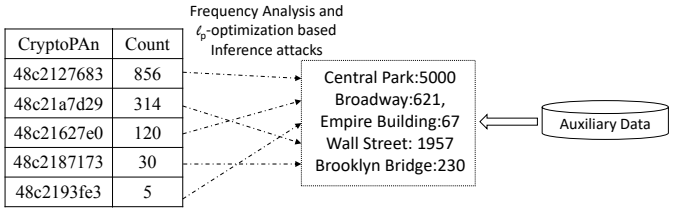


Fig. 2. Frequency and ℓ_p-Optimization

encrypted data [19], [31], [35]. Then the adversary could identify his/her own data from the encrypted data via the combination of other information, e.g., the timestamps and frequencies of the injected checked-in locations, the timestamps, port numbers and protocols of the network traffic data (with IP addresses). Then, the adversary can obtain a set of matches between the original data and the encrypted data, and eventually learn the prefixes of other values that share the prefix with the identified prefixes.

Example 3.2 (Fingerprinting based Inference Attacks [19], [20])**.** *Some detailed information of the same check-in location dataset in Example 3.1 are given in Figure 3. The leftmost table shows such real world dataset, the table in the middle gives the*

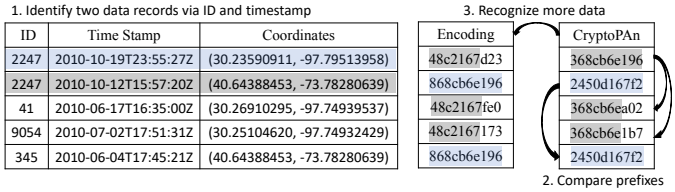


Fig. 3. Fingerprinting based Inferences

The adversary can inject his/her tuples in such real data (e.g., using the application with its account). Then, he/she can identify two of his/her encrypted bit strings (w.r.t. his/her two location records) via other original information which are retained for utility (e.g., combination of timestamps and pseudonyms). Then, he/she can infer more prefixes and subprefixes by comparing the identified prefixes with other encrypted locations (bit strings). For instance, as "368cb6e196" and "2450d167f2" in the first two rows are known (the adversary knows the location he/she has visited at that time), the location data of the first row ("368cb6e196") also shares 28-bit prefix ("368cb6e", 7 digits in hex) with the location in the 4th row. Similarly, the 6th record can also be breached since the encrypted bit string is identical to "2450d167f2".

Inference Attacks and Defense. Some other inference attacks [32], [33], [34] have been recently identified for OPE, such as exploiting the correlations among attributes [32] and learning query pattern access via statistical learning [34]. All these inference attacks (including the two detailed in the above examples) share some similarities, e.g., identifying the similar patterns from the original data and the encrypted data. We will propose a generalized multi-view outsourcing

framework which effectively obfuscates such patterns. The details are given in Section 4 and 5.

3.2 Threat Model and Privacy Notion

In our framework, the data owner outsources its encrypted data to the service provider, which is assumed to be *honest-but-curious*. The service provider has possessed background knowledge to implement a given set of inference attacks [13], [14], [19], [20] (*can be any emerging inference attacks on the property preserving encryption*). Moreover, the adversary is assumed to know the set of attributes in the outsourced data, and the domain for the attributes. We also assume that all the communications are in secure channel.

Then, we present the privacy notion in the threat model. Since the degree of privacy is quantified w.r.t. different inference attacks, we first formally give the following definition.

Definition 3.1 (Inference Attack Function)**.** *Function $\mathcal{I}(Enc(\mathcal{D}), \{\alpha\})$ is defined to quantify the leakage derived from performing a given set of inference attacks on encrypted data $Enc(\mathcal{D})$ with a set of background knowledge parameters $\{\alpha\}$.*

Example 3.3. *Assume that the adversary pose two kinds of inference attacks: (1) frequency and ℓ_p-optimization based inference attacks, and (2) fingerprinting based inference attacks, with two background knowledge parameters α_f, α_s, respectively:*

- *the adversary holds a public auxiliary dataset including any α_f of the original data (percent between 0% and 100% of the domain) and their similar count distribution.*
- *the adversary has already identified any α_s of the original data (percent between 0% and 100% out of the domain) via injecting data before encryption.*

As a result, the attack function $\mathcal{I}(Enc(\mathcal{D}), \{\alpha_f, \alpha_s\})$ returns the information leakage, which is defined as below:

Definition 3.2 (Information Leakage [10])**.** *The percent of bits in $\mathcal{D}$ (encoded as bit strings) inferred from the encrypted data $Enc(\mathcal{D})$ (including the partially inferred prefixes).*

Then, we bound the leakage derived from the inference attacks with the following privacy notion.

Definition 3.3 (Γ-Leakage)**.** *While performing a given set of inference attacks on the encrypted data $Enc(\mathcal{D})$ (generated by encrypting the original data $\mathcal{D}$ using PrefixPE), the information leakage is upper bounded by $\Gamma \in [0, 1]$.*

4 System and Privacy Properties

In this section, we present the system and privacy properties. Table 4 summarizes the frequently used notations.

4.1 System Model

As illustrated in Figure 4, the proposed outsourcing framework involves two entities: (1) data owner: the party owns the original data and outsources the *prefix preserving encrypted data* to the service provider for data analysis, and (2) service provider: a cloud platform or an external company who provides data analysis services, which might intend to infer the original data from the outsourced encrypted data.

$\mathcal{D}$	original data
A	number of distinct values in $\mathcal{D}$
$\mathcal{D}_p$	prefix-aware data
$\mathcal{D}_0$	initially encrypted data
$\mathcal{D}_s$	seed data
$\mathcal{D}_1 \dots \mathcal{D}_N$	N generated data views
α_f	background knowledge parameter on frequency and ℓ_p-optimization based inference attacks
α_s	background knowledge parameter on fingerprinting based inference attacks
$f^r(\cdot)$	executing r times CryptoPAn
$\mathcal{I}(\cdot)$	inference attack function: returns the leakage derived from the encrypted data in the attacks
$p(x)$	number of partitions with x-bit shared prefix
$\mathbb{R}$	a pseudorandom matrix
$G_1 \dots G_N$	vectors for generating data views
K_0	non-shared key for CryptoPAn
K_1	outsourced key for CryptoPAn
P_i	data partition i

Specifically, Mohammady et al. [13] proposed a multi-view approach to defend against inference attacks on the PrefixPE encrypted data. The core idea is to hide the real data view among other fake data views, where the service provider cannot identify the real data view. However, such approach is only limited to network traffic data. We make the following key improvements on the multi-view outsourcing:

- generalize the outsourcing framework to privately outsource a wide variety of data types (any length of the data vs. 32/128-bit IP addresses).
- generalize the defense (bounded leakage) against any given set of inference attacks (to make the illustration more concrete, we conducted experiments on more inference attacks, compared to [13]).
- improve the privacy guarantee (by specifying a privacy bound Γ-Leakage) and computational overheads on analyzing the outsourced data (minimizing the number of views while satisfying Γ-Leakage against a given set of inference attacks).

As shown in Figure 4, the generalized multi-view outsourcing framework is detailed as follows:

4.1.1 At the Data Owner

(1) **Prefix-aware Encoding:** encodes the sensitive attributes of the original data (e.g., locations, IP addresses, set of items) to prefix-aware data $\mathcal{D}_p$ with prefix-aware encoding, as discussed in Section 2.2.

(2) **Partitioning and Generating Seed Data:** initially encrypts the encoded data $\mathcal{D}_p$ to $\mathcal{D}_0$ using CryptoPAn with a non-shared CryptoPAn key K_0 (for preventing the brute force attack), and then partitions $\mathcal{D}_0$, as well as generates a seed data $\mathcal{D}_s$ by executing CryptoPAn in each partitions with a random number of iterations using another key K_1. This step obfuscates the data in the seed data $\mathcal{D}_s$, which is safe to share and can be used to generate multiple data views by the service provider (note that K_1 is shared to the service provider for generating the data views). See details in Section 5.1.

(3) **Outsourcing:** the Seed Data $\mathcal{D}_s$, the key K_1 and some other required parameters for generating multiple data views will be outsourced to the service provider.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3078099, IEEE Transactions on Knowledge and Data Engineering

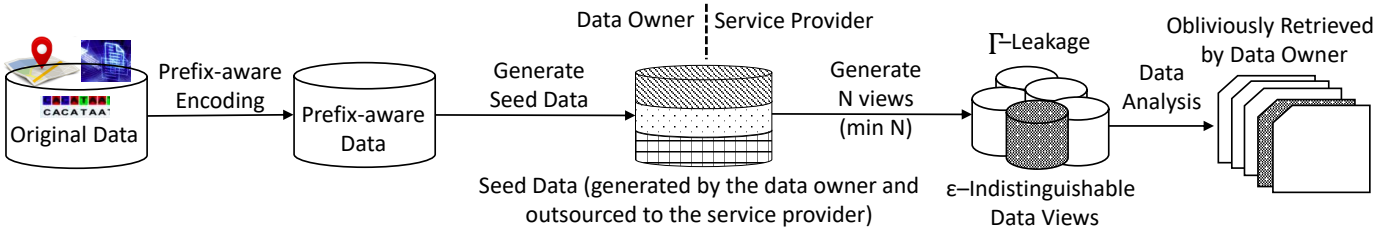


Fig. 4. Generalized Outsourcing Framework on the Multi-view Approach [13]

4.1.2 At the Service Provider

(4) **Generating N Data Views:** generates N data views using $\mathcal{D}_s$, the CryptoPAn key K_1 and other parameters, where *exactly one* out of N data views is the real data view (which fully preserves the prefixes).

(5) **Analyzing N Data Views:** performs the required data analysis on the N data views (note that N is minimized for satisfying Γ-Leakage). The real data views fully preserves the prefixes with the 100% analysis accuracy.

The details are given in Section 5.2.

4.1.3 Both Data Owner and Service Provider

(6) **Obliviously Retrieving Analysis Result:** the data owner leverages an oblivious random access memory (ORAM) protocol [36] to retrieve the correct analysis result out of N results while the service provider does not know which result is retrieved by the data owner.

4.2 Privacy Guarantees

We now summarize two desired privacy properties against the inference attacks in semi-honest model.

4.2.1 Indistinguishability

The proposed framework first ensures that N data views are *indistinguishable* (inspired from differential privacy [37]):

Definition 4.1. *[13] The generated N data views are ϵ-indistinguishable against inference attacks if and only if*

$$\exists\, \epsilon \geq 0,\ s.t.\ \forall i, j \in \{1, 2, \cdots, N\} \Rightarrow \\ e^{-\epsilon} \leq \frac{Pr[\textit{data view } i \textit{ may be real }]}{Pr[\textit{data view } j \textit{ may be real}]} \leq e^{\epsilon} \tag{3}$$

As ϵ is smaller, the N different data views would be more indistinguishable. As a result, a smaller number of fake data views will be filtered out by the adversary. For instance, $e^{-\epsilon} = 0.9$ means that at least 90% of the fake data views are indistinguishable from the real one. We give the indistinguishability analysis in Section 5.2.4 and experimentally validate that ϵ is quite small in Section 7.2.

4.2.2 Bounded Leakage against Inference Attacks

Besides ϵ-indistinguishability, our framework can ensure that all the encrypted data held by the adversary satisfy Γ-Leakage (Definition 3.3) as an additional layer of protection:

$$\mathcal{I}(\mathcal{D}_s, \mathcal{D}_1, \ldots, \mathcal{D}_N, \{\alpha\}) \leq \Gamma \tag{4}$$

where the adversary performs the inference attacks with background knowledge $\{\alpha\}$ on the seed data $\mathcal{D}_s$ (received by the service provider) and N different data views $\mathcal{D}_1, \ldots, \mathcal{D}_N$ (generated by itself). In the outsourcing framework, the information leakage is bounded by a very small Γ, as experimentally validated in Section 7.

Remark. The first privacy notion ϵ-indistinguishability is defined to measure how indistinguishable the generated N data views can be (hiding the real data view). Indeed, all the fake data views and one real data view could possibly leak information (though they are indistinguishable) if the adversary is armed with background knowledge to perform inference attacks. Thus, another privacy notion Γ-Leakage is defined to bound the overall information leakage in all the data obtained and generated by the adversary. Two privacy notions measure different aspects of the privacy and complement each other in the generalized multi-view outsourcing framework.

5 Generalized Framework Design

5.1 Prefix-based Partition

The initially encrypted data $\mathcal{D}_0$ is partitioned by assigning all the values sharing at least x-bit prefix into the same partition. Specifically, given $\mathcal{D}_0$, the data (e.g., location) has been encoded into L bits. We first traverse $\mathcal{D}_0$ to get the set of distinct values. Given the prefix length $x \in [1, L]$, we generate a mapping set by grouping all the values sharing length-x prefix to one subset. Finally, we obtain the partitions in $\mathcal{D}_0$ using the mapping set. Denote the number of partitions in $\mathcal{D}_0$ created with length-x prefix as $p(x)$. Thus, we have partitions P_i with length-x prefix $d_i, i \in [1, p(x)]$.

This prefix-based partitioning scheme can potentially result in the identification of the real data view by the adversary due to the collision property of CryptoPAn [10]. In [13], the identification of the real data view was proposed by the collision of the encrypted full IP addresses. Indeed, there also exists subprefix collision of the prefix which can possibly help identify the real data view. We generalize such attack and name it as *Subprefix Collision Attack*, which is caused by similar prefixes or subprefixes ("close prefix").

Closeness of Prefixes. The data in the same partition share a common prefix (length-x), the prefixes of two different partitions may also share a length-y subprefix where $y < x$. We use the following measure to define such relationship between such two partitions (with *close* prefix).

Definition 5.1 (β-closeness)**.** *While partitioning $\mathcal{D}_0$ using the shared length-x prefix, given two prefixes d_i and d_j where $i, j \in [1, p(x)]$, if d_i and d_j also share a length-y subprefix ($y < x$) such that $|x - y| \leq \beta$, the two partitions P_i and P_j are said to satisfy β-closeness (or P_i and P_j are β-close).*

As mentioned before, applying CryptoPAn in real data view would also preserve such *closeness* relationship across

different partitions in $\mathcal{D}_0$, which may cause *subprefix collision attack*: the real data view will preserve all the prefixes (including subprefixes) among all the partitions whereas the fake data views would not retain them (see Example 5.1).

Example 5.1 (Subprefix Collision Attack)**.** *As shown in Figure 5, the original data are encoded into prefix-aware bit strings and represented in hexadecimal (for simplicity of notations). The prefix length of the partition is $x = 20$ bits (dash line) in binary (5-digit in hexadecimal). The original data is divided into two partitions, P_1 with prefix 38456 and P_2 with 38457, both of which share a common subprefix of length 19; only the the least significant bit (LSB) is different (i.e., 0 and 1) – these two partitions satisfy 1-closeness. The real data view can be readily distinguished from other data views with the prefix 27c27 and 27c26, which still keep the subprefixes across partitions. Considering that CryptoPAn has collision-resistant property [10], the probability that every fake*

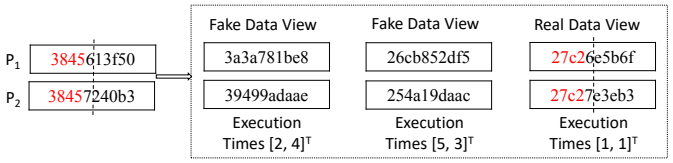


Fig. 5. Subprefix Collision Attack

Notice that, β-closeness has the following characteristics for this subprefix collision attack:

1) for a small β (different partitions share a longer subprefix), if many β-close partitions are identified, then the real data view can be simply identified;
2) for a large β (short subprefix; an extreme case, many partitions share the first bit), then the β-close partitions would not increase the confidence of such attack.

To tackle such attacks, the proposed scheme aims to create more similar collisions among β-close partitions while generating data views, thus the adversary cannot identify the real view from the subprefix collision (see Section 5.2).

5.2 Multiple Data Views Generation

5.2.1 Seed Data for Generating Views

The objective of partitioning $\mathcal{D}_0$ is to *obfuscate* the encrypted data across different partitions in the outsourced data but still be able to reconstruct the encrypted data with fully prefix preservation ($\mathcal{D}_0$ or similar data) for analysis. Thus, the next step is to generate the "Seed Data" which is safe to outsource (obfuscated) and oblivious to reconstruct a real data (fully prefix preserved).

Specifically, the framework applies CryptoPAn $f(\cdot)$ to different partitions in $\mathcal{D}_0$ with random number of iterations to generate the seed data $\mathcal{D}_s$. Given shared prefix length x, there are $p(x)$ partitions, $\mathcal{D}_0 = \{P_1, P_2, ..., P_{p(x)}\}$. Then, we define a random vector $G_0 = [v_1^0, v_2^0, ..., v_{p(x)}^0]^T$, where entry $v_i^0, i \in [1, p(x)]$ is an integer representing the number of iterations applying CryptoPAn on the partition P_i. Thus, we have $f(\mathcal{D}_0, G_0, K_1) = [f^{v_1^0}(P_1), f^{v_2^0}(P_2), \ldots, f^{v_{p(x)}^0}(P_{p(x)})]$, where K_1 is the CryptoPAn key for obfuscating the data across $p(x)$ partitions in the seed data $\mathcal{D}_s$ (the same key will be used for reconstructing prefix preserving data). Note that $v_i^0, i \in [1, p(x)]$ can be chosen from the domain $[-h, h]$ where h is comparable to $p(x)$.

5.2.2 N Data Views

As discussed in Section 4.1, generating multiple data views (say N) which include only one real data view (fully prefix preserving) and $N-1$ fake data views (non prefix preserving across partitions) could mitigate the leakage against inference attacks (*since the probabilities of matching the encrypted data to true values can be greatly reduced by providing more data views*). This is experimentally validated in Section 7.

To this end, the service provider generates N different data views based on the seed data and N pseudorandom vectors (similar to the procedure of generating the seed data $\mathcal{D}_s$). The data owner generates N pseudorandom vectors which form a matrix $\mathbb{R} = [G_1, \ldots, G_N]$, and then outsources the seed data $\mathcal{D}_s$, pseudorandom matrix $\mathbb{R}$, and the CryptoPAn key K_1 used for generating N views.

Definition 5.2 (data view)**.** *Given pseudorandom vectors $G_i = [v_1^i, v_2^i, \ldots, v_{p(x)}^i], i \in [1, N]$ in the matrix $\mathbb{R}$, the ith data view can be represented as:*

$$\mathcal{D}_i = [f^{\sum_{j=0}^{i} v_1^j}(P_1), f^{\sum_{j=0}^{i} v_2^j}(P_2), \ldots, f^{\sum_{j=0}^{i} v_{p(x)}^j}(P_{p(x)})]$$

With the associative and inverse property of CryptoPAn encryption $f(\cdot)$, the ith data view can be the prefix preserving if the entries in the pseudorandom vectors satisfy:

$$\sum_{j=0}^{i} v_1^j = \sum_{j=0}^{i} v_2^j = \cdots = \sum_{j=0}^{i} v_{p(x)}^j \tag{5}$$

In other words, if the aggregated pseudorandom vectors for the ith data view $G_0 + G_1 + \cdots + G_i$ have equivalent entries, the ith data view can be prefix preserving (real data view); otherwise, not prefix preserving (fake data view). If $G_0 + G_1 + \cdots + G_i = 0$, then the real data view would be the initially encrypted data $\mathcal{D}_0$. Note that only one prefix preserving data is necessarily generated out of N data views (for reducing the probability of identifying it and the leakage). Since the data owner generates all the pseudorandom vectors $G_0, G_1, \ldots, G_N$ and the first vector G_0 (for generating the seed data) is not shared to the service provider, the service provider would not know when Equation 5 holds for generating the real data view.

Mitigate Subprefix Collision Attacks. As discussed before, our multi-view outsourcing creates more collisions among these prefixes which have common subprefixes (in β-close partitions) to address the subprefix collision attack. Specifically, when generating the pseudorandom number of executions of CryptoPAn on the partitions, we can generate the same execution times (aggregated pseudorandom) for the β-close partitions in the fake data views while generating different aggregated pseudorandom numbers for partitions with different subprefixes or β is too large (collisions may naturally occur in this case). Thus, the random matrix $\mathbb{R}$ $(G_1, \ldots, G_N)$ to determine the execution times for each data view is generated as below:

1) the data owner first generates a random vector G_0 with the size $p(x)$. Then the data owner will determine the minimum N as illustrated in Section 5.2.3.

2) the data owner then generates N pseudorandom vectors (as $p(x) \times N$ matrix). $r \in [1, N]$ is the randomly generated index for the real data view (*only data owner knows*), and the generated pseudorandom vector $G_r, r \in [1, N]$ satisfies Equation 5 (e.g., $\sum_{i=0}^{r} G_i = [0, 0, ..., 0]^T$) to preserve all the original prefixes.
3) finally, the data owner generates a set of $N-1$ pseudorandom vectors for the fake data views. Recall that we expect to create as much collisions among the partitions as possible (for also satisfying β-closeness in the fake data views). Each aggregated vector of G_0 and $G_i, i \in [1, N]$ (e.g., $G_0 + G_1, G_0 + G_1 + G_2, \ldots, G_0 + G_1 + \cdots + G_N$) should have at least two equal execution times for each pair of β-close partitions.

Input : x, N, β, G_0
Output: pseudo random matrix $\mathbb{R} = [G_1, \ldots, G_N]$
Generate the set of prefixes with length-x
Group the β-close partitions (a reasonable β)
for $i = 1 : n$ **do**
  if $i \neq r$ **then**
    generate a length-$p(x)$ pseudorandom vector G_i such that the vector $\sum_{j=0}^{i} G_i$ includes two identical values if the corresponding two partitions are β-close
  if $i = r$ **then**
    generate a length-$p(x)$ pseudorandom vector G_i: the entries in $\sum_{j=0}^{i} G_i$ are identical

Algorithm 1: Pseudorandom Matrix Generation

Algorithm 1 gives the details for generating such pseudorandom vectors/matrix. Example 5.2 illustrates how such pseudorandom matrix can address the subprefix collision attacks and hide the real data view.

Example 5.2. *Figure 6 shows 5 partitions with 20-bit prefixes. The initial random vector $G_0 = [-2, 2, 3, 0, -1]^T$, and the pseudorandom vectors $G_1 = [3, 0, -2, 2, 3]^T, G_2 = [2, 1, 2, 1, 1]^T$, and $G_3 = [1, 1, -1, -2, -2]^T$ are generated by the data owner. $\mathcal{D}_2$ is the real data view while $\mathcal{D}_1$ and $\mathcal{D}_3$ are the fake data view. G_0 will be held privately, only the pseudorandom matrix $\mathbb{R} = [G_1, G_2, G_3]$ are outsourced. Thus, we have:*

- *generating $\mathcal{D}_1$ (fake) executes CryptoPAn for $G_0 + G_1 = [1, 2, 1, 2, 2]^T$ times in 5 partitions, resp. ($G_1 = [3, 0, -2, 2, 3]^T$ times by the service provider).*
- *generating $\mathcal{D}_2$ (real) executes CryptoPAn for $G_0 + G_1 + G_2 = [3, 3, 3, 3, 3]^T$ times in 5 partitions, resp. ($G_1 + G_2 = [5, 1, 0, 3, 4]^T$ times by the service provider).*
- *generating $\mathcal{D}_3$ (fake) executes CryptoPAn for $G_0 + G_1 + G_2 + G_3 = [4, 4, 2, 1, 1]^T$ times in 5 partitions, resp. ($G_1 + G_2 + G_3 = [6, 2, -1, 1, 2]^T$ times by the service provider).*

Note that G_0 for 5 partitions are locally executed to generate the seed data $\mathcal{D}_s$ by the data owner, and the service provider cannot reconstruct G_0 from the received data.

Example 5.3. *In Fig 6, the seed data generates 3 data views. The two fake data views have at least 2 subprefix collisions, e.g., 368cb and 368c8 in fake data view $\mathcal{D}_1$ and 5a502 and 5a503 in fake data view $\mathcal{D}_3$. Thus, $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ are subjected to some indistinguishablity (which will be formally analyzed below).*

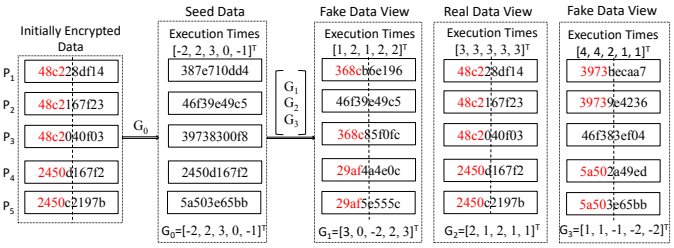


Fig. 6. Utilizing Pseudorandom Matrix to Generate 3 Data Views (negative execution times refer to repeating inverse CryptoPAn execution)

[3, 0, -2, 2, 3]T
[2, -2, -3, 0, 1]T
[5, 1, -1, 1, 2]T

5.2.3 Minimum N with Bounding Γ-Leakage

As shown in Equation 4, the leakage drawn from inference attacks is determined by background knowledge parameters $\{\alpha_s, \alpha_f\}$, the received seed data $\mathcal{D}_s$ and generated data views $\mathcal{D}_1, \ldots, \mathcal{D}_N$. As N grows, leakage can be smaller, but more data views should be generated for the same data analysis (more computation). Thus, our multi-view outsourcing seeks a minimum N while satisfying Γ-leakage.

More specifically, the prefix length x determines the partitions $P_1, \ldots, P_{p(x)}$. Then, given any $x \in [1, L]$, there exists a minimum N while bounding the leakage with Γ in our experiments (fixing $\{\alpha_f, \alpha_s\}$ for the background knowledge). As a result, before partitioning the data, the data owner can find an x such that the required minimum N for Γ-leakage is minimized – searching the x and minimum N takes $O(n \log(n))$ since the leakage derived from the fixed inference attacks is anti-monotonic on N.

5.2.4 Privacy Analysis

In practice, an adversary will exploit any related information (received data, background knowledge, etc.) to identify whether a data view is the real or fake one. Recall that only the sensitive attributes (prefix-aware encoded) are encrypted with CryptoPAn while other attributes are identical among all the data views. Thus, identifying the real data view only depends on the encrypted partitions, such as comparing N different data views and the leakage derived from them via inference attacks.

Theorem 5.1. *The N data views (generated by the service provider) satisfy ϵ-indistinguishability where*

$$\epsilon = \ln[\frac{\sum_{k=1}^{b}(\prod_{k=1}^{p(x)\alpha_s} |P_k|)(p(x)\alpha_s)!}{\prod_{j=0}^{A\alpha_f + p(x)\alpha_s - 1}(A-j)}] \quad (6)$$

, $b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$ and A is the number of distinct values in $\mathcal{D}$.

Proof. The adversary is armed with α_f knowledge of the original data for the frequency and ℓ_p-optimization attacks, and the α_s knowledge for the fingerprinting attacks. Then, we derive the indistinguishability bound ϵ as follows.

Recall that we generate $p(x)$ partitions in $\mathcal{D}_0$ based on the prefix length x. Denote the cardinality of each partition P_i as $|P_i|, i \in [1, p(x)]$. Then, the total number of possibilities (denoted as O) of dividing A distinct values into the $p(x)$ partitions is: $O = \frac{A!}{|P_1|!|P_2|!\cdots|P_{p(x)}|!}$.

Next, all the possible outcomes of the real data views for the adversary (denoted as T) depends on it's armed

background knowledge (α_s, α_f), we thus need to consider the following two aspects: (1) the adversary can reconstruct $A * \alpha_f$ out of A distinct values by ℓ_p-optimization based inference attacks while these compromised $A * \alpha_f$ distinct values are possibly across $p_f \in [1, A * \alpha_f]$ partitions, which will eleminate a number of partitions for at most $p(x) - p_f$; (2) the adversary matches the remaining partitions with the $p(x)\alpha_s$ inferred prefixes via the fingerprinting based inference attacks. Thus, we have the following equation:

$$T = \frac{\sum_{k=1}^{b}(\prod_{k=1}^{p(x)\alpha_s}|P_k|)(A*(1-\alpha_f)-p(x)\alpha_s)!(p(x)\alpha_s)!}{|P_1|!|P_2|!\cdots|P_{p(x)}|!} \tag{7}$$

where $b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$.

Finally, we thus have:

$$\begin{aligned}\forall i,j \in \{1,2,\cdots,N\}, \frac{Pr[\text{data view } i \text{ may be real }]}{Pr[\text{data view } j \text{ may be real}]} &= \frac{T}{O} \\ &= \frac{\sum_{k=1}^{b}(\prod_{k=1}^{p(x)\alpha_s}|P_k|)(p(x)\alpha_s)!}{\prod_{j=0}^{A\alpha_f+p(x)\alpha_s-1}(A-j)}\end{aligned} \tag{8}$$

where $b = \frac{(p(x)-p_f)!}{(p(x)\alpha_s)!}$, $k \in [1, p(x)]$. Per Definition 4.1, we can complete the proof. □

In Section 7.2, the experiments on the real datasets show $\epsilon \leq 1.5$ in general. Furthermore, the overall information leakage is also upper bounded as below.

Theorem 5.2. *Given the attacker with background knowledge $\{\alpha_f, \alpha_s\}$, the information leakage from the seed data and N data views ($\mathbb{D} = \{\mathcal{D}_s, \mathcal{D}_1, \ldots, \mathcal{D}_N\}$) satisfies the Γ-leakage where*

$$\mathcal{I}(\mathbb{D}, \{\alpha_f, \alpha_s\}) \leq \Gamma = \frac{\alpha_f}{N} + \frac{x(\sum_{\forall P_k \in C}|P_k| - A\cdot\alpha_f)}{A\cdot L\cdot N} \tag{9}$$

where A is the number of distinct values in $\mathcal{D}$, L is the length of the encoded bit strings, x is the prefix length used for partitioning, C is the union of the two sets of data partitions derived by the inference attacks with background knowledge $\{\alpha_f, \alpha_s\}$, and $\forall P_k \in C, |P_k|$ is the number of distinct values in partition P_k.

Proof. The adversary is armed with α_f knowledge of the original data for the frequency and ℓ_p-optimization attacks, and the α_s knowledge for the fingerprinting attacks. According to Definition 3.3, to derive the upper bound Γ of information leakage, we consider the worst case scenario by assuming that the unique data or prefixes inferred by two types of inference attacks are disjoint (to derive the highest leakage). Note that we also assume that the adversary attacks all the data $\mathbb{D}$ (seed data and N data views) and the adversary does not know which data view is the real one (indistinguishability). Then, we compute the leakage on the two types of inference attacks, respectively.

As depicted before, the adversary can reconstruct $A\cdot\alpha_f$ out of A distinct values by ℓ_p-optimization based inference attacks while the adversary can matches $p(x)\cdot\alpha_s$ inferred prefixes by the fingerprinting based inference attacks. Then, we get the bits of information leakage (as percents) by ℓ_p-optimization based inference attacks: $A\alpha_f \cdot L + x(\sum_{\forall P_k}|P_k| - A\cdot\alpha_f)$, where $\forall k \in [1, |A\alpha_f|], |P_k|$ are the number of distinct values across all the $A\cdot\alpha_f$ partitions. Similarly, we can compute the bits of information leakage (as percents) by the fingerprinting based inference attacks $\sum|P_j|\cdot x$, where $\forall j \in [1, |p(x)\alpha_s|], |P_j|$ are the number of distinct values across all the $p(x)\alpha_s$ partitions. To sum up, we can get the leakage (the percent of bits inferred by the adversary, Definition 3.2) as below:

$$\frac{A\alpha_f\cdot L + (\Sigma|P_k| - A\alpha_f)\cdot x + \Sigma|P_j|\cdot x}{N\cdot A\cdot L} \tag{10}$$

Thus, the overall leakage is upper bounded by

$$\Gamma = \frac{\alpha_f}{N} + \frac{x(\sum_{\forall P_k\in C}|P_k| - A\cdot\alpha_f)}{A\cdot L\cdot N} \tag{11}$$

where C is the union of the two sets of data partitions inferred by the two types of inference attacks with background knowledge $\{\alpha_f, \alpha_s\}$. This completes the proof. □

We also demonstrate the defense performance (on bounded leakage) of our proposed framework with the given inference attacks in Section 7.1.2. To sum up, our framework can ensure any bounded leakage against any given set of inference attacks, whereas the existing multi-view approach [13] cannot strictly bound it.

5.3 Privately Retrieving Analysis Result

In Step (7), the service provider performs the same analysis on all the N data views to derive N analysis results. Then, in Step (8), the data owner can privately retrieve the analysis result of the real data view ($\mathcal{D}_r$) via the oblivious random access memory (ORAM) [36] without letting the service provider know which analysis result has been retrieved.

Proposition 5.1. *The generalized outsourcing framework (with the prefix-aware encoding) ensures 100% accuracy for analyzing the prefix preserving encrypted data.*

Proof. Equation 5 ensures that exactly one real data view with fully prefix preserving encrypted data will be generated out of N data views. The accuracy for analyzing such real data view is 100%. Since the data owner knows the end-to-end data encryption (with two CryptoPAn keys K_0 and K_1 for multiple rounds of prefix preserving encryption), it knows the index for the real data view with its locally generated pseudorandom matrix $\mathbb{R} = [G_1, \ldots, G_N]$.

Thus, the data owner can privately retrieve the analysis result of the real data view, which ensures 100% utility on the fully preserved prefixes (validated in Section 7.3). □

6 Discussion

Worst Case Leakage and Amplification Effect. We bound the worst leakage for all the attacks with Γ, e.g., the maximum leakage resulted from different combinations of background knowledge in different inference attacks. Moreover, amplification effect of different inference attacks are also considered in the experiment. For instance, as fingerprinting based inference attacks have recovered some encrypted data with background knowledge α_s, then the accuracy of frequency and ℓ_p-optimization based inference attacks can be improved. Thus, the leakage bounded by Γ is derived from multiple attacks with the amplification effect.

New Inference Attacks. Our generalized outsourcing defines Γ to bound the leakage from any combinations of the inference attacks. In case of other threat models (e.g., other inference attacks [35], [38], or newly identified attacks [32], [33], [34], the data owner only needs to simulate the inference attacks to estimate the leakage and specify the x which results in the minimum N to bound the leakage.

Communication Overheads. Although the framework generates N (could be hundreds) data views to ensure privacy, the data owner only sends one seed data $\mathcal{D}_s$ *with the same size as the original data*, some pseudorandom vectors (matrix) $\mathbb{R}$ and the CryptoPAn key K_1 to the service provider. Moreover, the data owner privately retrieves the corresponding analysis result (with a small size in general) via ORAM. Thus, the total communication overheads are quite close to that of a regular data outsourcing.

7 Experimental Evaluations

We implemented our outsourcing framework on the CloudLab platform [39] where one server works as the client and another as the service provider. We utilize two different real world datasets in the experiments.

Traveler Check-in Location Data. It includes 6,442,890 check-ins records of 196,591 users on a social network (http://snap.stanford.edu/data/loc-gowalla.html). We integrated the data into 633,743 distinct locations in total. A single data record consists of the user IDs, timestamps, locations (GPS coordinates) and location IDs.

Network Traffic Data. It is collected from DoS attacks (https://www.unb.ca/cic/datasets/dos-dataset.html). We extracted the source/destination IPs, timestamps, packet types, and port numbers from a 4.8GB raw dataset. 104,820 records are attributed to 778 distinct source IPs.

We encode the traveler check-in location data (i.e., GPS locations) into bit strings with prefix-aware encoding. For the network traffic data, IP addresses can also be binarized. Then, CryptoPAn can be applied to preserve prefixes in the encrypted bit strings for both datasets.

7.1 Experiments on the Inference Attacks

We have implemented two common inference attacks on the encrypted data: 1) Frequency and ℓ_p-optimization ($p = 2$) based inference attacks [14]; 2) Fingerprinting based inference attacks [13], [19] and set the background knowledge parameters as α_f and α_s. While attacking two encrypted datasets, leakage [10] out of the original data (Definition 3.3) is adopted as the metric to evaluate the confidence of the attacks.

To model the background knowledge of the adversary in the frequency or ℓ_p-optimization based inference attacks, we setup the corresponding auxiliary dataset (including α_f of the original locations/IP addresses' similar frequencies). In addition, any α_s of the original locations/IP addresses are assumed to be identified by the adversary via fingerprinting. We repeated each attacking experiment for 100 times and average the results as the leakage. The average runtimes of different inference attacks on two datasets are shown in Table 2 (all the attacks can be efficiently performed by the service provider and simulated by the data owner).

TABLE 2
Average Runtime of Attacks (sec)

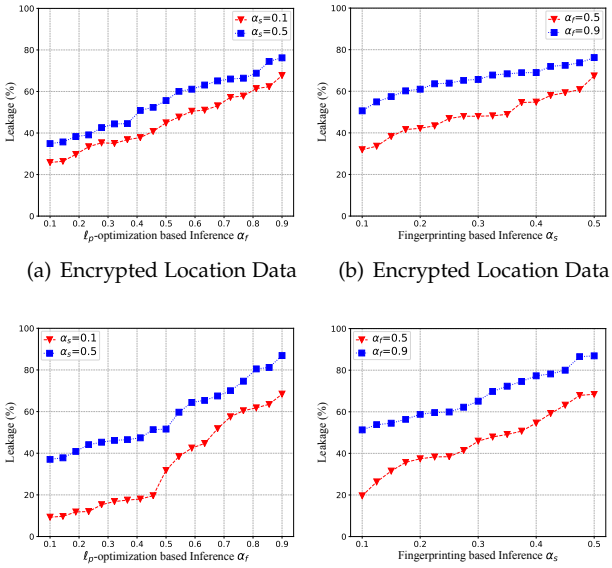
Data Attacks	Frequency	ℓ_2-opt	ℓ_3-opt	Fingerprinting
Location Data	2.64	15.19	18.73	5.24
Network Data	0.12	3.17	4.38	1.23

7.1.1 Attacking CryptoPAn

We first implement the attacks on the two datasets encrypted by CryptoPAn (keys are randomly generated).

1) fixing the fingerprinting-based background knowledge $\alpha_s = 10\%, 50\%$, and measure the leakage via varying the other background knowledge $\alpha_f \in [10\%, 90\%]$ (*from weak to very strong background knowledge*);
2) fixing the background knowledge for frequency and ℓ_p-optimization based inference attacks $\alpha_f = 50\%, 90\%$ and varying the fingerprinting inference $\alpha_s \in [10\%, 50\%]$ (*data injection does not exceed* 50% *in general*).

In Figure 7(a) and 7(c), the leakage grows from 40% to 80% of the original locations/IP addresses as α_f increases from 10% to 90% (changing $\alpha_s = 10\%$ to 50% does not increase the leakage much, compared to α_f). In Figure 7(b) and 7(d), we learn a similar trend for both datasets. These empirical results demonstrate that encrypted locations/IP addresses (by CryptoPAn) are very vulnerable to both ℓ_p-optimization and fingerprinting based inference attacks.



(a) Encrypted Location Data (b) Encrypted Location Data

(c) Encrypted Network Data (d) Encrypted Network Data

Fig. 7. Inference Attacks on Data Encrypted by CryptoPAn

7.1.2 Attacking Multi-view Outsourcing

Intuitively, the more N data views are generated, the less the leakage will be derived from the inference attacks since the adversary cannot distinguish them.

To validate this, we consider the worst case scenario. Given the indistinguishability bound ϵ, the real data view is ϵ-distinguishable with other generated $N-1$ fake data views. We also assume that the adversary attacks all the data views. Denote the probability of identifying any data view as the real data view as Pr_r and the leakage as γ_r.

Also, denote the probability of identifying any data view as a fake data view as Pr_i, $i \in [1, N], i \neq r$ and the leakage as γ_i (γ_r might be larger while γ_i might be smaller since the data is fake). Then, the leakage in the worst case can be obtained as $\gamma_{total} = \sum_{i=1, i\neq r}^{N} \gamma_i * Pr_i + \gamma_r * Pr_r$, where $Pr_i = \frac{1}{N-1+e^{\epsilon}}$, $Pr_r = \frac{e^{\epsilon}}{N-1+e^{\epsilon}}$ in the worst case (since $\forall i \in [1, N], i \neq r, \frac{Pr_r}{Pr_i} \leq e^{\epsilon}$).

We now examine the bounded leakage of multi-view outsourcing against the same inference attacks. Figure 8 presents the required minimum number of data views N on the encrypted location and network traffic data, respectively. Specifically, if the leakage bound Γ increases (from 0.1% to 5%), the required minimum number N declines from ~ 300 to ~ 50 (against strong attackers $\alpha_f = 50\%$ and $\alpha_s = 50\%$), and declines from ~ 50 to ~ 5 (against weak attackers $\alpha_f = 10\%$ and $\alpha_s = 10\%$). While increasing the background knowledge α_f from 10% to 90%, the required minimum N increases for all the leakage bound and α_s (Figure 8(c)). Similarly, while increasing the background knowledge of fingerprinting from 10% to 50%, the required minimum N also increases for all the leakage bound and α_f in the multi-view outsourcing (see Figure 8(e)). Figure 8 (b,c,d) shows a similar trend on network traffic data. Table 3 shows the optimal x for different leakage bound $\Gamma \in [0.1\%, 5\%]$ on the location data, and different background knowledge of two types of inference attacks (α_f, α_s). Most x values are greater than 20 (out of 46). Such long prefixes in the optimal case (minimum N) would generate more partitions.

TABLE 3
Optimal x for Encrypting Locations

Γ (%) / (α_f, α_s)	0.1	0.5	1	1.5	2	2.5	3	3.5	4	5
(0.1, 0.1)	30	25	27	29	20	26	27	20	28	23
(0.1, 0.5)	22	27	29	26	20	29	30	22	25	30
(0.5, 0.1)	24	26	26	23	26	29	21	29	26	29
(0.5, 0.5)	26	27	26	28	26	24	25	26	23	24

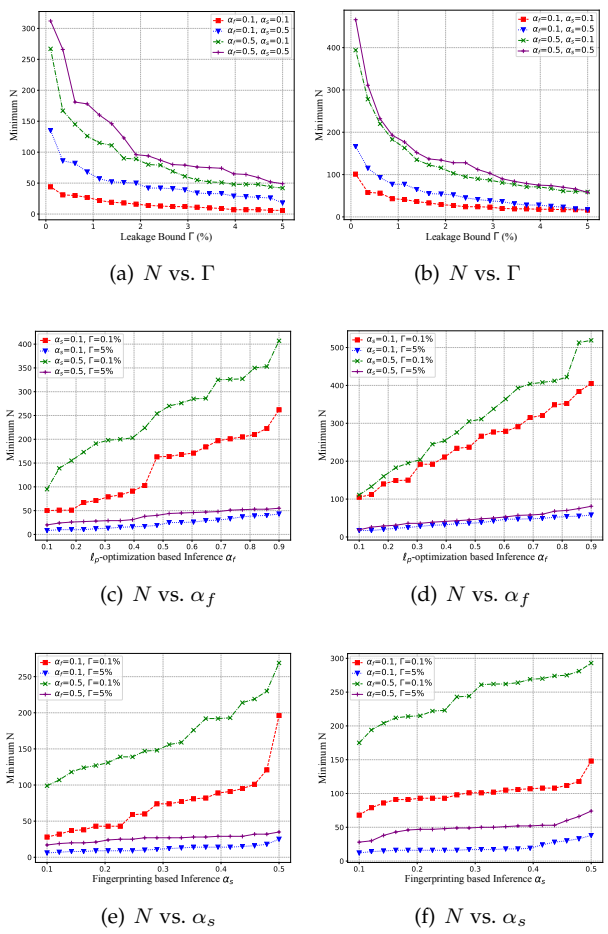
7.2 Indistinguishability

We also demonstrate the indistinguishablity bound ϵ w.r.t. different α_f, α_s and leakage bound Γ. As illustrated in Figure 9 (a,c) and (b,d), ϵ increases as α_s or α_f grows. This indicates that a stronger attacker would be more likely to identify the real data view. Moreover, ϵ is relatively small even if the adversary holds a strong background knowledge. For instance, in case of $\alpha_f = 90\%, \alpha_s = 50\%$, ϵ only equals 1.47 (which is also proven to be bounded in Theorem 5.1). Note that the leakage bound has no significant effect on ϵ since the indistinguishability among the data views is mainly determined by the background knowledge.

7.3 Utility of the Outsourced Data

Furthermore, we evaluate the utility for the outsourced location data using the Periodic Mobility Model (PMM) [21], which can be used to predict the mobility of the users in one week by analyzing their historical check-in data.

First, in Figure 10(a), we plot the location distribution of 100 blocks (each block includes multiple locations) at different times in the original data, real data view and fake data view. We observe that the distributions between the original data and the real data view are identical. Such

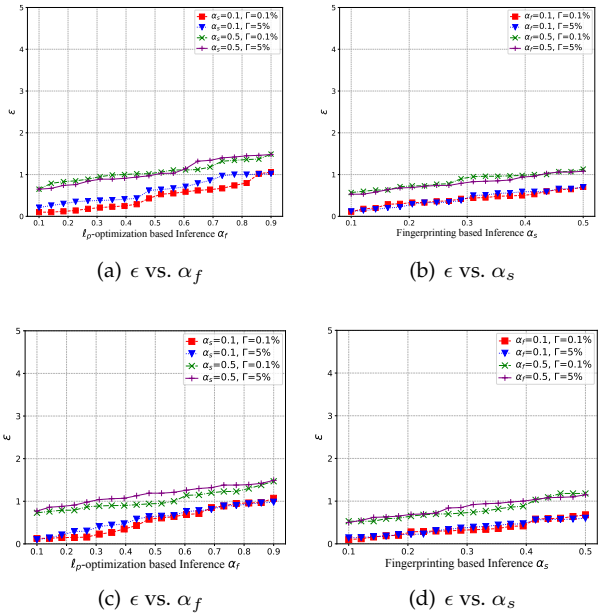


(a) N vs. Γ (b) N vs. Γ

(c) N vs. α_f (d) N vs. α_f

(e) N vs. α_s (f) N vs. α_s

Fig. 8. Minimum N on Location Data (a,c,e), Network Data (b,d,f)



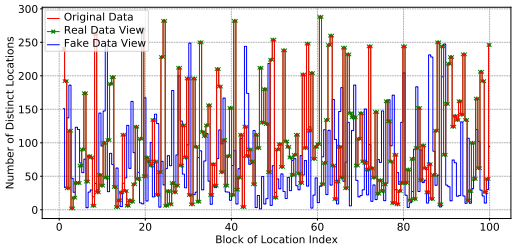
(a) ϵ vs. α_f (b) ϵ vs. α_s

(c) ϵ vs. α_f (d) ϵ vs. α_s

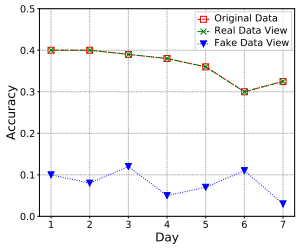
Fig. 9. Indistinguishability on Location Data (a,b) and Network Data (c,d)

100% accuracy is ensured by the prefix preserving property in the outsourced data. Second, we evaluate the accuracy and relative error distance for the real data view and one

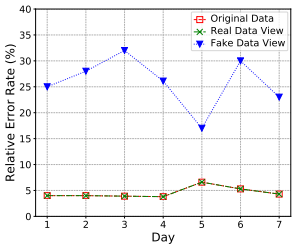
randomly selected fake data view. In Figure 10(b), the average accuracy of the real data views is exactly the same as the original dataset, both of which have a better accuracy than the fake data view. This also matches the fact that the outsourced data can fully preserve the prefixes of the real data without changing other attributes. Figure 10(c) demonstrates the results for relative error of distance, which also validate such excellent utility.



(a) Location Distribution



(b) Accuracy vs. Days



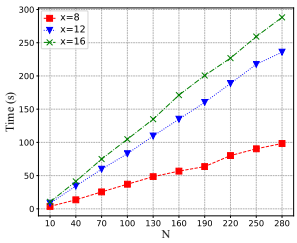
(c) Relative Error Rate vs. Days

Fig. 10. Utility Evaluation on Location Data

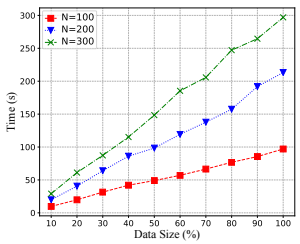
7.4 System Performance

Finally, we also evaluate the computational and communication overheads for outsourcing different datasets. In Figure 11(a), as the number of data views N grows, the runtime increases almost linearly for fixing different x as the prefix length to generate data partitions. While enlarging the prefix length x, the runtime also increases since the number of partitions also increases. Then, the overall computational costs become higher (with more CryptoPAn execution in more data partitions). Figure 11(b) shows the experimental results of runtime versus different data sizes with fixed prefix length $x = 16$ (which is also a linear increase). We leverage the Path-ORAM [40], [41] to implement private result retrieval. The communication bandwidth is around 0.93MB and runtime is only 289ms for outsourcing the location data on average. Such experimental results are reasonable since the data owner only retrieves the analysis result corresponding to the real data view rather than the entire dataset. This is also confirmed in [41].

Finally, bounded by the same information leakage Γ, the proposed generalized framework requires less number of data views compared to [13], and thus reduces the computational overheads at the service provider end (on analyzing all the generated data views). Thus, we also validate this using two real datasets. Specifically, given the prefix length, we apply both our generalized framework and multi-view framework for generating the data views to ensure the same bounded information leakage Γ. For



(a) Time vs. N



(b) Time vs. Data Size ($x = 16$)

Fig. 11. Running Time on Location Data

the multi-view framework, we fix the length of prefix to be 16/23 for the network traffic and location data, respectively. Table 4 summarizes the running time of conducting the analysis on all the generated data views by both approaches with different leakage bound Γ. We can observe that our generalized framework needs less time to analyze all the data views on both two datasets.

TABLE 4
Running Time (sec) vs. Information Leakage

Dataset	Scheme \ Leakage	1%	3%	5%	7%	10%
Network	Multi-view [13]	11.6	8.3	6.7	4.5	3.2
	Ours	8.2	6.5	5.8	3.7	2.8
Location	Multi-view [13]	16.2	13.3	11.0	7.5	5.4
	Ours	11.3	10.4	8.3	5.9	3.6

8 Related Work

Securely Outsourcing Analysis. Securely outsourcing data analysis to third-party service providers has recently grown rapidly, especially with the increasing popularity of cloud technology [42], [43]. For this purpose, provably secure outsourcing has attracted significant attention during past decade. For instance, Sion et al. [44] define the requirements to build a secure outsourcing mechanism. Zhou et al. [45] propose a secure key management scheme which ensures that the source of the data can be securely accessed by different parties under different requirements. Alternatively, oblivious random access memory (ORAM) [46] aims to hide the access patterns of the users, which has been well developed on different topics [40], [47], [48], [49]. In addition, Franz et al. [50] propose a method which can make the data owner delegate rights to new clients for accessing to the outsourced data via a curious server based on ORAM. Stefanov et al. [40] propose a simple ORAM protocol with a small amount of client storage, which is formally proven to require small bandwidth and overheads.

Property Preserving Encryption Schemes. Broadly, various encryption schemes have been proposed to protect the data in different security levels, including fully homomorphic encryption (FHE) [7], [51], functional encryption [52], [53], searchable symmetric encryption [54], [55] and oblivious RAM (ORAM) [36], [41]. Moreover, there are a number of property preserving encryption schemes based on the CryptDB [56], such as order preserving encryption [11], [12]

and deterministic encryption [9]. CryptoPAn [10] was proposed by Xu et al. to ensure the prefix preserving property on IP addresses from the cryptographic view. Kerschbaum [57] proposes a new order preserving encryption scheme which can hide the frequency pattern of plaintexts via randomizing the ciphertexts to mitigate frequency analysis. Wang et al. [58] design a more efficient oblivious data structure which achieves a high efficiency.

Inference Attacks. Brekne et al. [20] presents the attacks via frequency analysis to compromise IP addresses under two prefix preserving anonymization schemes. There are several works which focus on the practical attacks to the encrypted data [14], [15], [59], [60]. Islam et al. [60] introduce the first inference attack which leverages the leakage of access pattern and auxiliary information to get more information about the remaining queries. Naveed et al. [14] present a series of inference attacks on the property preserving encrypted database and implement the attacks on the medical databases to show the effectiveness of the attacks. Recently, Kellaris et al. [15] develop a generic reconstruction attacks on the range queries in the outsourced databases where the access patterns and communication volume are leaked.

9 Conclusion

In this paper, we have proposed a general-purpose prefix preserving encryption scheme that generalizes the previous encryption scheme for only encrypting IP addresses to many other datasets (e.g., geo-locations, market basket datasets, and timestamps) with appropriate prefix-aware encoding. To address the privacy concerns in the encrypted data such that the utilities of prefix preservation can be fully realized, we have proposed a generalized multi-view outsourcing framework that generates multiple *indistinguishable* data views in which one data view fully preserves the utility (prefixes) and its data analysis result can also be obliviously retrieved. We have also empirically evaluated the performance of different inference attacks on two different real datasets encrypted using CryptoPAn and our multi-view outsourcing. The experimental results have demonstrated that our proposed framework preserves both privacy (with bounded leakage and indistinguishability of data views) and utility (with fully preserved prefixes).

Acknowledgments

This work is partially supported by the National Science Foundation (NSF) under Grants No. CNS-1745894, CNS-2046335, and CNS-1564034, the NSERC/Ericsson IRC in SDN/NFV Security, and the National Institutes of Health (NIH) under awards R01GM118574 and R35GM134927. The work has also been supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme. The authors would like to thank the anonymous reviewers for their constructive comments.

References

[1] C. Lu, C. Hsieh, C. Chang and C. Yang, "An improvement to data service in cloud computing with content sensitive transaction analysis and adaptation," in *COMPSAC-W*, 2013, pp. 463–468.

[2] N. Perlroth, "All 3 billion yahoo accounts were affected by 2013 attack,", 2017.

[3] "https://gdpr-info.eu/art-28-gdpr/," 2019.

[4] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[5] P. Karn, W. A. Simpson, and P. Metzger, "The esp triple des transform," 1995.

[6] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.

[8] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *EUROCRYPT*, 2010, pp. 24–43.

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

[10] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," in *ICNP*, 2002, pp. 280–289.

[11] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-preserving symmetric encryption," in *EUROCRYPT*, 2009, pp. 224–241.

[12] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *SIGMOD*, 2004, pp. 563–574.

[13] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi, "Preserving both privacy and utility in network trace anonymization," in *CCS*, 2018, pp. 459–474.

[14] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *CCS*, 2015, pp. 644–655.

[15] G. Kellaris, G. Kollios, K. Nissim, and A. O'neill, "Generic attacks on secure outsourced databases," in *CCS*, 2016, pp. 1329–1340.

[16] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, "The role of network trace anonymization under attack," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 5–11, 2010.

[17] J. King, K. Lakkaraju, and A. Slagell, "A taxonomy and adversarial model for attacks against network log anonymization," in *SAC*, 2009, pp. 1286–1293.

[18] I. A. Al-Kadit, "Origins of cryptology: The arab contributions," *Cryptologia*, vol. 16, no. 2, pp. 97–126, 1992.

[19] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter, "Browser fingerprinting from coarse traffic summaries: Techniques and implications," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, U. Flegel and D. Bruschi, Eds, 2009, pp. 157–175.

[20] T. Brekne, A. Årnes, and A. Øslebø, "Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies," in *PETS*, 2005, pp. 179–196.

[21] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *KDD*, 2011, pp. 1082–1090.

[22] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *CCS*, 2015, pp. 492–503.

[23] Y. He and J. F. Naughton, "Anonymization of set-valued data via top-down, local generalization," *PVLDB*, vol. 2, no. 1, pp. 934–945, 2009.

[24] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *PAKDD*, 1997, pp. 21–34.

[25] C. Chatfield, *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2016.

[26] "https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system," 2019.

[27] J. Pei, J. Han, and L. V. Lakshmanan, "Mining frequent itemsets with convertible constraints," in *ICDE*, 2001, pp. 433–442.

[28] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[29] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Publishing search logs - A comparative study of privacy guarantees," *IEEE TKDE*, vol. 24, no. 3, pp. 520–532, 2012.

[30] A. Rheinländer, M. Knobloch, N. Hochmuth, and U. Leser, "Prefix tree indexing for similarity search and similarity joins on genomic data," in *SSDBM*, 2010, pp. 519–536.

[31] D. Riboni, A. Villani, D. Vitali, C. Bettini, and L. V. Mancini, "Obfuscation of sensitive data in network flows," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2372–2380.

[32] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?" in *CCS*, 2016, pp. 1155–1166.

[33] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *IEEE S&P*, 2017, pp. 655–672.

[34] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," 2019.

[35] J. Vaidya, B. Shafiq, X. Jiang, and L. Ohno-Machado, "Identifying inference attacks against healthcare data repositories," *AMIA Summits on Translational Science Proceedings*, vol. 2013, p. 262, 2013.

[36] X. S. Wang, Y. Huang, T.-H. H. Chan, A. Shelat, and E. Shi, "Scoram: Oblivious ram for secure computation," in *CCS*, 2014, pp. 191–202.

[37] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, 2006, pp. 1–12.

[38] K. Chen, G. Sun, and L. Liu, *Towards Attack-Resilient Geometric Data Perturbation*, pp. 78–89.

[39] "https://docs.cloudlab.com/," 2019.

[40] E. Stefanov, M. V. Dijk, E. Shi, T.-H. H. Chan, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: An extremely simple oblivious ram protocol," *Journal of the ACM*, vol. 65(4), p. 18, 2018.

[41] Z. Chang, D. Xie, and F. Li, "Oblivious ram: A dissection and experimental evaluation," in *PVLDB*, pp. 1113–1124, 2016.

[42] J. Zhang, N. Borisov, and W. Yurcik, "Outsourcing security analysis with anonymized logs," in *Securecomm Workshops*, 2006.

[43] W. Ding, W. Yurcik, and X. Yin, "Outsourcing internet security: Economic analysis of incentives for managed security service providers," in *Internet and Network Economics*, 2005, pp. 947–958.

[44] R. Sion, "Secure data outsourcing," in *VLDB*, 2007, pp. 1431–1432.

[45] M. Zhou, Y. Mu, W. Susilo, J. Yan, and L. Dong, "Privacy enhanced data outsourcing in the cloud," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1367 – 1373, 2012.

[46] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *STOC*, 1987, pp. 182–194.

[47] M. T. Goodrich and M. Mitzenmacher, "Privacy-preserving access of outsourced data via oblivious ram simulation," in *ICALP*. Springer, 2011, pp. 576–587.

[48] E. Stefanov and E. Shi, "Oblivistore: High performance oblivious cloud storage," in *IEEE S&P*, 2013, pp. 253–267.

[49] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious ram," *Journal of Cryptology*, 2007, pp. 22–57.

[50] M. Franz, P. Williams, B. Carbunar, S. Katzenbeisser, A. Peter, R. Sion, and M. Sotakova, "Oblivious outsourced storage with delegation," in *FC*, Springer, 2011, pp. 127–140.

[51] V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, Oct 2011, pp. 5–16.

[52] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, 2004, pp. 506–522.

[53] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *TCC*, 2009, pp. 457–473.

[54] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *CCS*, 2006, pp. 79–88.

[55] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *CRYPTO*, 2013, pp. 353–373.

[56] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *SOSP*, ACM, 2011, pp. 85–100.

[57] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *CCS*, 2015, pp. 656–667.

[58] X. S. Wang, K. Nayak, C. Liu, T.-H. H. Chan, E. Shi, E. Stefanov, and Y. Huang, "Oblivious data structures," in *CCS*, 2014.

[59] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley, "Analyzing privacy in enterprise packet trace anonymization," in *NDSS*, 2008.

[60] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *NDSS*, 2012.

Shangyu Xie received the B.Sc degree from Shanghai Jiao Tong University with dual major in Electrical Engineering and Information Engineering, affiliated with the IEEE Honor Class. He is currently a Ph.D student in the Department of Computer Science at Illinois Institute of Technology, USA. His research focuses on data privacy and security.

Meisam Mohammady is a Research Scientist at Data61 in Commonwealth Scientific and Industrial Research Organisation (CSIRO), with a background in privacy preserving mechanism designs, statistical inferences, and secure computation. He received his Ph.D. degree in information systems engineering from Concordia University, his M.Sc. degree in Electrical Engineering from Ecole Polytechnique Montreal, and his B.Sc. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran.

Han Wang is currently a Ph.D student in the Department of Computer Science at Illinois Institute of Technology. She got her M.Sc degree from Huazhong University of Science and Technology. Her research interests include data privacy and security.

Lingyu Wang received the B.E. degree from Shenyang Aerospace University, China, the M.E. degree from Shanghai Jiao Tong University, and the Ph.D. degree in information technology from George Mason University. He is currently a Professor with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada. His research interests include cloud computing security, network security metrics, and software security and privacy.

Jaideep Vaidya (F'21) received the PhD degree in computer science from Purdue University. He is currently the RBS Dean's Research Professor of computer information systems with Rutgers University. He has published over 140 papers in international conferences and journals. His research interests are in privacy, security, and data management. He is an IEEE Fellow and ACM Distinguished Scientist.

Yuan Hong (SM'18) received his Ph.D. degree in Information Technology from Rutgers University. He is currently an Assistant Professor in the Department of Computer Science at Illinois Institute of Technology. His research interests focus on data privacy, AI security, mechanism design and optimization. He is a recipient of the National Science Foundation (NSF) CAREER Award, and a Senior Member of the IEEE.