A Multi-view Approach to Preserve Both Privacy and Utility in Network Trace Anonymization

MEISAM MOHAMMADY, Concordia University
MOMEN OQAILY, Concordia University
LINGYU WANG, Concordia University
YUAN HONG, Illinois Institute of Technology
HABIB LOUAFI, University of Regina
MAKAN POURZANDI, Ericsson Research Security
MOURAD DEBBABI, Concordia University

As network security monitoring grows more sophisticated, there is an increasing need for outsourcing such tasks to third-party analysts. However, organizations are usually reluctant to share their network traces due to privacy concerns over sensitive information, e.g., network and system configuration, which may potentially be exploited for attacks. In cases where data owners are convinced to share their network traces, the data are typically subjected to certain anonymization techniques, e.g., CryptoPAn, which replaces real IP addresses with prefix-preserving pseudonyms. However, most such techniques either are vulnerable to adversaries with prior knowledge about some network flows in the traces, or require heavy data sanitization or perturbation, which may results in a significant loss of data utility. In this paper, we aim to preserve both privacy and utility through shifting the trade-off from between privacy and utility to between privacy and computational cost. The key idea is for the analysts to generate and analyze multiple anonymized views of the original network traces: those views are designed to be sufficiently indistinguishable even to adversaries armed with prior knowledge, which preserves the privacy; whereas one of the views will yield true analysis results privately retrieved by the data owner, which preserves the utility. We formally analyze the privacy of our solution and experimentally evaluate it using real network traces provided by a major ISP. The experimental results show that our approach can significantly reduce the level of information leakage (e.g., less than 1% of the information leaked by CryptoPAn) with comparable utility.

CCS Concepts: • **Security and privacy → Privacy protections**.

Additional Key Words and Phrases: Network Trace Anonymization, CryptoPAn, Semantic Attacks

1 INTRODUCTION

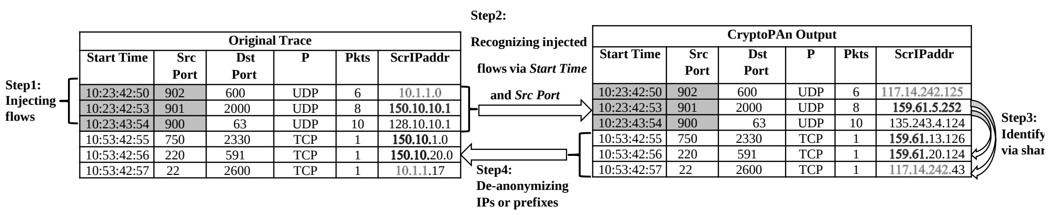
The owners of large-scale network data, ISPs and enterprises usually face a dilemma. As security monitoring and analytics grow more sophisticated, there is an increasing need for those organizations to outsource such tasks together with necessary network data to third-party analysts, e.g., Managed Security Service Providers (MSSPs) [6]. On the other hand, those organizations are typically reluctant to share their network trace data with third parties, mainly due to privacy concerns over sensitive information contained in such data. For example, important network configuration information, such as potential bottlenecks of the network, may be inferred from network traces and subsequently exploited by adversaries to increase the impact of the denial of service attacks [7].

In cases where data owners are convinced to share their network traces, the traces are typically subjected to some anonymization techniques. The anonymization of network traces has attracted significant attention (a more detailed review of related works will be given in Section 8). For instance, *CryptoPAn* replaces real IP addresses inside network flows with prefix preserving pseudonyms,

Authors' addresses: Meisam Mohammady, Concordia University, m_ohamma@encs.concordia.ca; Momen Oqaily, Concordia University, m_oqaily@encs.concordia.ca; Lingyu Wang, Concordia University, wang@ciise.concordia.ca; Yuan Hong, Illinois Institute of Technology, yuan.hong@iit.edu; Habib Louafi, University of Regina, habib.louafi@uregina.ca; Makan Pourzandi, Ericsson Research Security, makan.pourzandi@ericsson.com; Mourad Debbabi, Concordia University, debbabi@encs.concordia.ca.

such that the hierarchical relationships among those addresses will be preserved to facilitate analyses [8]. Specifically, any two IP addresses sharing a prefix in the original trace will also do so in the anonymized trace. However, CryptoPAn is known to be vulnerable to the *fingerprinting* and *injection* attacks [9, 13, 14]. In those attacks, adversaries either already know some network flows in the original traces (by observing the network or from other relevant sources, e.g., DNS and WHOIS databases) [15], or have deliberately injected some forged flows into such traces. By recognizing those known flows in the anonymized traces based on unchanged fields of the flows, namely, fingerprints (e.g., timestamps and protocols), the adversaries can extrapolate their knowledge to recognize other flows based on the shared prefixes [9]. We now demonstrate such an attack.

Example 1.1. *In Figure 1, the upper table shows the original trace, and the lower shows the trace anonymized using CryptoPAn. In this example, without loss of generality, we only focus on source IPs. Inside each table, similar prefixes are highlighted through similar shading.*


Step1: Injecting flows
Step2: Recognizing injected flows via *Start Time* and *Src Port*
Step3: Identify via shar
Step4: De-anonymizing IPs or prefixes

Original Trace					
Start Time	**Src Port**	**Dst Port**	**P**	**Pkts**	**ScrIPaddr**
10:23:42:50	902	600	UDP	6	10.1.1.0
10:23:42:53	901	2000	UDP	8	**150.10.10.1**
10:23:43:54	900	63	UDP	10	128.10.10.1
10:53:42:55	750	2330	TCP	1	**150.10.**1.0
10:53:42:56	220	591	TCP	1	**150.10.**20.0
10:53:42:57	22	2600	TCP	1	10.1.1.17

CryptoPAn Output					
Start Time	**Src Port**	**Dst Port**	**P**	**Pkts**	**ScrIPaddr**
10:23:42:50	902	600	UDP	6	117.14.242.125
10:23:42:53	901	2000	UDP	8	**159.61.5.252**
10:23:43:54	900	63	UDP	10	135.243.4.124
10:53:42:55	750	2330	TCP	1	**159.61.**13.126
10:53:42:56	220	591	TCP	1	**159.61.**20.124
10:53:42:57	22	2600	TCP	1	117.14.242.43

Fig. 1. An example of injection attack

(1) *Step 1: An adversary has injected 3 network flows, as the first 3 records in the original trace.*
(2) *Step 2: The adversary recognizes the 3 injected flows in the anonymized trace (lower table) through unique combinations of the unchanged attributes (*Start Time *and* Src Port*).*
(3) *Step 3: He/she can then extrapolate his/her knowledge from the injected flows to real flows as follows, e.g., since prefix* 159.61 *is shared by the second (injected), fifth (real) and sixth (real) flows, he/she knows all 3 must also share the same prefix in the original trace. Such identified relationships between flows in the two traces will be called* matches *from now on.*
(4) *Step 4: Finally, he/she can infer the prefixes or entire IPs of those anonymized flows in the original traces, as he/she knows the original IPs of his/her injected flows, e.g., the fifth and sixth flows must have prefix* 150.10*, and the IPs of the fourth and last flows must be* 10.1.1.0*.*

A powerful adversary who probes all the subnets of a network via injection/fingerprinting can potentially de-anonymize the entire CryptoPAn output via a more sophisticated frequency analysis attack [9].

Most subsequent solutions either require heavy data sanitization or can only support limited types of analysis. In particular, the (k, j)-obfuscation method first groups together k or more flows with similar fingerprints and then bucketizes (i.e., replacing original IPs with identical IPs) $j < k$ flows inside each group; all records whose fingerprints are not sufficiently similar to $k - 1$ others will be suppressed [7]. Clearly, both the bucketization and suppression may lead to significant loss of data utility. The differentially private analysis method first adds noises to analysis results and then publishes such aggregated results [36]. Although this method may provide privacy guarantee regardless of adversarial knowledge, the perturbation and aggregation prevent its application to analyses that demand accurate or detailed records in the network traces.

In this paper, we aim to preserve both privacy and utility by shifting the trade-off from between privacy and utility, as seen in most existing works, to between privacy and computational cost (which has seen a significant decrease lately, especially with the increasing popularity of cloud technology). The key idea is for the data owner to send enough information to the third party analysts such that they can generate and analyze many different anonymized views of the original network trace. Those anonymized views are designed to be sufficiently indistinguishable (which will be formally defined in Subsection 2.4) even to adversaries armed with prior knowledge and performing the aforementioned attacks, which preserves the privacy. On the other hand, one of the anonymized views will yield true analysis results, which will be privately retrieved by the data owner or other authorized parties, which preserves the utility. More specifically, the major contributions are summarized as follows.

(1) We propose a *multi-view* approach to the prefix-preserving anonymization of network traces. To the best of our knowledge, this is the first known solution that can achieve similar data utility as CryptoPAn does, while being robust against the so-called semantic attacks (e.g., fingerprinting and injection). In addition, we believe that the idea of shifting the trade-off from between privacy and utility to between privacy and computational cost can potentially be adapted to improve other privacy solutions.
(2) In addition to the general multi-view approach, we detail a concrete solution based on iteratively applying CryptoPAn to each partition inside a network trace such that different partitions are anonymized differently in all the views except one (which yields valid analysis results that can be privately retrieved by the data owner). In addition to privacy and utility, we design the solution in such a way that only one *seed* view needs to be sent to the analysts, which avoids additional communication overhead.
(3) We formally analyze the level of privacy guarantee achieved by our method, discuss potential attacks and solutions, and finally experimentally evaluate our solution using real network traces from a major ISP. The experimental results confirm that our solution is robust against semantic attacks with a reasonable computational cost.

A preliminary version of this paper proposing the multi-view idea has appeared in [5]. In this paper, we significantly extend our previous work with focuses on (1) improving the efficiency, (2) extending the evaluation, and (3) broadening the scope of applications, of the multi-view approach as a general technology in other privacy-preserving contexts. Specifically, our major extensions are as follows: i) in addition to the two existing schemes, we propose a new multi-view scheme which requires much lesser computation cost in its deployment while preserving the same level of privacy and utility (detailed in Section 4.3); ii) we conduct an extensive set of new experiments demonstrating the applicability of multi-view by evaluating computation time, privacy and utility for view generation, private retrieval, various network trace analyses and different sizes of dataset (in Section 6); iii) we propose a new extension to our multi-view solution to leverage many other datasets, e.g., location data and genome data, into our multi-view solution, which shows that the multi-view solution represents a very general concept which could potentially be applied to a broader range of datasets than network traces (in Section 5.1); iv) we extend the model of the multi-view to be used in designing schemes guaranteeing differential privacy as a prominent privacy standard (in Section 5.2); and v) finally, we extensively survey the literature over the period of 1998–2019 on network traffic anonymization techniques and implementations (in Section 8).

2 MODELS

In this section, we describe models for the system and adversaries, we briefly review CryptoPAn, we provide a high level overview of our multi-view approach, and finally, we define our privacy property. Essential definitions and notations are summarized in Table 1.

2.1 The System and Adversary Model

Denote by $\mathcal{L}$ a *network trace* comprised of a set of *flows* (or records) r_i. Each flow includes a confidential multi-value attribute $A^{\text{IP}} = \{\text{IP}_{src}, \text{IP}_{dst}\}$, and the set of other attributes $A = \{A_i\}$ is called the *Fingerprint Quasi Identifier (fp-QI)* [7]. Suppose the data owner would like the analyst to perform an analysis on $\mathcal{L}$ to produce a report Γ. To ensure privacy, instead of sending $\mathcal{L}$, an anonymization function $\mathcal{T}$ is applied to obtain an anonymized version $\mathcal{L}^*$. Thus, our main objective is to find the anonymization function $\mathcal{T}$ to preserve both the *privacy*, which means the analyst cannot obtain $\mathcal{T}$ or $\mathcal{L}$ from $\mathcal{L}^*$, and *utility*, which means $\mathcal{T}$ must be prefix-preserving.

In this context, we make following assumptions (similar to those found in most existing works [8, 9, 13, 14]). i) The adversary is a honest-but-curious analyst (in the sense that he/she will exactly follow the approach) who can observe $\mathcal{L}^*$; ii) The anonymization function $\mathcal{T}$ is publicly known, but the corresponding anonymization key is not known by the adversary; iii) The goal of the adversary is to find all possible matches (as demonstrated in Example 1.1, an IP address may be matched to its anonymized version either through the fp-QI or shared prefixes) between $\mathcal{L}$ and $\mathcal{L}^*$; iv) Suppose $\mathcal{L}$ consists of d groups each of which contains IP addresses with similar prefixes (e.g., those in the same subset), and among these the adversary can successfully inject or fingerprint α ($\leq d$) groups (e.g., the demilitarized zone (DMZ) or other subnets to which the adversary has access). Accordingly, we say that the adversary has $\mathcal{S}_\alpha$ knowledge; v) Finally, we assume the communication between the data owner and the analyst is over a secure channel, and we do not consider integrity or availability issues (e.g., a malicious adversary may potentially alter or delete the analysis report).

2.2 The CryptoPAn Model

To facilitate further discussions, we briefly review the CryptoPAn [8] model, which gives a baseline for prefix-preserving anonymization.

Definition 2.1. ***Prefix-preserving Anonymization*** *[8]: Given two IP addresses $a = a_1a_2....a_{32}$ and $b = b_1b_2....b_{32}$, and a one-to-one function $F(.) : \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$, we say that*

- *a and b share a k-bit prefix ($0 \leq k \leq 32$), if and only if $a_1a_2....a_k = b_1b_2....b_k$, and $a_{k+1} \neq b_{k+1}$.*
- *F is prefix-preserving, if, for any a and b that share a k-bit prefix, $F(a)$ and $F(b)$ also do so.*

Given $a = a_1a_2 \cdots a_{32}$ and $F(a) = a'_1a'_2 \ldots a'_{32}$, the prefix-preserving anonymization function F must necessarily satisfy the canonical form [8], as follows.

$$a'_i = a_i \oplus f_{i-1}(a_1a_2 \cdots a_{i-1}),\ i = 1, 2, \ldots, 32 \tag{1}$$

where f_i is a cryptographic function which, based on a 256/128-bit key K, takes as input a bit-string of length $i-1$ and returns a single bit. Intuitively, the i^{th} bit is anonymized based on K and the preceding $i-1$ bits to satisfy the prefix-preserving property. The cryptographic function f_i can be constructed as $L\Big(R\big(P(a_1a_2 \ldots a_{i-1}), K\big)\Big)$ where L returns the least significant bit, R can be a block cipher such as Rijndael [56], and P is a padding function that expands $a_1, a_2, \ldots, a_{i-1}$ to match the

Table 1. The Notation Table.

Symbol	Definition	Symbol	Definition
$\mathcal{L}$	Original network trace	$\mathcal{L}^*$	Anonymized trace
A^{IP}	IP attributes: source and destination IP	*fp-QI*	Fingerprint quasi identifier
r_i	Record number i	n	Number of records in $\mathcal{L}$
α	Number of IP prefixes known by the attacker	$\mathcal{S}_\alpha$	The set of addresses known by attacker
$\mathcal{S}_0^*$	The set of IP addresses in the seed view	$\mathcal{S}_i^*$	The set of IP addresses in view i
PP	CryptoPAn function	RPP	Reverse of CryptoPAn
P_i	Partition i	m	Number of partitions in $\mathcal{L}$
r	Index of real view	K_0, K_1	Private key and outsourced key

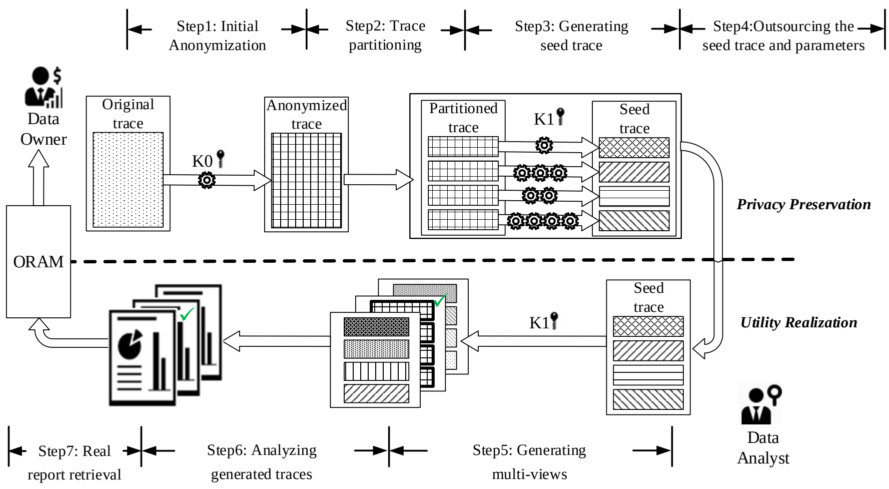

Step1: Initial Anonymization
Step2: Trace partitioning
Step3: Generating seed trace
Step4:Outsourcing the seed trace and parameters
Data Owner
Original trace
K0
Anonymized trace
Partitioned trace
K1
Seed trace
Privacy Preservation
ORAM
Seed trace
K1
Utility Realization
Data Analyst
Step7: Real report retrieval
Step6: Analyzing generated traces
Step5: Generating multi-views

Fig. 2. An overview of the multi-view approach

block size of R [8]. In the following, PP will stand for this CryptoPAn function and its output will be denoted by $a' = PP(a, K)$. The advantage of CryptoPAn is that it is deterministic and allows consistent prefix-preserving anonymization under the same K. However, as mentioned earlier, CryptoPAn is vulnerable to semantic attacks, which will be addressed in next section.

2.3 The Multi-View Approach

We propose a novel *multi-view* approach to the prefix-preserving anonymization of network traces. It preserves both the privacy and utility, while being robust against semantic attacks. The key idea is to hide a prefix-preserving anonymized view, namely, the *real view*, among $N - 1$ other *fake views*, such that an adversary cannot distinguish between those N views, either using his/her prior knowledge or through semantic attacks. Our approach is depicted in Figure 2 and detailed below.

2.3.1 *Privacy Preservation at the Data Owner Side.*

Step 1: The data owner generates two CryptoPAn keys K_0 and K_1, and then obtains an anonymized trace using the anonymization function PP (which will be represented by the gear icon inside this figure) and K_0. This initial anonymization step is designed to prevent the analyst from simulating the process as K_0 will never be given out. Note that this anonymized trace is still vulnerable to semantic attacks and must undergo the remaining steps. Besides, generating this anonymized trace will actually be slightly more complicated due to *migration* (see Section 3.3).

Step 2: The anonymized trace is then partitioned (as detailed in Sections 3.2 and 4).

Step 3: Each partition is anonymized using PP and key K_1, but the anonymization will be repeated, for a different number of times, on different partitions. For example, as the figure shows, the first partition is anonymized only once, whereas the second for three times, etc. The result of this step is called the *seed trace*. The idea is that, as illustrated by the different graphic patterns inside the seed trace, different partitions have been anonymized differently, and hence the seed trace in its entirety is no longer prefix-preserving, even though each partition is still prefix-preserving (note that this is only a simplified demonstration of the *seed trace generator scheme* which will be detailed in Section 4).

Step 4: The seed trace and some additional parameters, including K_1, are outsourced to the analyst.

2.3.2 Utility Realization at the Data Analyst Side.

Step 5: The analyst generates totally N *views* based on the received seed view and supplementary parameters. Our design will ensure one of those generated views, namely, the *real view*, will have all its partitions anonymized in the same way, and thus be prefix-preserving (detailed in Section 4), though the analyst (adversary) cannot tell which exactly is the real view.

Step 6: The analyst performs the analysis on all the N views and generates corresponding reports.

Step 7: The data owner retrieves the analysis report corresponding to the real view following an oblivious random access memory (ORAM) protocol [42], such that the analyst cannot learn which view has been retrieved.

Next, we define the privacy property for the multi-view solution.

2.4 Privacy Property against Adversaries

Under our multi-view approach, an analyst (adversary) will receive N different traces with identical fp-QI attribute values and different A^{IP} attribute values. Therefore, his/her goal now is to identify the real view among all the views, e.g., he/she may attempt to observe his/her injected or fingerprinted flows, or he/she can launch the aforementioned semantic attacks on those views, hoping that the real view might respond differently to those attacks. Therefore, the main objective in designing an effective multi-view solution is to satisfy the *indistinguishability* property which means the real view must be sufficiently indistinguishable from the fake views under semantic attacks. Motivated by the concept of *Differential Privacy* [67], we propose the ϵ-indisinguishablity property as follows.

Definition 2.2. ***ϵ-Indistinguishable*** *Views: A multi-view solution is said to satisfy* ϵ-Indistingui--shability *against an $\mathcal{S}_\alpha$ adversary if and only if (from the adversary's point of view)*

$$\exists\ \epsilon \geq 0,\ \ s.t.\ \forall i \in \{1, 2, \cdots, N\} \Rightarrow$$
$$e^{-\epsilon} \leq \frac{Pr(view\, i\ \ may\ be\ the\ real\ view\,)}{Pr(view\, r\ \ may\ be\ the\ real\ view)} \leq e^{\epsilon} \tag{2}$$

In Defintion 2.2, a smaller ϵ value is more desirable as it means the views are more indistinguishable from the real view to the adversary. For example, an extreme case of $\epsilon = 0$ would mean all the views are equally likely to be the real view to the adversary (from now on, we call these views the *real view candidates*). In practice, the value of ϵ would depend on the specific design of a multi-view solution and also on the adversary's prior knowledge, as will be detailed in the following sections.

Finally, since the multi-view approach requires outsourcing some supplementary parameters, we will also need to analyze the security/privacy of the communication protocol (privacy leakage in the protocol, which complements the privacy analysis in output of the protocol) in semi-honest model under the theory of secure multiparty computation (SMC) [71], [72] (see Section 4.3.2).

3 THE BUILDING BLOCKS

In this section, we introduce the building blocks for our approach, namely, the *iterative and reverse CryptoPAn*, *partition-based* prefix preserving, and *CryptoPAn with IP-collision (migration).*

3.1 Iterative and Reverse CryptoPAn

As mentioned in Section 2.3, the multi-view approach relies on iteratively applying a prefix preserving function PP for generating the seed view. Also, the analyst will invert such an application of PP in order to obtain the real view (among fake views). Therefore, we first need to show how PP can be iteratively and reversely applied.

First, it is straightforward that PP can be iteratively applied, and the result also yields a valid prefix-preserving function. Specifically, denote by $PP^j(a, K)$ $(j > 1)$ the *iterative* application of PP

on IP address a using key K, where j is the number of iterations, called the *index*. For example, for an index of two, we have $PP^2(a, K) = PP(PP(a, K), K)$. It can be easily verified that given any two IP addresses a and b sharing a k-bit prefix, $PP^i(a, K)$ and $PP^i(b, K)$ will always result in two IP addresses that also share a k-bit prefix (i.e., PP^i is prefix-preserving). More generally, the same also holds for applying PP under a sequence of indices and keys (for both IPs), e.g., $PP^i(PP^j(a, K_0), K_1)$ and $PP^i(PP^j(b, K_0), K_1)$ will also share k-bit prefix. Finally, for a set of IP addresses $\mathcal{S}$, iterative PP using a single key K satisfies the following associative property $\forall \mathcal{S}, K,$ and $i, j \in \mathbb{Z}$ (integers): $PP^i\big(PP^j(\mathcal{S}, K), K\big) = PP^j\big(PP^i(\mathcal{S}, K), K\big) = PP^{(i+j)}\big(\mathcal{S}, K\big)$. On the other hand, when a negative number is used as the index, we have a *reverse* iterative CryptPAn function (*RPP* for short), as formally characterized in Theorem 3.1 (see proof in [5]).

THEOREM 3.1. *Given IP addresses $a = a_1a_2 \cdots a_{32}$ and $b = PP(a, K) = b_1b_2 \cdots b_{32}$, the function $RPP(\cdot) : \{0, 1\}^{32} \to \{0, 1\}^{32}$ defined as $RPP(b, K) = c = c_1c_2 \cdots c_{32}$ where $c_i = b_i \oplus f_{i-1}(c_1 \cdots c_{i-1})$ is the inverse of the PP function given in Equation 1, i.e., $c = a$.*

3.2 Partition-based Prefix Preserving

As mentioned in Section 2.3, the core idea of the multi-view approach is to divide the trace into partitions (Step 2), and then anonymize those partitions iteratively, but for different number of iterations (Step 3). In this subsection, we discuss this concept.

Given $\mathcal{S}$ as a set of n IP addresses, we may divide $\mathcal{S}$ into partitions in various ways, e.g., forming equal-sized partitions after sorting $\mathcal{S}$ based on either the IP addresses or corresponding timestamps. The partitioning scheme will have a major impact on the privacy, and we will discuss three such schemes in next section. Once the trace is divided into partitions, we can then apply PP on each partition separately, denoted by $PP(P_i, K)$ for the i^{th} partition. Specifically, given $\mathcal{S}$ divided as a set of m partitions $\{P_1, P_2, \cdots, P_m\}$, we define a *key vector* $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_m \end{bmatrix}$ where each v_i is a positive integer indicating the number of times PP should be applied to P_i, namely, the *key index* of P_i. Given a cryptographic key K, we can then define the *partition-based* prefix preserving anonymization of $\mathcal{S}$ as $PP(\mathcal{S}, V, K) = \big[PP^{v_1}(P_1, K),\ PP^{v_2}(P_2, K), \ldots, PP^{v_m}(P_m, K)\big]$.

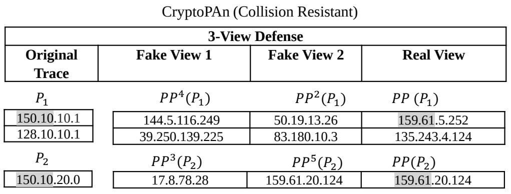
We can easily extend the associative property to this case as the following (which will play an important role in designing our multi-view mechanisms in next section).

$$PP[PP(\mathcal{S}, V_1, K), V_2, K] = PP(\mathcal{S}, (V_1 + V_2), K) \tag{3}$$

3.3 IP Migration: Introducing IP-Collision into CryptoPAn

As mentioned in Section 2.3, once the analyst (adversary) receives the seed view, he/she would generate many indistinguishable views among which only one, the real view, will be prefix preserving across all the partitions, while the other (fake) views do not preserve prefixes across the partitions (Step 5). However, the design would have a potential flaw under a direct application of CryptoPAn. Specifically, since the original CryptoPAn design is collision resistant [8], the fact that similar prefixes are only preserved in the real view across partitions would allow an adversary to easily distinguish the real view from others.

EXAMPLE 3.1. *This vulnerability is shown in Figure 3. The initial trace includes three distinct addresses were split into two partitions P_1 and P_2 partitions. In the figure, the real view is easily distinguishable from the two fake views as the shared prefixes (159.61) between addresses in P_1 and P_2 only show up in the actual vision. This is because, since the partitions in fake views have different rounds of PP applied, and since the original CryptoPAn design is collision resistant [8], the shared prefixes will no longer appear. Hence, the adversary can easily distinguish the real view from others.*



CryptoPAn (Collision Resistant)

3-View Defense			
Original Trace	**Fake View 1**	**Fake View 2**	**Real View**
P_1	$PP^4(P_1)$	$PP^2(P_1)$	$PP\ (P_1)$
150.10.10.1	144.5.116.249	50.19.13.26	159.61.5.252
128.10.10.1	39.250.139.225	83.180.10.3	135.243.4.124
P_2	$PP^3(P_2)$	$PP^5(P_2)$	$PP(P_2)$
150.10.20.0	17.8.78.28	159.61.20.124	159.61.20.124

Fig. 3. An example showing only the real view contains shared prefixes (can be identified by adversaries)

To address this issue, our idea is to create collisions between different prefixes in fake views, such that adversaries cannot tell whether the shared prefixes are due to prefix preserving in the real view, or due to collisions in the fake views. However, due to the collision resistance property of CryptoPAn [8], there is only a negligible probability that different prefixes may become identical even after applying different iterations of PP, as shown in the above example. Therefore, our key idea of *IP migration* is to first replace the prefixes of all the IPs with common values (e.g., zeros), and then fabricate new prefixes for them by applying different iterations of PP. This IP migration process is designed to be prefix-preserving (i.e,. any IPs sharing prefixes in the original trace will still share the new prefixes), and to create collisions in fake views since the addition of key indices during view generation can easily collide. Next, we demonstrate this IP migration in an example.

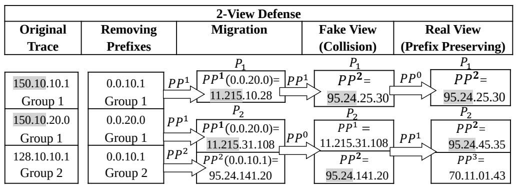


Fig. 4. An example showing, by removing shared prefixes and fabricating them with the same rounds of PP, both fake view and real view may contain fake or real shared prefixes (which makes them indistinguishable)

Example 3.2. *In Figure 4, the primary stage shows the identical original trace as in Example 3.1. In the second stage, we "remove" the prefixes of all IPs and replace them with all zeros (by xoring them with their own prefixes). Next, in the third stage, we fabricate new prefixes by applying different iterations of PP in a prefix preserving manner, e.g., the first two IPs still sharing a common prefix* (11.215) *different from that of the last IP. However, note that whether two IPs share the new prefixes only depends on their key indices now, e.g.,* 1 *for first two IPs and* 2 *for the last IP. This is how we can create collisions in the next stage (the fake view) where the first and last IPs coincidentally share the same prefix* 95.24 *due to their common key indices* 2 *(however, note these are the addition results of different key indices from the migration stage and the view generation stage, respectively). Now, the adversary will not be able to tell which of those views is real based on the existence of shared prefixes. We now formally define the migration function in the following.*

Definition 3.1. ***Migration Function**: Let* $\mathcal{S}$ *be a set of IP addresses consists of* d *groups of IPs* $S_1, S_2, \cdots, S_d$ *with distinct prefixes* $s_1, s_2, \cdots, s_d$ *respectively, and* K *be a random CryptoPAn key. Migration function* $M : \mathcal{S} \times C$*(set of positive integers)* $\longrightarrow \mathcal{S}^*$ *is defined as* $\mathcal{S}^* = M(\mathcal{S}) = \{S_i^* | \forall i \in \{1, 2, \cdots, d\}\}$ *where* $S_i^* = \{PP^{c_i}(s_i \oplus a_j, K), \forall a_j \in S_i\}$*, where* $C = PRNG(d, d) = \{c_1, c_2, \cdots, c_d\}$ *is the set of* d *non-repeating random key indices generated between* $[1, d]$ *using a cryptographically secure pseudo random number generator.*

4 ϵ-INDISTINGUISHABLE MULTI-VIEW MECHANISMS

We first present a multi-view mechanism based on IP partitioning in Section 4.1. We then propose more refined schemes based on distinct IP partitioning with key vector generator in Section 4.2. Finally, we present a third scheme using random IP addresses permutation which can significantly reduce the cost in the data analyst side.

4.1 Scheme I: IP-based Partitioning Approach

To realize the main ideas of multi-view anonymization, as introduced in Section 2.3, we need to design concrete schemes for each step in Figure 2. The key idea of our first scheme is the following. We divide the original trace in such a way that all the IPs sharing prefixes will always be placed in the same partition. This will prevent the attack described in Section 3.3, i.e., identifying the real view by observing shared prefixes across different partitions. As we will detail in Section 4.1.4, this scheme can achieve perfect indistinguishability without the need for IP migration (introduced in Section 3.3), although it has its limitations which will be addressed in our second scheme. Both schemes are depicted in Figure 5 and detailed below. Specifically, our first scheme includes three

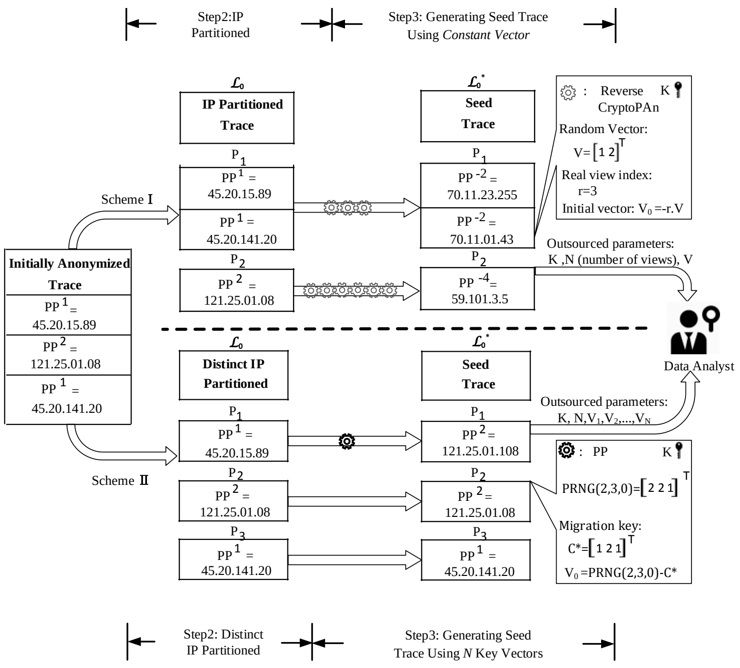


Fig. 5. An example of a trace which undergoes multi-view schemes I, II

main steps: privacy preservation (Section 4.1.1), utility realization (Section 4.1.2), and analysis report extraction (Section 4.1.3).

4.1.1 Privacy Preservation (Data Owner). The data owner performs a set of actions to generate the seed trace $\mathcal{L}_0^*$ together with some parameters to be sent to the analyst for generating different views. These actions are detailed as follows.

- *Applying CryptoPAn using K_0:* First, the data owner generates two independent keys, namely K_0 (key used for initial anonymization, which never leaves the data owner) and K (key used for multi views generation step). The data owner then generates the initially anonymized trace $\mathcal{L}_0$=$PP(\mathcal{L}, K_0)$. This step is designed to prevent the adversary from simulating the scheme, e.g., using a brute-force attack to revert the seed trace back to the original trace in which he/she can recognize some original IPs. The leftmost block in Figure 5 shows an example of the initially anonymized trace.

- *Trace partitioning based on IP-value:* The initially anonymized trace is partitioned based on IP values. Specifically, let $\mathcal{S}$ be the set of IP addresses in $\mathcal{L}_0$ consisting of d groups of IPs $S_1, S_2, \cdots, S_d$ with distinct prefixes $s_1, s_2, \cdots, s_d$, respectively; we divide $\mathcal{L}_0$ to d partitions, each of which is the collection of all records containing one of these groups. For example, the upper part of Figure 5 depicts our scheme I. The set of three IPs are divided into two partitions where P_1 includes both IPs sharing the same prefix, 45.20.15.89 and 45.20.141.20, whereas the last IP 121.25.01.08 goes to P_2 since it does not share a prefix with others.
- *Seed trace creation:* The data owner in this step generates the seed trace using a d-size (recall that d is the number of partitions) random key vector.
  - *Generating a random key vector:* The data owner generates a random vector V of size d using a cryptographically secure pseudo random number generator $PRNG(d\ , d)$ (which generates a set of d non-repeating random numbers between $[1, d]$). This vector V and the key K will later be used by the analyst to generate different views from the seed trace. For example, in Figure 5, for the two partitions, $V = \begin{bmatrix}1 & 2\end{bmatrix}$ is generated. Finally, the data owner chooses the total number of views N to be generated later by the analyst, based on his/her requirements about privacy and computational overhead, since a larger N will mean more computation by both the data owner and analyst but also more privacy (more real view candidates will be generated, and we will further study this via experiments).
  - *Generating a seed trace key vector and a seed trace:* The data owner picks a random number $r \in [1, N]$ and then computes $V_0 = -r \cdot V$ as the key vector of seed trace. Next, the data owner generates the seed trace as $\mathcal{L}_0^* = PP(\mathcal{L}_0, V_0, K)$. This ensures, after the analysts applies exactly r iterations of V on the seed trace, he/she would get $\mathcal{L}_0$ back (while not being aware of this fact since he/she does not know r). For example, in Figure 5, $r = 3$ and $V_0 = \begin{bmatrix}-3 & -6\end{bmatrix}$. We can easily verify that, if the analyst applies the indices in V on the seed trace three times, the outcome will be exactly $\mathcal{L}_0$ (the real view). This can be more formally stated as follows (the r^{th} view $\mathcal{L}_r^*$ is actually the real view). $\mathcal{L}_r^* = PP(\mathcal{L}_0^*, r \cdot V, K)$, using (3) which is to say $PP(\mathcal{L}_0, V_0 + r \cdot V, K)$, using (3), or identically $PP(\mathcal{L}_0, -r \cdot V + r \cdot V, K) = \mathcal{L}_0$.
- Outsourcing: Finally, the data owner outsources $\mathcal{L}_0^*$, V, N and K to the analyst.

4.1.2 Network Trace Analysis (Analyst). The analyst generates the N views requested by the data owner, which is formalized as the following $\mathcal{L}_0^*$, is the seed view, thus $\mathcal{L}_i^* = PP(\mathcal{L}_{i-1}^*, V, K)$, $i \in \{1, \dots, N\}$. Since boundaries of partitions must be recognizable by the analyst to allow him/her to generate the views, we modify the timestamp of the records that are on the boundaries of each partition by changing the most significant digit of the time stamps which is easy to verify and does not affect the analysis as it can be reverted back to its original format by the analyst. Next, the analyst performs the requested analysis on all N views and generates N analysis reports $\Gamma_1, \Gamma_2, \cdots, \Gamma_N$.

4.1.3 Analysis Report Extraction (Data Owner). The data owner is only interested in the analysis report that is related to the real view, which we denote by Γ_r. To minimize the communication overhead, instead of requesting all the analysis reports Γ_i of the generated views, the data owner can fetch only the one that is related to the real view Γ_r. He/she can employ the *oblivious random accesses memory* (ORAM) [42] to do so without revealing the information to the analyst (we will discuss alternatives in Section 8).

4.1.4 Security Analysis. We now analyze the level of indistinguishability provided by the scheme. The statement inside the probability in Definition 2.2 is the adversary's decision on a view, declaring it as fake or a *real view candidate*, using his/her $\mathcal{S}_\alpha$ knowledge. Moreover, we note that generated views differ only in their IP values (fp-QI attributes are similar for all the views). Hence, the adversary's decision can only be based on the published set of IPs in each view through comparing shared prefixes among those IP addresses which he/she already know ($\mathcal{S}_\alpha$). Accordingly, in the following, we define a function to represent all the prefix relations for a set of IPs.

Lemma 4.1. *For two IP addresses a and b, function* $Q : \{0,1\}^{32} \times \{0,1\}^{32} \longrightarrow \mathbb{N}$ *returns the number of bits in the prefix shared between a and b:* $Q(a,b) = 31 - \lfloor log_2^{a \oplus b} \rfloor$, *where* $\lfloor \cdot \rfloor$ *denotes the* floor.

Definition 4.1. *For a multiset of n IP addresses* $\mathcal{S}$, *the* Prefixes Indicator Set *(PIS)* $\mathcal{R}(\mathcal{S})$ *is defined as follows:* $\mathcal{R}(\mathcal{S}) = \{Q(a_i, a_j) | \; \forall a_i, a_j \in \mathcal{S}, i, j \in \{1, 2, \cdots, n\}\}$.

Note that PIS remains unchanged when CryptoPAn is applied on $\mathcal{S}$, i.e., $\mathcal{R}(PP(\mathcal{S}, K)) = \mathcal{R}(\mathcal{S})$. In addition, since the multi-view solution keeps all the other attributes intact, the adversary can identify his/her pre-knowledge in each view and construct prefixes indicator sets out of them. Accordingly, we denote by $\mathcal{R}_{\alpha,i}$ the PIS constructed by the adversary in view i.

Definition 4.2. *Let* $\mathcal{R}_\alpha$ *be the PIS for the adversary's knowledge, and* $\mathcal{R}_{\alpha,i}, i \in \{1, \cdots, N\}$ *be the PIS constructed by the adversary in view i. A multi-view solution then generates* ϵ-indistinguishable *views against an* $\mathcal{S}_\alpha$ *adversary if and only if* $\exists \, \epsilon \geq 0, \; s.t. \; \forall i \in \{1, 2, \cdots, N\} \Rightarrow e^{-\epsilon} \leq \frac{Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha)}{Pr(\mathcal{R}_{\alpha,r} = \mathcal{R}_\alpha)} \leq e^{\epsilon}$.

Lemma 4.2. *The indistinguishability property, defined in Definition 4.2 can be simplified to* $\exists \, \epsilon \geq 0, \; s.t. \; \forall i \in \{1, 2, \cdots, N\} \Rightarrow Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha) \geq e^{-\epsilon}$.

Proof. $Pr(\mathcal{R}_{\alpha,r} = \mathcal{R}_\alpha) = 1$ as view r is the prefix preserving output. Moreover, $\forall \epsilon \geq 0$, we have $e^{\epsilon} \geq 1$. Thus, we only need to show $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ (each generated view i is a real view candidate). □

Theorem 4.3 (Proof in [5]). *Scheme* I *satisfies Definition 4.2 with* $\epsilon = 0$.

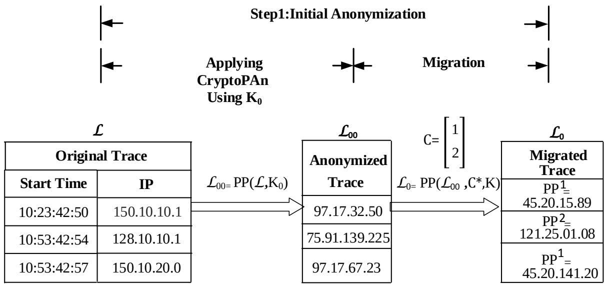
It shows that scheme I produces perfectly indistinguishable views ($\epsilon = 0$). In fact, it is robust against the attack explained in Section 3.3 and thus does not required IP migration, because the partitioning algorithm already prevents addresses with similar prefixes from going into different partitions (the case in Figure 3). However, although adversaries cannot identify the real view, they may choose to live with this fact, and attack each partition inside any (fake or real) view instead, using the same semantic attack as shown in Figure 1. Note that our multi-view approach is only designed to prevent attacks across different partitions, and each partition itself is essentially still the output of CryptoPAn and thus still inherits its weakness.

Fortunately, the multi-view approach gives us more flexibility in designing schemes to further mitigate such a weakness of CryptoPAn. We next present scheme II which sacrifices some indistinguishability (with slightly less real view candidates) to achieve better protected partitions.

4.2 Scheme II: Multi-view Using N Key Vectors

To address the limitation of our first scheme, we propose the next scheme, which is different in terms of the initial anonymization step, IP partitioning, and key vectors for view generation.

4.2.1 Initial Anonymization with Migration. First, to mitigate the attack on each partition, we must relax the requirement that all shared prefixes go into the same partition. However, as soon as we do so, the attack of identifying the real view through prefixes shared across partitions, as demonstrated in Section 3.3, might become possible. Therefore, we modify the first step of the multi-view approach (initial anonymization) to enforce the IP migration technique. Figure 6



Step1:Initial Anonymization
Applying CryptoPAn Using K₀
Migration
$\mathcal{L}$
$\mathcal{L}_{00}$
$C = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
$\mathcal{L}_0$
$\mathcal{L}_{00} = PP(\mathcal{L}, K_0)$
$\mathcal{L}_0 = PP(\mathcal{L}_{00}, C^*, K)$

Original Trace	
Start Time	IP
10:23:42:50	150.10.10.1
10:53:42:54	128.10.10.1
10:53:42:57	150.10.20.0

Anonymized Trace
97.17.32.50
75.91.139.225
97.17.67.23

Migrated Trace
PP^1= 45.20.15.89
PP^2= 121.25.01.08
PP^1= 45.20.141.20

Fig. 6. The updated initial anonymization (Step 1 in Figure 2) for enforcing migration

demonstrates this. The original trace is first anonymized with K_0, and then the anonymized trace goes through the migration process, which replaces the two different prefixes (97.17 and 75.91) with different iterations of PP, as discussed in Section 3.3.

4.2.2 Distinct IP Partitioning and N Key Vectors Generation. For the scheme, we employ a special case of IP partitioning where each partition includes exactly one distinct IP (i.e., the collection of all records containing the same IP). For example, the trace shown in Figure 5 includes three distinct IP addresses 150.10.10.1,128.10.10.1, and 150.10.20.0. Therefore, the trace is divided into three partitions. Next, the data owner will generate the seed view as in the first scheme, although the key V_0 will be generated completely differently, as detailed below.

Let $\mathcal{S}^* = \{S_1^*, S_2^*, \cdots, S_d^*\}$, be the set of IP addresses after the migration step. Suppose $\mathcal{S}^*$ consists of D distinct IP addresses. We denote by C^* the multiset of totally D migration keys for those distinct IPs (in contrast, the number of migration keys in C is equal to the number of distinct prefixes, as discussed in Section 3.3). Also, let $PRNG(d, D, i)$ be the set of D random number generated between $[1, d]$ using a cryptographically secure pseudo random number generator at iteration i^{th}. The data owner will generate $N + 1$ key vector V_i as follows $V_i = PRNG(d, D, i) - PRNG(d, D, i-1), \forall i \neq r \in [1, 2 \cdots, N]$ and $V_0 = PRNG(d, D, 0) - C^*,\ V_r = C^* - PRNG(d, D, r-1)$.

Example 4.1. *In Figure 7, the migration and random vectors are* $C^* = [1\ 1\ 2]$, $PRNG(2, 3, 0) = [1\ 2\ 2]$, $PRNG(2, 3, 1) = [1\ 2\ 1]$, *and* $PRNG(2, 3, 2) = [2\ 2\ 1]$, *respectively. The corresponding key vectors will be* $V_0 = [0\ 1\ 0]$, $V_1 = [0\ 0\ -1]$ *and* $V_2 = [1\ 0\ 0]$ *where only* V_1 *and* V_2 *are outsourced.*

In this scheme, the analyst at each iteration i generates a new set of IP addresses $\mathcal{S}_i^* = \{S_1^i, S_2^i, \cdots, S_d^i\}$ by randomly grouping all the distinct IP addresses into a set of d prefix groups. In doing so, each new vector V_i essentially cancels out the effect of the previous vector V_{i-1}, and thus introduces a new set of IP addresses $\mathcal{S}_i^*$ consisting of d prefix groups. Thus, we can verify that the r^{th} generated view will prefix preserving (the addresses are migrated back to their groups using C^*).

Example 4.2. *Figure 7 shows that, in each iteration, a different set (but with an equal number of elements) of prefix groups will be generated. For example, in the seed view, IP addresses* 150.10.20.0 *and* 128.10.10.1 *are mapped to prefix group* 11.215.

4.2.3 Indistinguishability Analysis. By placing each distinct IP in a partition, our second scheme is not vulnerable to semantic attacks on each partition, since such a partition contains no information about the prefix relationship among different addresses. However, compared with scheme I, as we show in the following, this scheme achieves a weaker level of indistinguishability (higher ϵ). Specifically, to verify the indistinguishability of the scheme, we calculate $Pr(\mathcal{R}_\alpha = \mathcal{R}_{\alpha,i})$ for scheme II in the following. First, the number of all possible outcomes of grouping D IP addresses into d groups with predefined cardinalities is $N_{total} = \frac{D!}{|S_1|!|S_2|!\cdots|S_d|!}$ where $|S_i|$ denotes the cardinality

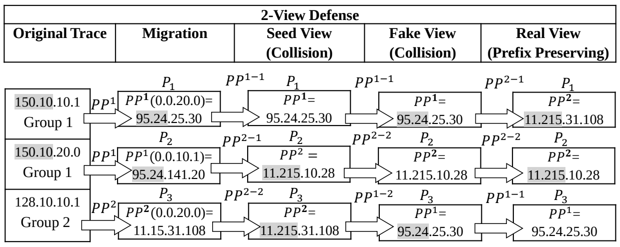


Fig. 7. An example of two views generation under scheme II

of group i. Also the number of all possible outcomes of grouping D IP addresses into d groups while still having $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ is $N_{\text{real view candidates}} = \frac{\alpha!\ (D-\alpha)!\ \sum_{i=1}^{\binom{d}{\alpha}}\left(\Pi_{i=1}^{\alpha}|S_{a_i}|\right)}{|S_1|!|S_2|!\cdots|S_d|!}$ for some $a_i \in \{1, 2, \cdots, d\}$. This equation gives the number of outcomes when a specific set of α IP addresses ($\mathcal{S}_\alpha$) are distributed into α different groups and hence keeping $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ (i.e., the adversary cannot identify collision). Note that term $\sum_{i=1}^{\binom{d}{\alpha}}\left(\Pi_{i=1}^{\alpha}|S_{a_i}|\right)$ is all the combinations of choosing this α groups for the numerator to model all the $(|S_{a_i}|-1)!$ combinations.

Finally, we have $\forall i \le N: \ \ Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha) = \frac{N_{\text{real view candidates}}}{N_{total}}$ which is equal to

$$\mathcal{A} = \frac{\alpha!\ \sum_{i=1}^{\binom{d}{\alpha}}\left(\Pi_{i=1}^{\alpha}|S_{a_i}|\right)}{\Pi_{i=0}^{\alpha-1}(D-i)} \ge e^{-\epsilon} \tag{4}$$

Thus, to ensure the ϵ-indistinguishability, the data owner needs to satisfy the expression in equation 4 which is a relationship between the number of distinct IP addresses, the number of groups, the cardinality of the groups in the trace and the adversary's knowledge.

Theorem 4.4. *ϵ in the indistinguishability of the generated views in scheme* II *is lower-bounded by*

$$\ln\left[\frac{D^\alpha}{d^\alpha}\cdot\Pi_{i=0}^{\alpha-1}\frac{(d-i)}{(D-i)}\right] \tag{5}$$

Proof. Let $b_1, b_2, \cdots, b_n$ be positive real numbers, and for $k = 1, 2, \cdots, n$ define the averages M_k as $M_k = \frac{\sum\limits_{1\le i_1\le i_2\le\cdots\le i_k\le n} b_{i_1}b_{i_1}\cdots b_{i_k}}{\binom{n}{k}}$. By Maclaurin's inequality [55], which is the following chain of inequalities $M_1 \ge \sqrt[2]{M_2} \ge \sqrt[3]{M_3} \ge \cdots \ge \sqrt[n]{M_n}$ where $M_1 = \frac{\sum_{i=1}^{n} b_i}{n}$, we have $\mathcal{A} = \frac{\alpha!\binom{d}{\alpha}M_\alpha}{\Pi_{i=0}^{\alpha-1}(D-i)} \le \frac{\Pi_{i=0}^{\alpha-1}(d-i)(M_1)^\alpha}{\Pi_{i=0}^{\alpha-1}(D-i)}$ and since $M_1 = \frac{\sum_{i=1}^{n}|S_i|}{n} = \frac{D}{d}$, we have $A \le \frac{D^\alpha}{d^\alpha}\cdot\Pi_{i=0}^{\alpha-1}\frac{(d-i)}{(D-i)}$. □

Figure 8(a) shows how the lower-bound in Equation 5 changes with respect to different values of fraction d/D and also the adversary's knowledge. As it is expected, stronger adversaries have more power to weaken the scheme which results in increasing ϵ or increasing the chance of identifying the real view. Moreover, as it is illustrated in the figure, when fraction d/D grows, ϵ tends to converge to very small values. Hence, to decrease ϵ, the data owner may increase $d/D \in [0, 1]$ by grouping addresses based on a bigger number of bits in their prefixes, e.g., a certain combination of 3 octets would be considered as a prefix instead of one or two. Another solution could be aggregating the original trace with some other traces for which the cardinalities of each prefix group are small. We study this effect in our experiments in Section 6, especially in Figures 15 and 16.

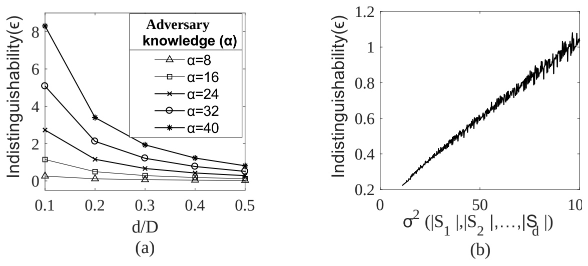


Fig. 8. (a) The trend of bound 5 for ϵ when adversary's knowledge varies. (b) The trend of exact value of ϵ in equation 4 for $\alpha = 16$, $d/D = 0.1$ and when variance of cardinalities varies

Finally, Figure 8(b) shows how variance of the cardinalities affects the indistinguishability for a set of fixed parameters d, D, α. In fact, when the cardinalities of the prefix groups are close (small σ), $\mathcal{A}$ grows to meet the lower-bound in Theorem 4.4. Hence, from the data owner perspective, a trace with a lower variance of cardinalities and a bigger fraction d/D has a better chance of misleading adversaries who wants to identify the real view.

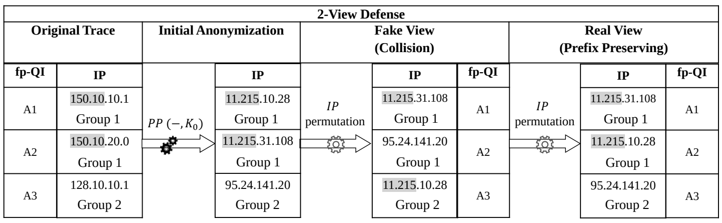
4.2.4 Security of the communication protocol. We now analyze the security/privacy of our protocol in semi-honest model under the secure multiparty computation (SMC) [71], [72] theory.

Lemma 4.5 (Proof in [5]). *Scheme* II *only reveals the CryptoPAn Key K and the seed trace* $\mathcal{L}_0^*$.

Note that outsourcing the $\mathcal{L}_0^*$ and the outsourced key are trivial leakage. The outsourced key can be considered as a public key and the leakage of $\mathcal{L}_0^*$ which was studied earlier. Finally, we study the *setup leakage* and show that the adversary cannot exploit outsourced parameters to increase ϵ (i.e., decrease the number of real view candidates) by building his/her own key vector.

Lemma 4.6 (Proof in [5]). *For an* $\mathcal{S}_\alpha$ *adversary, who wants to obtain the least number of real view candidates, if condition* $(2d-2)^D > N$ *holds, the best approach is to follow scheme* II, *(scheme* II *returns the least number of real view candidates).*

4.3 Scheme III: IP Attribute Permutation

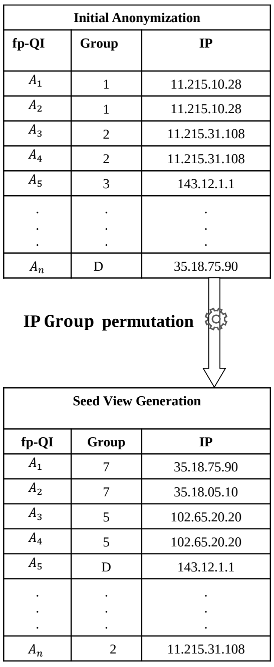


2-View Defense									
Original Trace		Initial Anonymization		Fake View (Collision)			Real View (Prefix Preserving)		
fp-QI	IP		IP		IP	fp-QI		IP	fp-QI
A1	150.10.10.1 Group 1	$PP(-, K_0)$	11.215.10.28 Group 1	*IP* permutation	11.215.31.108 Group 1	A1	*IP* permutation	11.215.31.108 Group 1	A1
A2	150.10.20.0 Group 1		11.215.31.108 Group 1		95.24.141.20 Group 1	A2		11.215.10.28 Group 1	A2
A3	128.10.10.1 Group 2		95.24.141.20 Group 2		11.215.10.28 Group 2	A3		95.24.141.20 Group 2	A3

Fig. 9. An example of two views generation under scheme III

As we have seen, scheme II requires applying CryptoPAn over the (sheer size) network trace for a large number of times. Therefore, we propose a third scheme which achieves the same indistinguishability bound as scheme II does among the generated views, yet requires much less computation cost in its deployment. The key idea is to fulfill the IP distribution task required in each view generation by randomly permuting/assigning IP addresses to other attributes. Specifically, instead of using associative property of CryptoPAn to randomly diffuse IP addresses in the fake views and reconstruct the real view, in scheme III, we simply permute IP addresses and randomly assign to them the fp-QI attributes to achieve the diffusion and the reconstruction. The following example demonstrates different operations to be performed on a sample network trace under scheme III.

Example 1. Figure 9 is an example of generating two views under scheme III. The original trace is illustrated in Figure 9 with three records and each record is a pair (fp-QI Activity, IP). As illustrated in Figure 9, the CryptoPAn is applied only once in the initialization step on the data owner's side. The view generation is then performed through shuffling (or permuting) the IP addresses and assigning them to one of the fp-QI activities. For example, in Figure 9, the fake view is generated by switching the IP addresses of group 1 with those from group 2, and the real view is reconstructed by permuting IP addresses back to their original positions. Clearly, the IP permutation algorithm is the main building block of this scheme. To satisfy the same level of indistinguishability as scheme II, the permutation algorithm follows the key vector generation method of scheme III. The following steps are the main operations that the data owner needs to perform under scheme III.



Initial Anonymization

fp-QI	Group	IP
A_1	1	11.215.10.28
A_2	1	11.215.10.28
A_3	2	11.215.31.108
A_4	2	11.215.31.108
A_5	3	143.12.1.1
⋮	⋮	⋮
A_n	D	35.18.75.90

Seed View Generation

fp-QI	Group	IP
A_1	7	35.18.75.90
A_2	7	35.18.05.10
A_3	5	102.65.20.20
A_4	5	102.65.20.20
A_5	D	143.12.1.1
⋮	⋮	⋮
A_n	2	11.215.31.108

Fig. 10. Seed view generation in scheme III for a trace of n records and D distinct IPs

```
Input:
  $\mathcal{L}, \mathcal{S}, K_0, r, D$, fp-QI: same as in scheme II
  $V_0, V_1, \cdots, V_N$: Vectors of size $D$ defined by data owner same as in scheme II
  Permute $(\mathcal{S}, V)$: Permuting Distinct IP Addresses
  Assign $(fp-QI, \mathcal{S})$: Assigning IPs to Corresponding fp-QI
Output:
  $\mathcal{L}^*$: Anonymized trace to be outsourced
Function: anonymize $(\mathcal{L}, D, K_0, r, V_0)$
begin
1       $\mathcal{L} := PP(\mathcal{L}, K_0)$
2       $\mathcal{S}^* :=$ Permute $(\mathcal{S}, V_0)$
3       $\mathcal{L}^* :=$ Assign $(fp-QI, \mathcal{S}^*)$
4     end
5    return $\mathcal{L}^*, V_1, V_2, \cdots, V_N$
end
```

Algorithm 1: Data owner: network trace anomymization (scheme III)

- *Initial anonymization*: The data owner generates the initially anonymized IP addresses using the set of original IPs $\mathcal{S}$ and a confidential key K_0, i.e., $\mathcal{S}^* = PP(\mathcal{S}, K_0)$.
- *N+1 permutation vectors generation*: The data owner then generates $N + 1$ permutation vectors to be used by the data analyst to perform permutation of IP addresses. Specifically, $V_i = PRNG(D, D, i)$, $i \in \{0, 1, \cdots, N\}$, is the set of D distinct random numbers generated between $[1, D]$ using a cryptographically secure pseudo random number generator at iteration i.
- *Seed trace creation*: After generating vector V_0, the data owner performs permutation on the initially anonymized trace for each distinct IP according to the group indices in random vector V_0. Figure 10 illustrates the seed view generation under this scheme.

Algorithm 1 summarizes these actions. Finally, the operations of the data analyst in generating different views and analyze them are identical to the one under scheme II.

4.3.1 Indistinguishability Analysis. By placing each distinct IP in a partition through permutation, as we show in the following, our third scheme is not vulnerable to semantic attacks on each partition similar to scheme II. Specifically, to verify the indistinguishability of the scheme, we calculate $Pr(\mathcal{R}_\alpha = \mathcal{R}_{\alpha,i})$ for scheme II in the following. First, the number of all possible outcomes of permuting D is $N_{total} = \frac{D!}{|S_1|!|S_2|!\cdots|S_d|!}$, where $|S_i|$ denotes the cardinality of group i. Also the number of all possible outcomes of permuting D IP addresses while still having $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ is $N_{\text{real view candidates}} = \frac{\alpha!\ (D-\alpha)!\ \sum_{i=1}^{\binom{d}{\alpha}} \left(\Pi_{i=1}^{\alpha}|S_{a_i}|\right)}{|S_1|!|S_2|!\cdots|S_d|!}$ for some $a_i \in \{1, 2, \cdots, d\}$ (refer to scheme II for details). Finally, we have $\forall i \leq N: \quad Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha) = \frac{N_{\text{real view candidates}}}{N_{total}} = \frac{\alpha!\ \sum_{i=1}^{\binom{d}{\alpha}} \left(\Pi_{i=1}^{\alpha}|S_{a_i}|\right)}{\Pi_{i=0}^{\alpha-1}(D-i)} \geq e^{-\epsilon}$. Thus, the ϵ-indistinguishability of scheme III is identical to the one in scheme II.

Corollary 4.7. *The indistinguishability parameter ϵ of the generated views in scheme* II *is lower-bounded by* $\ln\left[\frac{D^\alpha}{d^\alpha} \cdot \Pi_{i=0}^{\alpha-1}\frac{(d-i)}{(D-i)}\right]$.

Similar to scheme II, the more the fraction d/D is the better the indistinguishability of the scheme will be. Hence, to decrease ϵ, the data owner may increase $d/D \in [0, 1]$ by grouping addresses based on a bigger number of bits in their prefixes, e.g., a certain combination of 3 octets would be considered as a prefix instead of one or two.

4.3.2 Security of the communication protocol. We now analyze the security/privacy of our communication protocol in semi-honest model under the theory of secure multiparty computation (SMC) [71], [72].

Lemma 4.8. *Scheme* III *only reveals the seed trace $\mathcal{L}_0^*$ in semi-honest model.*

Proof. Since our communication protocol only involves one-round communication between two parties, we only need to examine the data analyst's view (messages received from the protocol), which includes (1) $\mathcal{L}_0^*$: the seed trace, and (2) $V_1, V_2, \cdots, V_N$: the key vectors. Similar to scheme II, each of $V_1, V_2, \ldots, V_N$ can be simulated by generating a single random number from a uniform random distribution (which proves that they are not leakage in the protocol). Specifically, all the entries in $V_1, V_2, \cdots, V_N$ are in $[1, D]$. Thus, all the random entrees in $V_1, V_2, \cdots, V_N$ can be simulated in polynomial time using a simulator (based on the knowledge data analyst already knew, i.e., his/her input and/or output of the protocol). □

Finally, similar to scheme II, we study the *setup leakage.*

Lemma 4.9. *For an $\mathcal{S}_\alpha$ adversary, who wants to obtain the least number of real view candidates, if condition $(2d-2)^D > N$ holds, the best approach is to follow scheme* III, *(scheme* III *returns the least number of real view candidates).*

5 APPLICATION OF THE MULTI-VIEW APPROACH IN OTHER DOMAINS

In this section, we discuss some important applications of the multi-view approach in other privacy-preserving contexts. In particular, we extend (1) our framework to leverage many other datasets, e.g., location data and genome data, and (2) the multi-view model to be used in designing non-interactive schemes guaranteeing differential privacy with boosted utility.

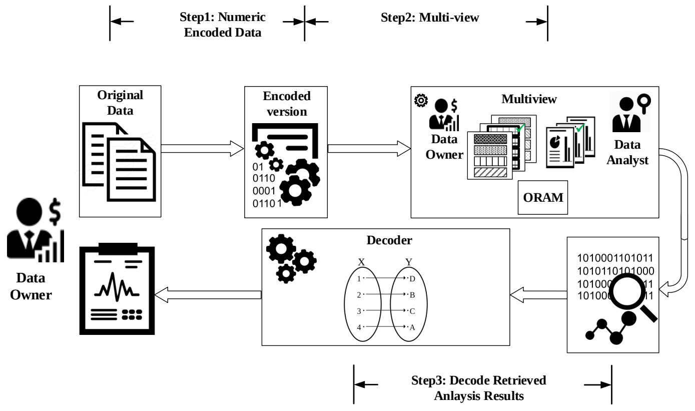


Fig. 11. Multi-view can be applied to many other data types using data encoder and decoder patches

5.1 Multi-view for Outsourcing Other Types of Datasets

Multi-view represents a very general concept which could potentially be applied in a broader range of contexts. In fact, Property preserving encryption (PPE) techniques have significantly advanced the utility of encrypted data in various data outsourcing settings (e.g., the cloud) [33, 34]. However, while preserving certain properties (e.g., order and prefixes) in the encrypted data, PPEs are typically limited to specific data types (e.g., IP addresses) [8], applications (e.g., range query) [34] and are highly vulnerable to the aforementioned inference attacks which greatly limit their applications in practice. Therefore, we consider the potential of generalizing our multi-view approach to various data types (e.g., geo-locations, DNA sequences, market basket data, numeric data and timestamps) while providing an additional layer of protection against the inference attacks. For instance, location data includes the two-dimensional latitude and longitude coordinates of different places, which are highly precise float numbers (up to 8 decimal digits). In the *Bing Map Tiles System*,[1] the map is recursively divided into four tiles with equal size to reach the required resolution for quick map zoom in/out. Motivated by such a hierarchical data or structures, we can encode the coordinates into bit strings by concatenating the index of each level for one specific location , e.g., the coordinates of "New York" are (40.730610, −73.935242). At the 23rd level, the pixel coordinates are (632700926, 807272436) and tile coordinates are (2471487, 3153407). Then, the encoding bit string can be derived as "e1147b6afff" in hexadecimal format.

In the encoded bit strings for coordinates, if prefixes can be preserved in the encrypted locations (while preserving the privacy), utility can be significantly preserved for analysis. 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 length of prefix (e.g., two other places in London with the same level of proximity). Thus, the structure of the locations and the distance between such locations (besides other features such as frequency for property preserving encryption and deterministic encryption) can be preserved in the outsourced data. Next by feeding this numeric encoded version of the original data to our multi-view approach and subsequently decoding the numeric analysis results as shown in Figure 11, data owners can benefit from accurate analysis results while being protected from a variety of inference attacks on PPEs [58–61].

We note that the utility of the location data under the multi-view solution will only be partially preserved, depending on the common prefixes which can be determined by different partitioning

[2]https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

schemes and the privacy protection level requested by the data owner. Like the network data, the prefixes are only preserved inside the same partition under the multi-view solution. Thus, the distances between locations can be preserved inside the same partition (with the common prefixes). Given larger partitions, more utility will be preserved with longer common prefixes in the output. Nonetheless, applying the multi-view solution with property preserving encryption can still be found useful in many applications as the semantic of the data is mostly preserved for different types of analyses.

5.2 Multi-view and Differential Privacy

Differential privacy (DP) has emerged as a de facto standard privacy notion for a wide range of applications [50, 51]. By requiring the presence of any individual's data in the input to only marginally affect the distribution over the output, differential privacy provides strong protection against adversaries with arbitrary background knowledge about the individuals. Differentially

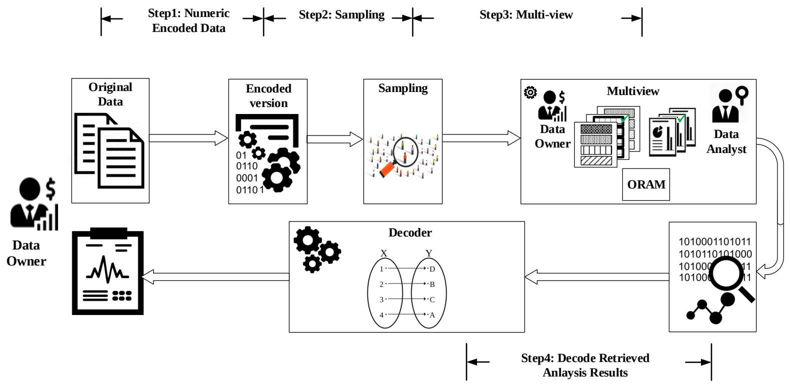


Fig. 12. Multi-view can be remodeled (using sampling) to achieve differential privacy

private mechanisms are typically special-purpose algorithms developed for specific applications, e.g., [45, 50, 51]. Many of those existing works address the interactive scenario, i.e., they provide perturbed answers to (limited sets of) queries. In contrast, micro-data publishing methods aim to release a sanitized dataset that supports a variety of use cases [12]. On the other hand, development of non-interactive methods which satisfy differential privacy while retaining sufficient data quality has remained challenging. Previous work has shown that algorithms which draw a random sample of data followed by generic k-anonymization can fulfill differential privacy [46, 47, 49]. These results are notable, as they combine statistical disclosure control, data anonymization and differential privacy. Specifically, Ninghui li et al. [46] show that any strongly-safe k-anonymization algorithm satisfies (like the multi-view approach) $(\beta, \epsilon, \delta)$-DPS (Differential Privacy under Sampling) for any $0 < \beta < 1, \epsilon \geq -\log(1-\beta)$, and $\delta = d(\beta, \epsilon, \delta)$, where d is a function of other parameters. Based upon these approaches, we consider the application of the multi-view approach for implementing a non-interactive scheme with differential privacy guarantee. Li et al. [46] showed that sampling records and reducing the uniqueness of their features through generic k-anonymity ensure Differential Privacy.

The multi-view approach can provide such a generic (independent from the dataset) k-anonymity solution through generating multiple views from one view. Specifically, since all attributes other than IPs are kept intact, and since the inter-partition prefix relation between IPs is not preserved, by properly grouping IP values (so that IP partitions become identical), multi-view can provide such generic k-anonymity, where k is the number of views. We note that here the number of views must always be in order of m! where m is the number of partitions after sampling. This is because the multi-view solution to be a perfect k-anonymous algorithm must generate all possible instances of a k-anonymous data. Figure 13 illustrates different steps of generating a $(0.35, 8 \times 10^{-6})$-DP network trace out of an original trace with four records by means of 0.4-sampling and two view generation.

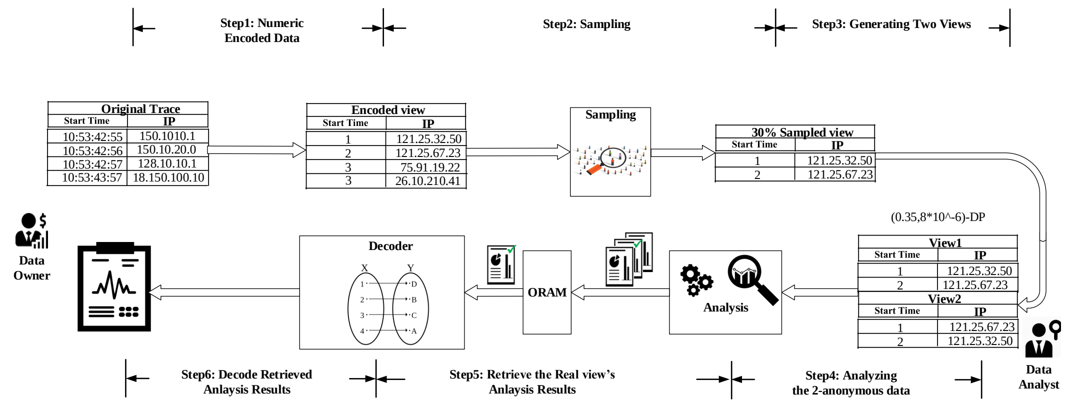


Fig. 13. An example of differentially private network trace generation using multi-view

6 EXPERIMENTS

This section evaluates our multi-view scheme through experiments with real data.

6.1 Setup

To validate our multi-view anonymization approach, we use a set of real world network traces collected by an anonymous ISP. We focus on attributes *Timestamp*, *IPaddress*, and *PacketSize* in our experiments, and the meta-data are summarized in the table in Figure 14(a).

Attribute	Value
Type Header	Header
Start date	August 1st, 2016
End date	October 21st, 2016
Size	70 GB
Format	pcap and TXT
Distinct IP addresses	883
Number of packets	1M

(a)

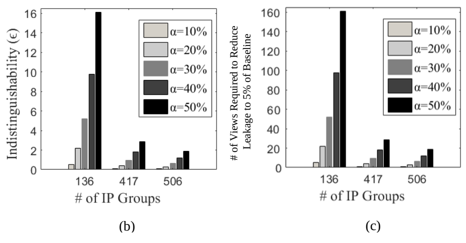


Fig. 14. (a) Metadata of the collected traces (b) ϵ for different number of prefix groups and different adversary knowledge, and (c) the required number of views to reduce the leakage of CryptoPAn down to 5%

In order to measure the security of the proposed approach, we implement the frequency analysis attack [88], [9]. This attack can compromise individual addresses protected by existing prefix-preserving anonymization in multi-linear time [9]. We note that in the setting of EDBs (encrypted database systems), an attack is successful if it recovers even partial information about a single cell of the DB [88]. Accordingly, we define the information leakage metric to evaluate the effectiveness of our solution against the adversary's semantic attacks. Several measures have been proposed in literature [8, 57] to evaluate the impact of semantic attacks. Motivated by [8], we model the information leakage (number of matches) as the number of records/packets, their original IP addresses are known by the adversary either fully or partially. More formally,

Information leakage metric [8]: We measure F_i defined as the total number of addresses that has at least i most significant bits known, where $i \in \{1, 2, \cdots, 32\}$.

To model adversarial knowledge, we define a set of prefixes to be known by the adversary ranging from 10% up to 100% of all the prefixes in the trace. This knowledge is stored in a two dimensional vector that includes α different addresses and their key indexes. Next, using our multi-view schemes, we generate all the N views. Before we apply the frequency analysis attack, we simulate how an

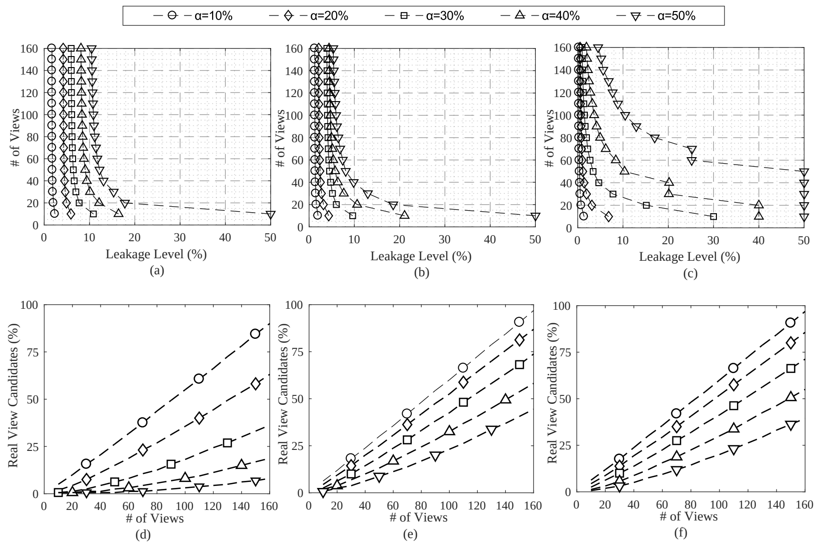


Fig. 15. Percentage of the compromised packets in schemes II, III (out of 1M) and number of real view candidates when number of views and the adversary knowledge vary and for the three different cases (1) Figures (a),(d) (2) Figures (b),(e) (3) Figures (c),(f)

adversary may eliminate some fake views from further consideration as follows. For each view, we check if two addresses from the adversary's knowledge set with different prefixes now share prefixes in that view. If we find such a match in the key indices, the corresponding view will be discarded from the set of the real view candidates and will not be considered in our experiments since the adversary would know it is a fake view.

In the following, we present an extensive set of experiments evaluating the utility and the performance of the multi-view approach. Moreover, we justify the choice of ORAM in our setup using a comprehensive study on the scalability of ORAM in the literature. We validate the effectiveness of our schemes by showing the number of real view candidates and the percentage of the packets in the trace that are compromised (i.e., the percentage of IP packets whose addresses have at least eight most significant bits known). Each experiment is repeated more than 1,000 times and the end results are the average results of the frequency analysis algorithm applied to each of the real view candidates. Finally, we evaluate the performance of our solution by measuring memory and CPU consumption as well as the time required to run our solution for one million packets while increasing the number of generated views. We conduct all experiments on a machine running Windows with an Intel(R) Core(TM) i7-6700 3.40 GHz CPU, 4 GB Memory, and 500 GB storage.

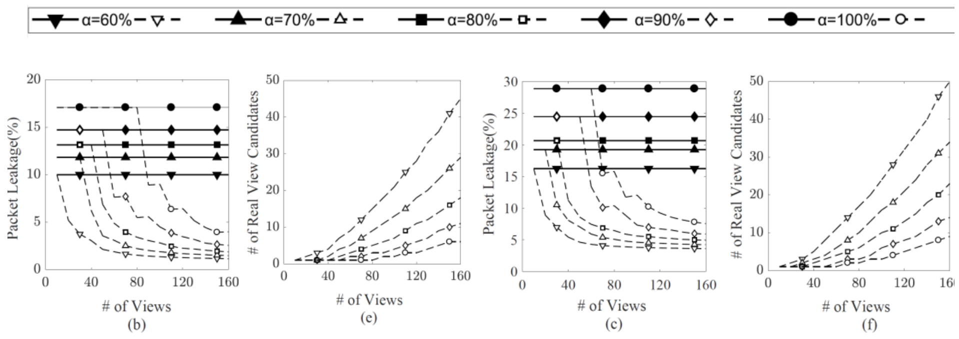


Fig. 16. Percentage of the compromised packets in schemes II, III (out of 1M) and number of real view candidates as number of views and the adversarial knowledge vary and for case (1) Figures (b),(e), and (2) Figures (c),(f) where *CP* denotes the CryptoPAn result and *MV* denotes the multi-view results

6.2 Information Leakage Analysis

First, the numerical results of the indistinguishability parameter ϵ, and the required number of views to reduce the leakage of CryptoPAn down to 5%, under different levels of adversary knowledge are depicted in Figure 14 (b,c). Those results correspond to three different cases, i.e., when addresses are grouped based on (1) only the first octet (136 groups), (2) the first and the second octets (417 groups), and (3) the first three octets (506 groups). As we can see from the results, ϵ decreases (meaning more privacy) as the number of prefix groups increases, and it increases as the amount of adversarial knowledge increases.

We next validate those numerical results through experiments by measuring the required number of views to reduce the leakage of CryptoPAn down to a certain level. Specifically, we first analyze the behavior of our second and third multi-view schemes (introduced in Sections 4.2 and 4.3) before comparing the information leakage of the three schemes in Section 7. Figure 15 presents different facets of information leakage when our approach is applied in various grouping cases. The results in Figure 15 are for adversaries who has knowledge of no more than 50% of the prefix groups (Figure 16 presents the more extreme cases for the same experiments, i.e., up to 100% knowledge). The analysis of these figures is detailed in the following.

Effect of the number of prefix groups: As discussed earlier, three different IP grouping cases are studied. Figures 15 (a) and (d) show respectively the results of packet leakage and number of real view candidates when $d = 136$. As the numerical results in Figure 8 anticipate, because the fraction $d/D = 0.154$ is relatively low, the indistinguishability of generated views diminishes especially for stronger adversary knowledges. Consequently, the adversary discards more views and the rate of leakage increases, compared with Figures 15 (b), (e) and Figures 15 (c), (f) for which the fraction d/D is 0.47 and 0.57, respectively. In particular, for the worst case of 50% adversary knowledge and when the number of views is less than 50, we can verify that the number of real view candidates for case (1) remains one resulting in packet leakage comparable to that of CryptoPAn.

Effect of the number of views: As it is illustrated in Figure 15, increasing the number of views always improves both the number of real view candidates and the packet leakages. Figure 15 (d-f), show a near linear improvement for real view candidates evaluation, where the slope of this improvement inversely depends on the adversary's knowledge. For the packet leakages, we can observe that the improvement converges to a low packet leakage rate under a large number of views. This is reasonable, as each packet leakage result is an average of leakages in all the real view candidates. However, since each of the fake views leaks a certain amount of information, increasing the number of views beyond a certain value will no longer affect the end result. In other words, the packet leakage converges to the average of leakages in the (fake) real view candidates. Finally, the results show that our proposed scheme can more efficiently improve privacy by (1) increasing the fraction d/D (*number of views/number of distinct addresses*) or (2) increasing the number of views. The first option may affect utility (since inter-group prefix relations will be removed), while the second option is more aligned with our objective of trading off privacy with computation. Specifically, Figure 15 shows that to reduce the leakage down to 10%, generating only 40 views in all the configurations is effective.

Privacy against very strong adversaries: Figure 16 shows the leakage and the real view candidates results for stronger adversaries ($\alpha \in [60, 100]$). Note that Figure 16 only shows results for case (2) and (3) because our evaluation for case (1) does not show a significant improvement compared with CryptoPAn results due to the fact that the multi-view approach with fraction of $d/D = 0.154$ is not effective against very strong adversaries ($\epsilon > 16$).

6.3 Utility Analysis

We now evaluate the utility of the approach using a real network dataset (1M records). For this purpose, we implemented a tool that can parse a network trace, anonymize it w.r.t. an anonymization method from the set of well-known methods including black marker, truncation, random shift [27] (refer to Section 8 for details on existing anonymization methods), and our multi-view approach, and finally evaluate the utility of the output using the set of well-known analyses, e.g., IP distribution, packet length distribution, heavy hitter analysis, throughput [36]. To make our discussion more concrete, we consider two disjoint categories of analysis, namely, all types of statistics (1) merely based on the packet characteristics, e.g., timestamp, packet size, etc. or (2) involving IP addresses or subnets that are more important in defining network behavior. Correspondingly, our utility evaluation results are divided into the following two categories.

6.3.1 Statistics on fp-QI attributes. Multi-view does not change any of the fp-QI attributes in a network trace, and its methodology focuses on IP addresses, as discussed in Section 2. Thus, all kinds of statistics merely depending on fp-QI attributes over each of the generated views are identical to those over the original trace. For instance, Figure 17 shows the results of *empirical cumulative distribution function (CDF)* for the three traces, i.e., (a) the original trace and the real view and (b) one of the fake views. Our results clearly show that all results are identical since multi-view will not have any impact on fingerprinting quasi-identifier attributes.

6.3.2 Statistics related to IP addresses. In the second set of experiments, we study the impact of our multi-view over several types of analyses which are related to IP addresses from the networking literature. We note that while we consider a wide range of analyses, our experiences may not be representative. Moreover, each of our reproductions is only one of many possible ways of reproducing an analysis; different ways of measuring the same quantity may lead to different results. Our evaluation considers four types of weighted IP topology, namely, (1) distribution of distinct IP addresses in different subnets (the weighted topology 1), (2) frequency of the records per subnet (the weighted topology 2), (3) traffic throughput per subnet (the weighted topology 3), and (4) packet throughput per subnet (the weighted topology 4).

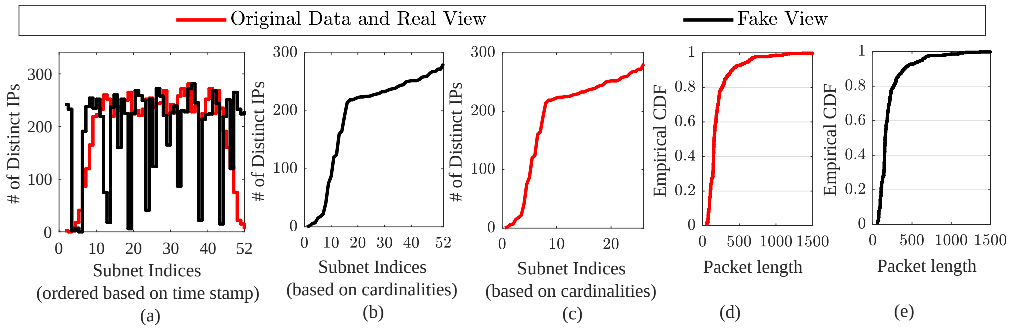


Fig. 17. Distribution of distinct IP addresses in different subnets (a-c) (the weighted topology 1) and overall distribution of the packet lengths in (d) the real view and the original trace (e) one of the fake views

Distribution of distinct IP addresses in different subnets: Figure 17 presents a sample IP distribution [4] in the trace, which represents the number of distinct addresses within each subnet (IP group). We compare the distribution of distinct IP addresses inside the original trace, real view and one of the fake views for both *temporal* distribution (if subnets are indexed based on their timestamps), and *cardinality-based* distribution result (if subnets are indexed based on their cardinalities). As shown in Figure 17(a), while the fake view hides this type of IP topology of the network trace, the real view can preserve the topology since the real view is a prefix preserving mapping of IPs. Moreover, the cardinality-based distribution results, generated from the fake view is identical to those in the original trace and the real view (see Figure 17(b)) which is a result of the imposed indistinguishability in scheme II or III.

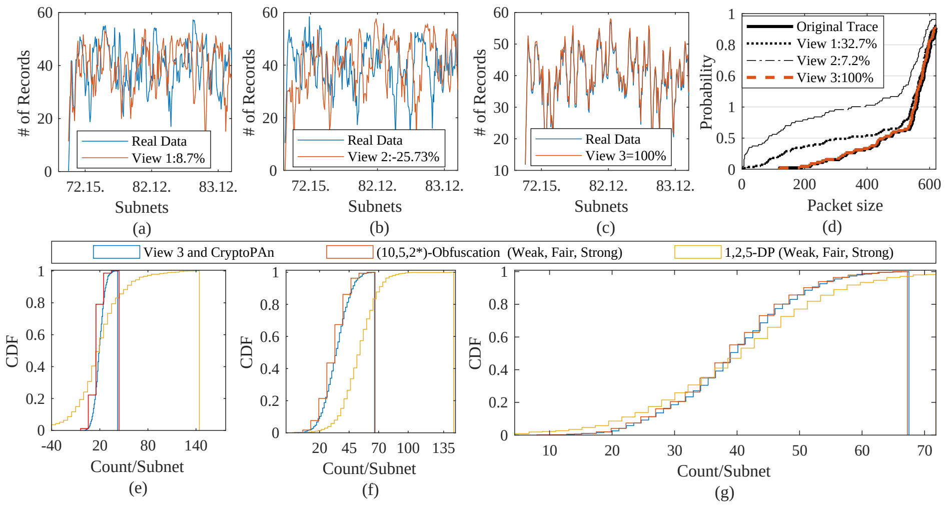


Fig. 18. Evaluating frequency of the records per subnet (the weighted topology 2) for different views (a-c), and comparing the CDFs in different views (d), and different privacy metrics (e-g)

Distribution of frequency of the records in different subnets: Figure 18 presents records distribution in a sample collected from the trace, which represents the number of records within

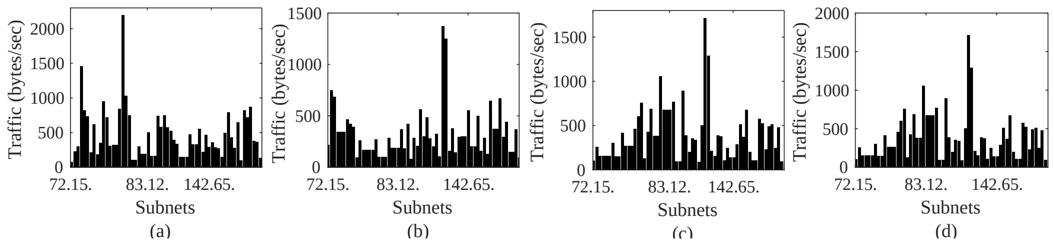


Fig. 19. Evaluating the traffic throughput per subnet (the weighted topology 3) for (a),(b) fake views, (c) real view and (d) original trace

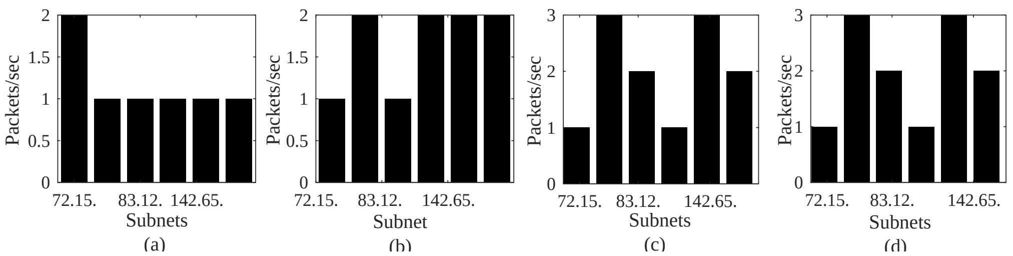


Fig. 20. Evaluating the packet throughput per subnet (the weighted topology 4) for (a),(b) fake views, (c) real view and (d) original trace

each subnet (IP group). This analysis is important in studying various applications, e.g., heavy hitter identification [36], anomaly detection and certain types of attacks like distributed denial of service attack (DDoS) [4]. We compare the results of the distribution for a set of three generated views (a-c) using our multi-view approach where the third view is the real view. We observe that hat, while the real view can completely preserve the statistics, other views incur significant errors. Thus, we conclude that multi-view can also preserve privacy of some types of analysis results. Furthermore, a cumulative distribution function (CDF) of the frequencies is reported in this figure from which we can draw a similar observation. Moreover, Figure. 18 (e-g). compare the accuracy of IP distribution in (1) the multi-view approach, (2) DP with three categories of privacy guarantees (weak, medium and strong regimes) and (3) (k,j)-obfuscation of Riboni et al. [7]. The results clearly show that for most of analyses our approach outperforms other solutions with relatively weak privacy protections.

Distribution of throughput in different subnets: Figure 19 presents the throughput distribution [4] in a sample of the trace, which represents the rate of the traffic over each subnet (IP group). This analysis is not only important in studying various applications, e.g., heavy hitter identification and computing flow properties such as round trip time (RTT) [4] but also able to reveal confidential information about network activities of the data owners. We compare the distribution results for the three generated views and we observe that, while the real view completely preserves this statistics, other views incur significant errors. Thus, we conclude that multi-view preserves both privacy and utility for this analysis.

Distribution of number of packets per seconds in different subnets: Figure 20 presents the distribution of the number of packets per second in different subnets, which represents the rate of forwarding and receiving packets within each subnet (IP group). All histograms correspond

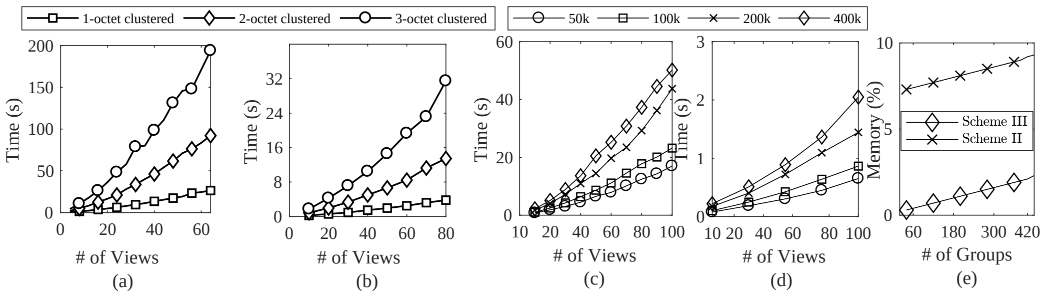


Fig. 21. Computation time of schemes II, III for (a,b) different prefix grouping cases, and (c,d) different sizes of dataset, and memory consumption of the two schemes when generating 20 views

to 2 minutes minute traces collected at the same time of the day. This analysis is important in studying various applications, e.g., certain types of attacks especially, worm fingerprinting [36]. We compare the distribution results for the three generated views. We observe that while the real view completely preserves this statistics, other views incur significant errors. However, the distribution of the packets per seconds in different subnets are designed in a way that the views become indistinguishable.

6.4 Performance analysis

In this section, we evaluate the computational cost of our proposed solution in terms of time and memory to investigate its scalability.

Computation time required for generating views: Figure 21 shows the results obtained from running our solution on one million packets while varying the number of generated views. We evaluate the computational overhead incurred by our approach. Figure 21 (a) and (b) show the time required for three different grouping cases under scheme II and scheme III, respectively. We observe that, when the number of views increases, the computational overhead increases nearly linear. However, each case shows a different slope depending on the number of groups. This is reasonable as our schemes generate key/permutation vectors with a larger number of elements for more groups, which leads to applying CryptoPAn/permutation for a larger number of times. Furthermore, we observe that the time required for scheme III to generate a certain number of views is around 10% of the time for scheme II since applying permutation is much faster than cryptographic operations in CryptoPAn. Finally, linking the results in this figure to the information leakage results shown in Figure 15 demonstrates the trade-off between privacy and computational overhead.

Time and memory overhead for different sizes of data: We evaluate the computation time when varying the size of the dataset. Figure 21 (c), (d) depicts the computation costs of schemes II, III, respectively, when varying the number of views and for different sizes of the dataset. We observe that by increasing the number of views, the computation time increases with a slope depending on the size of the dataset. However, this slope does not increase monotonically. For instance, we observe a bigger jump in the slope of the results of $100k$ to $200k$ than the results of $200k$ to $400k$. This is reasonable as the most important parameter of a dataset in determining the computation time is the number of distinct IP addresses, which does not necessarily increases linearly by size of the dataset. Furthermore, we compare the memory consumption in schemes II and III when generating 20 views. Our results shown in Figure 21 (e) demonstrate that both schemes require a reasonable amount of memory where this the requirement of scheme III is 75% less.

Computation time required for different types of analysis: Finally, we present results on the computation time required for conducting a set of three analyses, when varying the number of views. We find that there is a significant difference between the observations drawn from different types of the analysis. In particular, our results in Table 2 show that the computation time required for conducting an analysis which is merely based on fp-QI, e.g., the overall throughput, is constant for any number of views because the multi-view approach keeps those attributes intact. In contrast, those analyses that depend on IP addresses require a linearly increasing computation time. However, for those analyses that do not directly depend on IP addresses, e.g., most of the subnet level statistics discussed earlier, the computation time can be saved significantly through techniques like caching. Therefore, the practicality of N times computation will mainly depend on the type of analysis.

Table 2. Computation time per second for different types of analysis

# of Views	Subnet Level	Overall Throughput	IP Level
20	7.62	8.49	71.8
40	11.15	8.49	120.3
60	14.01	8.49	160.34
80	17.42	8.49	205.13

6.5 Implementing ORAM in Multi-view

The last step of our solution requires a data owner to privately retrieve an audit report of the real view, which can be based on existing private information retrieval (PIR) techniques. A PIR approach usually aims to conceal the objective of all queries independent of all previous queries [39, 68]. Since the sequence of accesses is not hidden by PIR while each individual access is hidden, the amortized cost is equal to the worst-case cost [39]. Since the server computes over the entire database for each individual query, the results can become impractical. On the other hand, ORAM [69] has verifiably low amortized communication complexity and does not require much computation on the server but rather periodically requires the client to download and reshuffle the data [39]. For our multi-view scheme, we choose ORAM as it is relatively more efficient and secure, and also the client (data owner in our case) has sufficient computational power and storage needed to locally store a small number of blocks (audit reports in our case) in a local stash. In practice, we expect that the analysis reports would have significantly smaller sizes in comparison to the views, and considering the one round communication with ORAM ($O(logN)$-complexity), we believe the solution would have acceptable scalability. Experiments using our dataset and existing ORAM implementation (an implementation [89] of non-recursive Path-ORAM [91] has been made public) have further confirmed this. We generated various sets of analyses reports using *snort* [90]. Our large-scale experimental dataset only results in *Kilobytes*-level audit reports, which can be practically used with fast ORAM protocols, e.g., Path-ORAM [89]. Specifically, for Path-ORAM, Figure 5 (b) in [89] shows a less than 1MB communication overhead for the worst-case cost of up to 2^{24} number of blocks of size 4KB.

7 DISCUSSIONS

In this section, we discuss various aspects and limitations of our approach.

(1) **Exploring background knowledge:** We agree that not all possible attacks on network traces can be addressed by our scheme. As mentioned above, if adversaries possess arbitrary background knowledge, then we would require a DP-based solution, e.g., Mcsherry et al.

[36] (which prevents access to raw records and can only support a limited range of analysis). Since this work specifically focuses on network trace anonymization (instead of data privacy in general), we have followed the literature on network trace anonymization to consider the most widely studied threat model (adversarial knowledge of subsets of records or frequency distribution of IP prefixes) as well as utility requirement (releasing a prefix-preserving version of the raw trace) [7, 8, 9, 13, 14, 16, 17, 31]. Such threat model and utility requirement are common since they reflect the most plausible threats in networks (adversaries in a network can either passively observe the traffic in their own subnets, hence the knowledge of subsets of records or frequency distribution, or they can also actively fabricate malicious traffic, hence the injection attack), as well as the common needs for analysis (e.g., analyzing individual records for network anomaly detection or IP traceback). We emphasize that the multi-view solution does not require estimating the exact knowledge (which subsets or prefixes are known) of adversaries, which is certainly impractical, but only the percentage of known data (denoted as alpha-knowledge in the paper). Estimating the value of alpha is similar to estimating the epsilon value of DP, which depends on the desired level of protection (larger alpha means more protection). However, different from most existing solutions (which would provide less utility for more protection), a unique advantage of the multi-view solution is that, as the level of protection increases (with larger alpha), we can still maintain the same level of utility (at the cost of more computation for generating more views).
On the other hand, some attacks are not mentioned in the paper but still can be handled by the multi-view solution, e.g., port-based [14], known mapping [16] and machine attribute [17] (those attacks share some characteristics with the injection and fingerprinting attacks by identifying individual machines/IPs in the logs). Moreover, there also exist other attacks which cannot be handled by the existing multi-view solution, but it is feasible to extend the solution to mitigate them. For instance, behavioral profiling on network logs based on distributional adversary knowledge (the overall distribution of certain attribute, e.g., port, and packet size) can be handled by an extended multi-view approach with a redefined partitioning algorithm. Finally, the general idea of multi-view can certainly find applications beyond the domain of network trace protection, and extending the solution with DP is indeed an interesting future direction.

(2) **Application to EDB:** We believe the multi-view solution may be applicable to other related areas. For instance, processing on encrypted databases (EDB) has a rich literature including searchable symmetric encryption (SSE) [81], [82], fully-homomorphic encryption (FHE) [83], oblivious RAMs (ORAM) [72], functional encryption [84], and property preserving encryption (PPE) [85], [86]. All these approaches achieve different trade-offs between protection (security), utility (query expressiveness), and computational efficiency [88]. Extending and applying the multi-view approach in those areas could lead to interesting future directions.

(3) **Comparing the three schemes:** As we discussed earlier, scheme I achieves a better indistinguishability but less protected partitions in each view. Figure 22 compares the relative effectiveness of the three schemes on a real trace under 40% adversary knowledge. In particular, Figure 22 (a), (b) demonstrate the fact that despite the lower number of real view candidates in schemes II and III compared with scheme I (30 vs 160 out of 160), the end results of the leakage in schemes II and III is much more appealing (3% vs 35%). Therefore, our experimental section has mainly focused on schemes II and III.

(4) **Anonymizing more attributes:** In the context of network trace anonymization, IP addresses are generally regarded as the main quasi identifier and have been the central focus of the literature. On the other hand, the multi-view concept (as 2nd layer of protection) can be applied on top of different anonymization techniques (as 1st layer), and not limited to

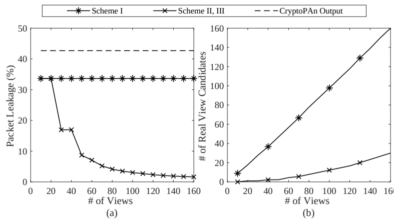


Fig. 22. Comparison between the privacy of scheme I vs. schemes II and III with 137 partitions (prefix groups based on the first octet sharing)

only IP prefix preserving scheme (CryptoPAn) as studied in this paper. For instances, since CryptoPAn is a bit-wise operation, it can be potentially extended to other numeric-value attributes, e.g., timestamps, port number and packet sizes, with appropriate indices in order to allow the multi-view solution to work on those attributes.
As to anonymizing multiple attributes simultaneously, there exists a trade-off between the number of attributes to be anonymized and the overall privacy protection. Specifically, the more attributes are anonymized, the lower probability for the adversary to successfully identify the records due to more limited information that can be used for reconstruction via injection and fingerprinting. However, he/she with a higher probability can discard fake generated views because the chance of detecting more inconsistencies among correlated attributes can increase.

(5) **Choosing the number of views** N**:** The number of views N is an important parameter of our approach that determines both the privacy and computational overhead. Specifically, as it is implied by Lemma 4.2 and demonstrated in our experimental results in Section 6, the number of real view candidates is approximately $e^{-\epsilon} \cdot N$. Moreover, the overhead is almost proportional to the number of views because most of the operations are sublinear. However, we stress that in general, generating 100 views can effectively protect the information leakage of CryptoPAn. This reduction squeezes the adversary's post knowledge down to 1-3% of the leakage appears in our baseline, when assuming an unrealistically very strong adversary who has prior knowledge about great number of subnets. Moreover, Figure. 21 and Table. 2 present the computation time of view generation and conducting different categories of analysis (both on the analyst's side). These results clearly show the practicality of the Multi-view approach especially when instead of redundantly applying CryptoPAn an efficient rotation of partitions is employed (scheme III). The data owner could choose this value based on the level of trust on the analysts and the amount of computational overhead that can be afforded. An alternative solution is to sacrifice some utility (by giving up some prefix relations among IPs) through increasing the number of prefix groups (D), e.g., grouping them based on the first 3 octets.

(6) **Utility:** The main advantage of the multi-view approach is it can preserve the data utility while protecting privacy. In particular, we have shown that the data owner can receive an analysis report based on the real view (Γ_r) which is prefix-preserving over the entire trace. This is more accurate than the obfuscated (through bucketization and suppression) or perturbed (through adding noise and aggregation) approaches. Specifically, in case of a security breach, the data owner can easily compute $\mathcal{L}_r$ (migration output) to find the mapped IP addresses corresponding to each original address. Then the data owner applies necessary security policies to the IP addresses that are reported violating some policies in Γ_r.

A limitation of our work is it only preserve the prefix of IPs, and a potential future direction is to apply our approach to other property-preserving encryption methods such that other properties may be preserved similarly.

(7) **Communicational and computational cost:** One of our contributions in this paper is to minimize the communication overhead by only outsourcing one (seed) view and some supplementary parameters. This is especially critical for large scale network data like network traces from the major ISPs. On the other hand, one of the key challenges to the multi-view approach is that it requires N times computation for both generating the views and analysis. Our experiments in Figure 21 shows that generating 160 views for a trace of $1milion$ packets takes approximately 4 minutes and we describe analytic complexity results in Tables 3 and 4. These tables present overhead analysis, from both the data owner's and the data analyst's side. In particular, table 3 summarizes the overhead for all the action items in the data owner side. Here, $C(n)$ is the computation overhead of CryptoPAn and D is the number of the distinct IP addresses. Finally, table 4 summarizes the overhead for all the action items in the data analyst side where $N \cdot CV(n)$ is the cost of N times verifying the compliances (auditing).

Table 3. Overhead on the data owner side

Blocks in Multi-view	Computation Overhead	Communication Overhead
Initial anonymization	$C(n)$	–
Migration function	$O(nlog^n) + \frac{\sum_{i=1}^{d} C(i)}{d} C(n)$	–
Prefix grouping	–	–
Index generator	$N \cdot O(D)$	$N \cdot O(D)$
Seed trace	$\frac{\sum_{i=1}^{D} V_0(i)}{D} C(n)$	$O(n)$
Report retrieval (ORAM)	–	$O(log^N)\omega(1)$

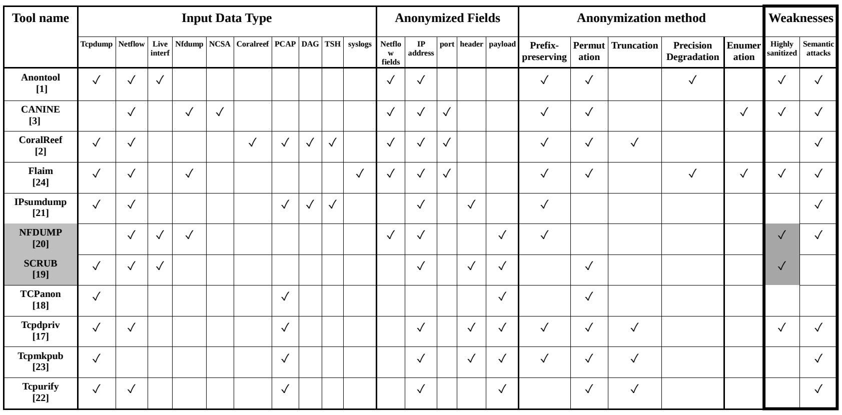
Table 4. Overhead on the data analyst side

Blocks in Multi-view	Computation Overhead	Communication Overhead
Seed view	–	$O(n)$
N views generation	$\frac{\sum_{i=1}^{N} \sum_{j=1}^{D} V_i(j)}{D} C(n)$	–
Compliance verification (Analysis)	$N \cdot CV(n)$	–

We note that the practicality of N times computation will mainly depends on the type of analysis, and certainly may become impractical for some analyses under large N. How to enable analysts to more efficiently conduct analysis tasks based on multiple views through techniques like caching is an interesting future direction. Another direction is to devise more accurate measures for the data owner to more precisely determine the number of views required to reach a certain level of privacy requirement.

8 RELATED WORK

In the context of anonymization of network traces, as surveyed in [37], many solutions have been proposed [1–3, 22–29]. In Figure 23, we have summarized the scope (e.g., accepted input data, anonymization fields, etc.) of some popular tools in anonymizing network traces. Generally, these may be classified into different categories, such as *enumeration* [38], *partitioning* [40], and *prefix-preserving* [41, 44]. These methods include removing rows or attributes (suppression) and generalization of rows or attributes [64]. Some of the solutions [16, 57] are designed to address specific attacks and are generally based on the permutation of some fields in the network trace to blur the adversary's knowledge. Later studies either prove theoretically [13] or validate empirically [31] that those works can be defeated by semantic attacks. There are only two tools that can resist



Tool name	Input Data Type										Anonymized Fields					Anonymization method					Weaknesses	
	Tcpdump	Netflow	Live interf	Nfdump	NCSA	Coralreef	PCAP	DAG	TSH	syslogs	Netflow fields	IP address	port	header	payload	Prefix-preserving	Permut ation	Truncation	Precision Degradation	Enumer ation	Highly sanitized	Semantic attacks
Anontool [1]	✓	✓	✓								✓	✓				✓	✓		✓		✓	✓
CANINE [3]		✓		✓	✓						✓	✓	✓			✓	✓			✓	✓	✓
CoralReef [2]	✓	✓				✓	✓	✓	✓		✓	✓	✓			✓	✓	✓				✓
Flaim [24]	✓	✓		✓						✓	✓	✓	✓			✓	✓		✓	✓	✓	✓
IPsumdump [21]	✓	✓					✓	✓	✓			✓		✓		✓						✓
NFDUMP [20]		✓	✓	✓							✓	✓			✓	✓					✓	✓
SCRUB [19]	✓	✓	✓									✓		✓	✓		✓				✓	
TCPanon [18]	✓						✓								✓		✓					
Tcpdpriv [17]	✓	✓					✓					✓		✓	✓	✓	✓	✓			✓	✓
Tcpmkpub [23]	✓						✓					✓		✓	✓	✓	✓	✓				✓
Tcpurify [22]	✓	✓					✓					✓			✓		✓	✓				✓

Fig. 23. Network trace anonymization tools and semantic attacks

semantic attacks, i.e., SCRUB [24] and TCPanon [23]. Unfortunately, as shown in Figure 23, these two tool require heavy sanitization which render the released data less useful. We now review some important categories of network trace anonymization methods [27].

1. **Format-preserving encryption [11].** or (pseudo-)random permutation of data can map original data to to an encrypted version in the space of the original data. An example is the format preserving encryption of credit-card numbers, which facilitates compatible input for existing devices yet provide some additional security against eavesdropping. Format-preserving encryption of IP or MAC addresses is extremely desirable in network trace anonymization as the anonymized version can be parsed by IDS or other network tools.
2. **Prefix preservation [8].** Since prefixes in network traffic have special meanings, e.g., the first 6 bytes of a MAC address field (manufacturer) or the leading bytes of an IPv4 address (registered subnet), performing prefix preserving anonymization is highly desirable in that context. One big problem with this method is that the frequency is preserved due to the use of deterministic encryption, which can easily leak information about the true records since the attacker may know some prior background information of the frequency distribution. Specifically, Naveed et al. [88] present a series of attacks that recover the plaintext from deterministic encryption (DTE) and order-preserving encryption (OPE); using only the encrypted column and publicly-available auxiliary information. They have considered well-known attacks, including frequency analysis and sorting, as well as new attacks based on combinatorial optimization.
   However, these attacks under multi-view can be successfully mitigated. We note that only the real view would preserve the true frequency of the records inside the dataset. Therefore, an adversary cannot apply existing frequency-based attacks as usual. Adversaries armed with some prior knowledge about frequencies can at most (as shown in Section 4.3.2) discard some of the fake views, and the remaining views will still be indistinguishable and render the frequency attacks difficult. To evaluate the impact of such an attack, our experiments have been based on the frequency analysis attack of Brekene et al. [13], which is a sophisticated attack for optimally leveraging the background knowledge to infer results using frequency analysis.
3. **Replacement [30].** This method applies a one-to-one mapping of a field to a new value of the same type. Moreover, to provide enough flexibility replacement is often applied with regular expressions and is suitable for both headers and full packet [19].

4. **Filtering and data removal [10].** Also called as truncation and black marking, results in data removal by overwriting it with fixed values, often zeros.
5. **Generalization [29].** is the act of replacing a data with more general data through partitioning (also called grouping or binning) information. For instance, TCP/UDP port numbers can be presented as ephemeral ($\geq$ 1024) or non-ephemeral ($<$ 1024).
6. **Precision degradation [38].** This method is comparable with black marking, whereas by degradation only the least significant information of a data field is removed. Examples include precision degradation of timestamps to less specific values. Another example is rounding of numeric values.
7. **Enumeration [29].** is an example of collision resistant mapping which provides order preserving and uniqueness. For example, applying enumeration to timestamps keeps the order but lose precision or distance.
8. **Cryptographic permutation [13].** using a block cipher or hash function can be applied to uniquely permutate network data. The security of the cryptographic permutation depends largely on the input entropy. For instance, for very short values (e.g., 32 bit IPv4 addresses), a simple hashing is easily reversible due to lack of input entropy [9]. However, HMACs have better resistance to chosen plain text attacks than regular hashes [9].

Remarks. When choosing anonymization primitives, it is important to ensure compatible, and, to some degree, meaningful transformations. Moreover, applying an anonyization method requires certain actions to be taken so that a valid network packet will be recieved (by IDS), e.g., TCP sequence/acknowledgement numbers (if used), all relevant checksums, and IP packet length fields should be corrected to reflect the proper data lengths [48].

As our proposed anonymization solution falls into the category of prefix-preserving solutions, which aims to improve the utility, we review in more details some of the proposed solutions in this category. First effort to find a prefix preserving anonymization was done by Greg Minshall [76] who developed TCPdpriv which is a table-based approach that generates a function randomly. Fan et al. [8] then developed CryptoPAn with a completely cryptographic approach. Several publications [9], [16, 57] have then raised the vulnerability of this scheme against semantic attacks which motivated query based [36] and bucketization based [7] solutions.

Among the works that address such semantic attacks, Riboni et al. [7] propose a (k,j)-obfuscation methodology applied to network traces. In this method, a flow is considered obfuscated if it cannot be linked, with greater assurance, to its (source and destination) IPs. First, network flows are divided into either confidential IP attributes or other fields that can be used to attack. Then, groups of k flows having similar fingerprints are first created, then bucketed, based on their fingerprints into groups of size $j < k$. However, utility remains a challenge in this solution, as the network flows are heavily sanitized, i.e., each flows is blurred inside a bucket of k flows having similar fingerprints. An alternative to the aforementioned solutions, called *mediated trace analysis* [32, 35], consists in performing the data analysis on the data-owner side and outsourcing analysis reports to researchers requesting the analysis. In this case, data can only be analyzed where it is originally stored, which may not always be practical, and the outsourced report still needs to be sanitized prior to its outsourcing [36]. In contrast to those existing solutions, our approach improves both the privacy and utility at the cost of a slightly higher computational overhead. ,

9 CONCLUSION

In this paper, we have proposed a multi-view anonymization approach mitigating the semantic attacks on CryptoPAn while preserving the utility of the trace. This novel approach shifted the trade-off from between privacy and utility to between privacy and computational cost; the later has seen significant decrease with the advance of cloud computing, making our approach a more preferable solution for applications that demand both privacy and utility. We propose three different schemes for our multi-view approach to mitigate different types of attacks. Our experimental results showed that our proposed approach significantly reduced the information leakage compared to CryptoPAn. For example, for the extreme case of adversary pre-knowledge of 100%, the information leakage of CryptoPAn was 100% while under our approach it was still less than 10%. Besides addressing various limitations, our future works will adapt the idea to improve existing privacy-preserving solutions in other areas, e.g., we will extend our work to the multi-party problem where several data owners are willing to share their traces to mitigate coordinated network reconnaissance by means of distributed (or inter-domain) audit [79].

10 ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers for their valuable comments. This work was supported partially by the Natural Sciences and Engineering Research Council of Canada and Ericsson Canada under the Industrial Research Chair (IRC) in SDN/NFV Security. It is also partially supported by the National Science Foundation under Grant No. CNS-1745894 and the WISER ISFG grant.

REFERENCES

[1] Foukarakis, Michalis, Demetres Antoniades, Spiros Antonatos, and Evangelos P. Markatos. Flexible and high-performance anonymization of NetFlow records using anontool. In 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007, pp. 33-38. IEEE, 2007.

[2] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and K. C. Claffy. 2001. The CoralReef Software Suite as a Tool for System and Network Administrators. In Proceedings of the 15th USENIX conference on System administration (LISA '01). USENIX Association, Berkeley, CA, USA, 133-144.

[3] Li, Yifan & Slagell, Adam & Luo, Katherine & Yurcik, William. (2005). CANINE: A combined conversion and anonymization tool for processing NetFlows for security.

[4] Hautakorpi, Jani, and Gonzalo Camarillo Gonzalez. IP Address Distribution in Middleboxes. U.S. Patent Application No. 12/518,452.

[5] Meisam Mohammady, Lingyu Wang, Yuan Hong, Habib Louafi, Makan Pourzandi, and Mourad Debbabi. 2018. Preserving Both Privacy and Utility in Network Trace Anonymization. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18). ACM, New York, NY, USA, 459-474. DOI: https://doi.org/10.1145/3243734.3243809

[6] Ding, Wen, William Yurcik, and Xiaoxin Yin. Outsourcing internet security: Economic analysis of incentives for managed security service providers. In International Workshop on Internet and Network Economics, pp. 947-958. Springer Berlin Heidelberg, 2005.

[7] Riboni, Daniele, Antonio Villani, Domenico Vitali, Claudio Bettini, and Luigi V. Mancini. Obfuscation of sensitive data in network flows. In INFOCOM, 2012 Proceedings IEEE, pp. 2372-2380. IEEE, 2012.

[8] Xu, Jun, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In 10th IEEE International Conference on Network Protocols, 2002. Proceedings., pp. 280-289. IEEE, 2002.

[9] Brekne, T., Årnes, A., & Øslebø, A. (2005, May). Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In International Workshop on Privacy Enhancing Technologies (pp. 179-196). Springer Berlin Heidelberg.

[10] Slagell, Adam J., Yifan Li, and Katherine Luo. "Sharing network logs for computer forensics: A new tool for the anonymization of netflow records." In Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, 2005., pp. 37-42. IEEE, 2005.

[11] Dworkin, Morris. Recommendation for block cipher modes of operation: methods for format-preserving encryption. NIST Special Publication 800 (2016): 38G.

[12] Zhu, Tianqing, Gang Li, Wanlei Zhou, and S. Yu Philip. Differentially private data publishing and analysis: A survey. IEEE Transactions on Knowledge and Data Engineering 29, no. 8 (2017): 1619-1638.

[13] Brekne, Tønnes, and André Årnes. Circumventing IP-address pseudonymization. In Communications and Computer Networks, pp. 43-48. 2005.

[14] Yen, Ting-Fang, Xin Huang, Fabian Monrose, and Michael K. Reiter. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In International Conference on Detection of Intrusions and Malware, and

Vulnerability Assessment, pp. 157-175. Springer, Berlin, Heidelberg, 2009.

[15] Burkhart, Martin, Daniela Brauckhoff, Martin May, and Elisa Boschi. The risk-utility tradeoff for IP address truncation. In Proceedings of the 1st ACM workshop on Network data anonymization, pp. 23-30. ACM, 2008.

[16] Pang, R., Allman, M., Paxson, V. and Lee, J., 2006. The devil and packet trace anonymization. ACM SIGCOMM Computer Communication Review, 36(1), pp.29-38.

[17] Coull, Scott E., Michael P. Collins, Charles V. Wright, Fabian Monrose, and Michael K. Reiter. On Web Browsing Privacy in Anonymized NetFlows. In USENIX Security. 2007.

[18] Wong, Wai Kit, David W. Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. Security in outsourcing of association rule mining. In Proceedings of the 33rd international conference on Very large data bases, pp. 111-122. VLDB Endowment, 2007.

[19] Koukis, Dimitris, Spyros Antonatos, Demetres Antoniades, Evangelos P. Markatos, and Panagiotis Trimintzios. A generic anonymization framework for network traffic. In 2006 IEEE International Conference on Communications, vol. 5, pp. 2302-2309. IEEE, 2006.

[20] Ferrara, Ed, Christopher McClean, and Michael Caputo. The forrester wave: Managed security services: North america, q4, 2014. Forrester Research, November (2014).

[21] Tai, Chih-Hua, Philip S. Yu, and Ming-Syan Chen. k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 473-482. ACM, 2010.

[22] Minshall, Greg. Tcpdpriv. http://ita. ee. lbl. gov/html/contrib/tcpdpriv. html (1997).

[23] Universita degli Studi di Brescia. 2009. tcpanon. Retrieved November 7, 2017 from http://netweb.ing.unibs.it/tools/tcpanon/index.php

[24] Yurcik, William, Clay Woolam, Greg Hellings, Latifur Khan, and Bhavani Thuraisingham. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007, pp. 49-56. IEEE, 2007.

[25] Haag, Peter. Watch your Flows with NfSen and NFDUMP. In 50th RIPE Meeting. 2005.

[26] Jelena Mirkovic. 2008. Privacy-safe network trace sharing via secure queries. In Proceedings of the 1st ACM workshop on Network data anonymization (NDA '08). ACM, New York, NY, USA, 3-10. DOI: https://doi.org/10.1145/1456441.1456445

[27] Niels Van Dijkhuizen and Jeroen Van Der Ham. 2018. A Survey of Network Traffic Anonymisation Techniques and Implementations. ACM Comput. Surv. 51, 3, Article 52 (May 2018), 27 pages. DOI: https://doi.org/10.1145/3182660.

[28] Matt Roughan. 2006. Public review for the devil and packet trace anonymization. SIGCOMM Comput. Commun. Rev. 36, 1 (January 2006), 27-28. DOI=http://dx.doi.org/10.1145/1111322.1111329

[29] Slagell, Adam J., Kiran Lakkaraju, and Katherine Luo. FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs. In LISA, vol. 6, pp. 3-8. 2006.

[30] Foukarakis, Michael, Demetres Antoniades, and Michalis Polychronakis. Deep packet anonymization. In Proceedings of the Second European Workshop on System Security, pp. 16-21. ACM, 2009.

[31] Burkhart, Martin, Dominik Schatzmann, Brian Trammell, Elisa Boschi, and Bernhard Plattner. The role of network trace anonymization under attack. ACM SIGCOMM Computer Communication Review 40, no. 1 (2010): 5-11.

[32] Mogul, Jeffrey C., and Martin Arlitt. Sc2d: an alternative to trace anonymization. In Proceedings of the 2006 SIGCOMM workshop on Mining network data, pp. 323-328. ACM, 2006.

[33] Florian Kerschbaum. 2015. Frequency-Hiding Order-Preserving Encryption. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA, 656-667. DOI: https://doi.org/10.1145/2810103.2813629

[34] Liu, Dongxi, and Shenlu Wang. Nonlinear order preserving index for encrypted database query in service cloud environments. Concurrency and Computation: Practice and Experience 25, no. 13 (2013): 1967-1984.

[35] Mittal, Prateek, Vern Paxson, Robin Sommer, and Mark Winterrowd. Securing Mediated Trace Access Using Black-box Permutation Analysis. In HotNets. 2009.

[36] McSherry, Frank, and Ratul Mahajan. Differentially-private network trace analysis. In ACM SIGCOMM Computer Communication Review, vol. 40, no. 4, pp. 123-134. ACM, 2010.

[37] Mivule, Kato, and Blake Anderson. A study of usability-aware network trace anonymization. In 2015 Science and Information Conference (SAI), pp. 1293-1304. IEEE, 2015.

[38] Farah, Tanjila, and Ljiljana Trajković. Anonym: A tool for anonymization of the Internet traffic. In 2013 IEEE International Conference on Cybernetics (CYBCO), pp. 261-266. IEEE, 2013.

[39] Mayberry, Travis, Erik-Oliver Blass, and Agnes Hui Chan. Efficient Private File Retrieval by Combining ORAM and PIR. In NDSS. 2014.

[40] Slagell, Adam J., Kiran Lakkaraju, and Katherine Luo. FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs. In LISA, vol. 6, pp. 3-8. 2006.

[41] Xu, Jun, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In 10th IEEE International Conference on Network Protocols, 2002. Proceedings., pp. 280-289. IEEE, 2002.

[42] Wang, Xiao Shaun, Yan Huang, TH Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: oblivious RAM for secure computation. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 191-202. ACM, 2014.

[43] Yurcik, William, Clay Woolam, Greg Hellings, Latifur Khan, and Bhavani Thuraisingham. Measuring anonymization privacy/analysis tradeoffs inherent to sharing network data. In NOMS 2008-2008 IEEE Network Operations and Management Symposium, pp. 991-994. IEEE, 2008.

[44] Gattani, Shantanu, and Thomas E. Daniels. Reference models for network data anonymization. In Proceedings of the 1st ACM workshop on Network data anonymization, pp. 41-48. ACM, 2008.

[45] Jorgensen, Zach, Ting Yu, and Graham Cormode. Conservative or liberal? Personalized differential privacy. In 2015 IEEE 31St international conference on data engineering, pp. 1023-1034. IEEE, 2015.

[46] Ninghui Li, Wahbeh Qardaji, and Dong Su. 2012. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12). ACM, New York, NY, USA, 32-33.

[47] Li, Ninghui & Qardaji, Wahbeh & Su, Dong. (2011). Provably Private Data Anonymization: Or, k-Anonymity Meets Differential Privacy. CoRR. abs/1101.2604.

[48] Ruoming Pang and Vern Paxson. 2003. A high-level programming environment for packet trace anonymization and transformation. In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03). ACM, New York, NY, USA, 339-351. DOI: https://doi.org/10.1145/863955.863994

[49] Gehrke, Johannes, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In Annual Cryptology Conference, pp. 479-496. Springer, Berlin, Heidelberg, 2012.

[50] Bild, Raffael, Klaus A. Kuhn, and Fabian Prasser. Safepub: A truthful data anonymization algorithm with strong privacy guarantees. Proceedings on Privacy Enhancing Technologies 2018, no. 1 (2018): 67-87.

[51] Liyue Fan and Hongxia Jin. 2015. A Practical Framework for Privacy-Preserving Data Analytics. In Proceedings of the 24th International Conference on World Wide Web (WWW '15). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 311-321. DOI: https://doi.org/10.1145/2736277.2741122

[52] Zhang, Jianqing, Nikita Borisov, and William Yurcik. Outsourcing security analysis with anonymized logs. In Securecomm and Workshops, 2006, pp. 1-9. IEEE, 2006.

[53] Saroiu, Stefan, P. Krishna Gummadi, and Steven D. Gribble. Measurement study of peer-to-peer file sharing systems. In Electronic Imaging 2002, pp. 156-170. International Society for Optics and Photonics, 2001.

[54] Chor, Benny, et al. Private information retrieval. Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on. IEEE, 1995.

[55] Biler, Piotr, and Alfred Witkowski. Problems in mathematical analysis. (1990).

[56] Zhang, Qianli, and Xing Li. An IP address anonymization scheme with multiple access levels. In International Conference on Information Networking, pp. 793-802. Springer, Berlin, Heidelberg, 2006.

[57] Ribeiro, Bruno F., Weifeng Chen, Gerome Miklau, and Donald F. Towsley. Analyzing Privacy in Enterprise Packet Trace Anonymization. In NDSS. 2008.

[58] eorgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. 2016. Generic Attacks on Secure Outsourced Databases. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 1329-1340. DOI: https://doi.org/10.1145/2976749.2978386.

[59] F. Betül Durak, Thomas M. DuBuisson, and David Cash. 2016. What Else is Revealed by Order-Revealing Encryption?. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 1155-1166. DOI: https://doi.org/10.1145/2976749.2978379.

[60] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. 2018. The tao of inference in privacy-protected databases. Proc. VLDB Endow. 11, 11 (July 2018), 1715-1728. DOI: https://doi.org/10.14778/3236187.3236217.

[61] Paul Grubbs, Marie-Sarah Lacharite, Brice Minaud, Kenneth G. Paterson. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. IEEE Symposium on Security and Privacy (S&P) 2019, May 2019, San Francisco, United States.

[62] Coull, Scott E., Charles V. Wright, Fabian Monrose, Michael P. Collins, and Michael K. Reiter. Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces. In NDSS, vol. 7, pp. 35-47. 2007.

[63] Yurcik, William, and Yifan Li. Internet security visualization case study: Instrumenting a network for NetFlow security visualization tools. In 21st Annual Computer Security Applications Conference (ACSAC). 2005.

[64] Coull, S. E., Monrose, F., Reiter, M. K., & Bailey, M. (2009, March). The challenges of effectively anonymizing network data. In Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology (pp. 230-236).

IEEE.

[65] Del Piccolo, Valentin, et al. A Survey of network isolation solutions for multi-tenant data centers. IEEE Communications Surveys & Tutorials 18.4 (2016): 2787-2821.

[66] Dwork, Cynthia. Differential privacy. Encyclopedia of Cryptography and Security (2011): 338-340.

[67] Dwork, Cynthia. Differential privacy: A survey of results. In International Conference on Theory and Applications of Models of Computation, pp. 1-19. Springer, Berlin, Heidelberg, 2008.

[68] Kushilevitz, Eyal, and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on, pp. 364-373. IEEE, 1997.

[69] Oded Goldreich and Rafail Ostrovsky. 1996. Software protection and simulation on oblivious RAMs. J. ACM 43, 3 (May 1996), 431-473. DOI=http://dx.doi.org/10.1145/233551.233553

[70] Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference, pp. 265-284. Springer, Berlin, Heidelberg, 2006.

[71] Yao, Andrew Chi-Chih. How to generate and exchange secrets. In Foundations of Computer Science, 1986., 27th Annual Symposium on, pp. 162-167. IEEE, 1986.

[72] Goldreich, Oded. Secure multi-party computation. Manuscript. Preliminary version (1998): 86-97.

[73] Cormen, Thomas H., et al. Data structures for disjoint sets. Introduction to Algorithms (2001): 498-524

[74] Sedgewick, Robert. Implementing quicksort programs. Communications of the ACM 21.10 (1978): 847-857.

[75] Slagell, Adam, Jun Wang, and William Yurcik. Network log anonymization: Application of crypto-pan to cisco netflows. Proceedings of the Workshop on Secure Knowledge Management 2004. 2004.

[76] Minshall G. TCPdpriv command manual. 1996. http://ita. ee. lbl. gov/html/contrib/tcpdpriv. 0. txt. 1996.

[77] Paul, Ruma R., Victor C. Valgenti, and Min Sik Kim. Real-time Netshuffle: Graph distortion for on-line anonymization. In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pp. 133-134. IEEE, 2011.

[78] Aggarwal, Gagan, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 153-162. ACM, 2006.

[79] Burkhart, Martin, Strasser, Mario , Many, Dilip, and Dimitropoulos, Xenofontas. (2010). SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. Proceedings of USENIX Security Symposium. 1.

[80] Boldyreva, Alexandra, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 224-241. Springer, Berlin, Heidelberg, 2009.

[81] Curtmola, Reza, Juan Garay, Seny Kamara, and Rafail Ostrovsky. "Searchable symmetric encryption: improved definitions and efficient constructions." Journal of Computer Security 19, no. 5 (2011): 895-934.

[82] Song, Dawn Xiaoding, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, pp. 44-55. IEEE, 2000.

[83] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC '09). ACM, New York, NY, USA, 169-178.

[84] Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." In Theory of Cryptography Conference, pp. 253-273. Springer, Berlin, Heidelberg, 2011.

[85] Bellare, Mihir, Alexandra Boldyreva, and Adam O'Neill. "Deterministic and efficiently searchable encryption." In Annual International Cryptology Conference, pp. 535-552. Springer, Berlin, Heidelberg, 2007.

[86] Boldyreva, Alexandra, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 224-241. Springer, Berlin, Heidelberg, 2009.

[87] Islam, Mohammad Saiful, Mehmet Kuzu, and Murat Kantarcioglu. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In Ndss, vol. 20, p. 12. 2012.

[88] Naveed, Muhammad, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 644-655. ACM, 2015.

[89] Chang, Zhao, Dong Xie, and Feifei Li. Oblivious ram: a dissection and experimental evaluation. Proceedings of the VLDB Endowment 9, no. 12 (2016): 1113-1124.

[90] Caswell, Brian, and Jay Beale. Snort 2.1 intrusion detection. Elsevier, 2004.

[91] Stefanov, E., Van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X. and Devadas, S., 2013, November. Path ORAM: an extremely simple oblivious RAM protocol. In Proceedings of the 20th CCS, pp. 299-310. ACM, 2013.

[92] Justin King, Kiran Lakkaraju, and Adam Slagell. 2009. A taxonomy and adversarial model for attacks against network log anonymization. In Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09). ACM, New York, NY, USA, 1286-1293. DOI: https://doi.org/10.1145/1529282.1529572