Privacy-Preserving Network Flow Recording

Bilal Shebaro and Jedidiah R. Crandall
University of New Mexico, Dept. of Computer Science
{bshebaro,crandall}@cs.unm.edu

Abstract

Network flow recording is an important tool with applications that range from legal compliance and security auditing to network forensics, troubleshooting, and marketing. Unfortunately, current network flow recording technologies do not allow network operators to enforce a privacy policy on the data that is recorded, in particular how this data is stored and used within the organization. Challenges to building such a technology include the public key infrastructure, scalability, and gathering statistics about the data while still preserving privacy.

We present a network flow recording technology that addresses these challenges by using Identity Based Encryption in combination with privacy-preserving semantics for on-the-fly statistics. We argue that our implementation supports a wide range of policies that cover many current applications of network flow recording. We also characterize the performance and scalability of our implementation and find that the encryption and statistics scale well and can easily keep up with the rate at which commodity systems can capture traffic, with a couple of interesting caveats about the size of the subnet that data is being recorded for and how statistics generation is affected by implementation details. We conclude that privacy-preserving network flow recording is possible at 10 gigabit rates for subnets as large as a `/20` (4096 hosts).

Because network flow recording is one of the most serious threats to web privacy today, we believe that developing technology to enforce a privacy policy on the recorded data is an important first step before policy makers can make decisions about how network operators can and should store and use network flow data. Our goal in this paper is to explore the tradeoffs of performance and scalability *vs.* privacy, and the usefulness of the recorded data in forensics *vs.* privacy.

1 Introduction

Network flow recording, such as Cisco's NetFlow [7], is a tool that network administrators use to record information about network traffic sessions. This data can be used for network forensics and traffic accounting, usage-based network billing, network planning, security, denial-of-service monitoring, and network monitoring. Network flows also provide valuable information about network users and applications, peak usage times, and traffic routing. Along with the related practice of clickstream recording, network flow recording is one of the most serious threats to Internet and web privacy today. Network flow data reveals what websites a network user visited, along with timing information, what services they used, and how much data was transferred. An especially important issue is how this data is stored, since a compromise of this data reveals so much about so many users. Another important issue is how the data can be accessed for the purposes it was collected for without enabling accesses that violate the privacy policy. In nowadays forensics, a digital crime occuring in a network will cause the whole network traffic to be analyzed and investigated on, thus time consuming for invesigators as well as invading the privacy of users that are on the network and not involved in the crime.

Most ISPs use or implement network analyzer tools that read network flow data and do their own traffic analysis, such as Orion NetFlow Traffic Analyzer (NTA) which enables network administrators to capture data from continuous streams of network traffic and convert those raw numbers into easy-to-interpret charts and tables that quantify exactly how the corporate network is being used, by whom, and for what purpose [21]. Another tool is NetFlow Analyzer, that uses Cisco NetFlow to monitor bandwidth and to gather information on the top network users, applications, and many other features [16]. These tools produce reports on in-depth traffic analysis and network bandwidth monitoring. Network flow data can also be requested by law enforcement agencies to aid in investigations. Because network flow data is used for so many critical functions, it is important to provide a way for network operators to record and use this data while also enforcing a privacy policy on the data.

While ISPs care about the privacy of their customers as well as providing the most secure and convenient service, in the U.S. there have been many calls from the FBI and multiple members of congress requesting that ISPs perform mandatory data retention, that is, keeping records of their users' activities for later review by law enforcement [9]. This can aid in the investigations of serious crimes such as hosting child pornography. At the same time, the potential for data breaches and abuses by employees of the network operator means that network operators should enforce privacy policies on the data collected, and such enforcement may become mandatory in the future. This paper is about how to develop technologies that aid in enforcing such privacy policies.

The network flow recording system that we have developed, and describe in this paper, can provide ISPs with common traffic analysis and statistics as well as the ability to retain more detailed information about each session, while encrypting the session data and enforcing privacy-preserving semantics on queries for the statistics. We consider two different threats: threats against the stored session data, and abuses of the statistics. To address the first, we propose to use Identity Based Encryption (IBE) because it limits the public key infrastructure necessary such that the network users do not need to be aware of their public keys nor modify their computers or network traffic in any way. To address the second threat, we define a privacy-preserving semantics for network flow data that provides access to statistical reports about groups of individuals, while restricting access to information about any particular individual unless enough data is aggregated to mitigate any privacy threats.

This paper is organized as follows. First, we describe our threat model in Section 2. This is followed by Section 3 where we introduce some basic concepts and background. Section 4 shows the implementation of our system. In Section 5 we explain our evaluation methodology, and then our results in Section 6. A discussion of performance issues and future work in Section 7 is then followed by related works and the conclusion.

2 Threat Model

First, we must distinguish between forward security and backward security as it applies to network flow recording privacy. Forward security means protecting information that will be recorded in the future, while backward security means that after a compromise the data that was recorded in the past is still secure. Forward security is not possible in the context of network flow recording. At the moment that a gateway router is compromised or an employee of the network operator with the necessary access decides to violate the policy, they can simply record network flow data as plaintext. It is impossible for web users to hide their source and destination IP addresses and timing information from those that they trust to route their packets.

Backward security, on the other hand, is very important for network flow data, and is the focus of this paper. Network operators reported that they retain network flows and clickstreams for anywhere from 2 weeks to 6 months and even, in some cases, indefinitely [2]. It is this stored data that must be protected, so that exposure of private information is limited to the time that a breach of policy or a router compromise persists undetected, and no longer. So, the first of two threats that we address in this paper is how to encrypt this data in a way that enforces the privacy policy but requires no complicated public key infrastructure or modifications to protocols or implementations outside of the network flow recording system. Using Identity Based Encryption gives us three important properties in this regard. First, the public key (or encryption key) for each individual IP address can be the IP address itself, possibly combined with a timestamp. Thus the network users do not need to encode their public keys into the network traffic, nor even be aware that any public key cryptography is involved. The network hosts and network itself operate as they normally would. Second, the network flow recorder can encrypt the session data for an IP address in a given time period using the IP address and timestamp for that period as the encryption key, and does not need to make any queries to a public key infrastructure to know the required public key. Third, the secret that decrypts the traffic does not need to be stored where the network flow data is recorded and encrypted, and this secret can be divided into pieces so that separation of duty requires multiple parties to be involved to decrypt the data.

The second threat that we address in this paper has to do with the fact that not all of the information can be encrypted in most applications of network flow recording, since statistics about how much and when network links are used are also very important just for day-to-day network operations. The threat is that information about the amount of network traffic an individual used must be stored for statistical purposes for the queries to be accurate (for example, for billing purposes), but this information is often enough to infer certain facts about an individual user's web activities if it is not aggregated properly with other data before being presented. We assume that the stored statistical data is presented to the rest of the organization (the billing department, traffic engineers, quality-of-service experts, *etc.*) through a database interface, and present a privacy-preserving semantics for that database interface that enables a wide range of queries but preserves the privacy of individuals.

As one example of an organization that might want to enforce a privacy policy on network flow data, consider a university. In the U.S., universities collect network flow data for network management reasons but are often also required to retain network flow data for periods of weeks due to state laws about employing state employees or federal laws designed to curb the pirating of copyrighted music and videos. This period of weeks or months is a very long exposure window in which any breach of that data or possible violation of the privacy policy by a university employee is a major vulnerability in terms of the web privacy of the network's users. Our proposed system would allow for the network flow recording to support both the legal obligations and network operations of the university, while virtually eliminating this window of exposure. The secret necessary to decrypt traffic for law enforcement or public freedom of information purposes could be divided, for example, among the regents, faculty senate, and university counsel so that the enforcement of the university's privacy policy for this data has the property of separation of duty.

Also consider an Internet Service Provider (ISP). In this case the secret used for decryption could be divided between the customer service department and another department that is tasked with enforcing and auditing the privacy policy of the organization. If a customer service agent needed to look at detailed network flow data for a particular customer to debug a network connection problem, they could obtain the customer's consent and then request the keys to decrypt that customer's network traffic 24 hours into the past and several hours into the future. The department that enforces and audits the privacy policy could confirm customer consent and then use their part of the decryption secret to provide keys for only the requested time period, and these keys could not be used to decrypt traffic for other customers or in other periods of time. More routine queries of data (how much bandwidth is a particular neighborhood using during peak hours, what are the most common applications being used by a large group of customers, how much did a particular customer go over their allowed traffic quota in a billing period, and so on) can be handled by the privacy-preserving statistical database.

We wish to reiterate that we are not attempting to protect web privacy against untrusted network controllers. Our goal is to give network controllers the technological tools they need to enforce privacy policies so that network users can place their trust in the network controllers as an organization.

3 Background

Our system uses the concepts of traffic flow recording, data collection, Identity Based Encryption, and AES as well as statistical database modeling and inference controls. These concepts are key to our our implementation so we describe some background about them here.

3.1 Traffic flow recording and data collection

The most common form of network flow recording is Cisco's NetFlow, which is a format and tool designed for network administrators that provides a set of services for IP applications [7]. It can capture all traffic that passes through a tapped network interface or router or sample that traffic at some specified rate, and there are tools for combining network flow data from multiples routes. In our implementation, a tool called fprobe is used to record network traffic, which is a libpcap-based tool that records network traffic data in real-time and emits it as session information that can then be turned into network flow records with a tool such as nfcapd. Each network flow record contains the source and destination IP addresses of the recorded session, with the total number of bytes that were transfered, as well as the protocol type, mainly TCP or UDP, with a timestamp and duration for the whole session. Each of these records is collected after the session is closed and these fields will be known and ready to be printed as a whole record line.

3.2 Identity Based Encryption and AES

An Identity Base Encryption (IBE) scheme is a public-key cryptosystem where the public key can be any string. Such a scheme was first proposed by Shamir in 1984 [20] but it was not until 2001 that Boneh and Franklin [5] gave a practical way to do identity-based encryption based on the Weil pairing. In IBE the strings for public keys can be email addresses, phone numbers, IP addresses, or any identifier, and can be merged with dates or represented hierarchically so that notions of time and hierarchical trust relationships can be included in the public key. The cryptosystem we used has chosen-ciphertext security in the random oracle model, assuming an elliptic curve variant of the computational Diffie-Hellman problem.

IBE has a lot of interesting features. One is that it allows encryption without the need to look up the public key or a certificate for the identity associated with the encryption key. Also, individual private keys within the scheme need not be generated or stored anywhere until decryption is necessary. If an encryption key is a public key that is a string merged with a timestamp, any attacker who discovers the private key associated with that timestamp will only be able to decrypt the messages for that particular time. Ideally IBE is used in systems where encryption is applied more often than decryption. One last property of IBE that we use in this paper is that the secret needed to generate private keys associated with identities can be broken up among more than one party for separation of duty. This prevents abuse and can ensure that only trusted authorities combined can generate the private key that decrypts the suspect's records. For example, law enforcement agencies are only given data for the user and time period a warrant specifies, requiring the cooperation of multiple parties within an organization to obtain the necessary keys.

Advanced Encryption Standard (AES) is a symmetric-key encryption standard that comprises several sizes of block ciphers: AES-128, AES-192, and AES-256 [23]. The AES ciphers have been analyzed extensively and are now used worldwide. We use AES to encrypt network flow records for its speed, and then apply IBE to the stored AES key for each individual IP address and time period.

3.3 Statistical database modeling and inference controls

Statistics are computed for subgroups of records having common attributes [18]. A characteristic formula is any logical formula over the values of attributes where the set of records that matches these values is called a query set. A statistic is considered sensitive if it discloses too much confidential information about some individual, and with the use of some criteria for each query we prevent this type of query from returning an answer.

Inference control mechanisms are those which protect all sensitive statistics. The problem is that it can be extremely difficult to determine whether releasing a statistic will lead to disclosure of sensitive statistics information or not, but through the concept of query set overlap control we are able to keep our statistical reports from leading to the inference of any individual's sensitive data.

4 Implementation

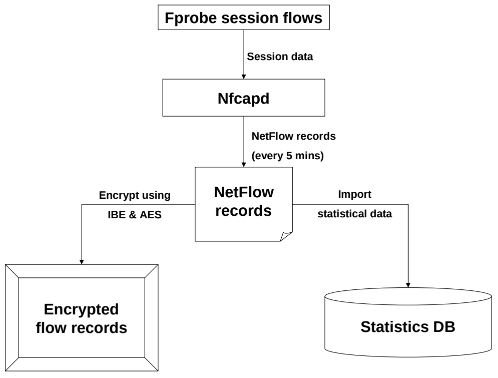

Figure 1: **Data collection steps, and examples of plaintext and encrypted network flow records.**

Figure 1 shows the steps flow records go through in our implementation. Our system is implemented in two main phases. Phase one addresses threats against stored, complete network flow records by encrypting them. Phase two is focused on data collection for producing statistical reports that are privacy-preserving based on specific criteria on statistical queries. This second phase addresses threats where these statistics, possibly combined, can reveal information about individual network users.

4.1 Data collection

Starting with data collection, fprobe 1.1 is used to record the network traffic that passes though our tapped network interface as sessions. Then nfcapd is used to collect the recorded data in a series of rotated files in Cisco's NetFlow format, each file containing 5 minutes of recorded data. After the completion of each of these files, the file is read separately and required data for statistics is imported from it and then stored into the statistical database. This database stores statistical network data that is needed for queries for statistical reports. Immediately after this, we encrypt the file. Thus, complete network flow data records will not be stored in plaintext for more than five minutes. Each encrypted file contains only the network flow records for a single IP address in a single 5-minute period.

4.2 Phase 1: Encryption vs. decryption

We encrypt using a combination of identity based encryption (IBE) and AES encryption. Our IBE implementation is custom written, using the pairing based crypto library [22] as a reference. Since our privacy guarantee is per-user, we identify in each record the IP address that belongs to our domain (whether it is the source or destination) and merge it with the timestamp associated with this file that includes records for that IP address in that 5-minute period. All flow records for a particular IP address in a 5-minute period are merged into one file and encrypted with AES using a random 128-bit key that is different for each IP-timestamp tuple. After this, each of these AES keys is encrypted with IBE using the IP-timestamp tuple as the public key for this encryption. All plaintext files are immediately deleted, and the end result is a log file where each five minute period has a list of IP addresses with records for that time period. Each record contains the IP address and an AES decryption key that is encrypted with IBE. This information or the information in the statistical database could violate the criteria of our queries for generating statistical reports, but this is a separate threat and we have two separate threat models for the two separate threats. See Section 7 for more discussion about this.

When law enforcement requests the decrypted records for a particular IP address within a particular time period, the parties that have pieces of the secret needed for decryption must each aid in generating the decrypting IBE keys. If they all agree and cooperate, they can provide law enforcement with a set of keys that only decrypts the set of 5-minute periods specified and only for the IP address given. This set of decryption keys can be used to obtain the stored AES key for each of the relevant records.

4.3 Phase 2: Generating statistical reports

Statistical reports are generated after instantiating certain queries that comply with specific criteria designed to protect the privacy of individual network users. These criteria ensure that the queries not only prevent the direct revelation of information about individual users, but also prevent inferences of such information using any combination of possible queries. At the stage where the timestamped file of plaintext records is complete, before it is encrypted and the plaintext version deleted, some values needed for statistical reports are taken from this file and stored in a database that will be used to generate the statistical reports. This database does contain some individual information but cannot be accessed except through an interface with our predefined queries that are designed in order not to reveal any private information of a particular user, not even through inference. Generating statistical reports allows ISPs to monitor their network performance and the bandwidth of users, generate bills, and many other critical functions, so the major tradeoff is accuracy of aggregated statistics *vs.* privacy of individual statistics.

Before we discuss the queries and how privacy-preserving reports are generated, we need to explain the format of the database that such queries can be executed on. As explained before, every five minutes a new file of network flow records is completed. We take the statistical values we need from this file for statistics, then encrypt each record in the file as explained before, and then the original plaintext file is deleted. The statistical values are stored in a database as one record per IP address per time period (TP). Note the difference between the recording time interval and the time period TP; a recording time interval is the five minute period in which network flow records are recorded in plaintext and then encrypted in a rotating fashion, a time period TP is a 12-hour period chosen such that the statistical reports generated will consists of at least 12-hours of statistical data records.

Another problem that might lead to the inference of an individual user's private information (such as what websites they visit) is if, for example, they only visited one website in one TP. The number of bytes transferred in this 12-hour time period can be used as a signature to make inferences about what website was visited. For this reason, one of our criteria is that there should be enough bytes transfered by an IP address in a given query so that very little can be inferred about specific websites visited. In this case, however, it is not enough to define some queries as legal and others as not legal. Two legal queries can form two datasets whose intersection leads to some confidential data about an individual. To solve this, we check the query criteria on the records for every TP and decide which IPs do not match the criteria within that TP. For IP records, we merge them with the corresponding IP record in either the pre- or post TP according to where that record was originally found. After merging these records, they are given a timestamp of 24 hours instead of 12 hours if they were merged with one TP, or 36 hours if merged with two TPs, and so on.

Our statistical database is populated immediately after the network flow file is completed, new values in this database are entered or merged every five minutes in the following form:

- IP: IP Address
- TP: Time Period (time-stamped)
- TTI: Total TCP bytes In
- TTO: Total TCP bytes Out
- TUI: Total UDP bytes In
- TUO: Total UDP bytes Out
- LPI: List of Ports In
- LPO: List of Ports Out
- BI: Bytes In
- BO: Bytes Out
- PI: Packets In
- PO: Packets Out

The values of these attributes for a single record are updated every five minutes for every TP, per IP address, then a new record for the same IP address is created for the next TP and so on.

Generating the privacy-preserving statistical reports requires us to design a set of queries with corresponding criteria for each to generate the required report that ISPs need, as well as not revealing any confidential data about individual users. Below is a list of some of the queries with their corresponding criteria:

$$Q1 : Sum[BI, (TP \geq \alpha) \bullet IP] \;\; \& \; result \geq \beta$$

$$Q2 : Sum[BO, (TP \geq \alpha) \bullet IP] \;\; \& \; result \geq \beta$$

$$Q3 : Sum[BI + BO, (TP \geq \alpha) \bullet IP] \;\; \& \; result \geq \beta$$

Q1, Q2, and Q3 are queries used to calculate the link utilization of a particular IP in a specified time interval for bytes-in, bytes-out, and total bytes respectively. The criteria and conditions for such queries is that $TP \geq \alpha$ and $result \geq \beta$ are satisfied. The values of α and β should be defined by the ISP depending on how large and busy the network is. Basically, the conditions for these queries state that if the number of TPs required is greater than α, a predefined number of 12-hour period statistical reports, the answer for such a query should also be above a certain threshold β. Otherwise confidential data of that particular IP could be inferred. In other words, if that particular IP was not active enough during these TPs, then the results output from this query could make it possible to deduce something about the websites or Internet services being used by a specific user. For example, Hintz discusses the concept of website fingerprinting [13] where it is possible to infer that a website has been visited through a signature based on the number of bytes transferred. So the value of β in the above queries should be chosen to be large enough such that no such patterns exist. One key feature about these conditions in the above three queries is that they must have the same conditional values (*i.e.*, the

α values of the three queries must be equal, as must the β values), so that their combination cannot dissatisfy the conditions of any other statistical query.

$$Q4 : 8 \times \frac{\sum_{i}^{n}[BI+BO,(TP \geq \alpha) \bullet IP_i}{TP_{\text{sec}}}$$

$$\forall IP_i \in subnet,\ count(IP_{i_s}) > \delta\ \ \&\ result \geq \beta$$

Q4 is another example of a privacy-preserving statistical query that calculates the total bytes per time per subnet. Such queries require all the records of the IP addresses that fall in the given subnet in the given number of time periods such that $TP \geq \alpha$ and satisfying the same conditions that were specified in Q1, Q2, and Q3. This is important for the same reason of avoiding any combination of queries to infer individual queries that would violate their conditions.

$$Q5 : list[LPI, (TP \geq \alpha) \bullet IP_i] \\ + \quad list[LPO, (TP \geq \alpha) \bullet IP_i]$$

$$\forall IP_i \in subnet,\ count(IP_{i_s}) > \delta$$

One very useful query for ISPs is the list of applications used over a time period for particular subnetworks. For simplicity, we define applications in terms of ports for our queries. Q5 is designed so that queries can return aggregate statistical data about ports used for large-enough groups of network users, but cannot query information pertaining to any single user. This query requires that enough data be included in the aggregation, both in terms of the size of the subnet and the amount of data available for that subnet.

Our set of queries was designed to demonstrate that useful statistical data can be extracted from network flow records while still preserving privacy. We have focused on basic network administration and billing tasks. Additional queries could be defined for other purposes, the major challenge being the tradeoff between privacy and accuracy of the report.

5 Experimental Methodology

Here we explain our experimental setup and methodology. **The purpose of our experiments was to characterize the way that the traffic recording, identity-based encryption, and statistics generation scaled with respect to each other. In particular, we are interested in which of these might present problems as either the rate of traffic that is recorded goes up or the size of the subnet that recording is being performed for goes up.** Because each IP address is an identity, performance scaling depends not only on the rate of flow records generated but also on the number of local IP addresses involved since the identity based encryption workload is per IP address.

5.1 Live experiments

One specific question we wanted to answer is: for a reasonably sized subnet, are our encryption and statistics-generating implementations fast enough to keep up with the rate at which network flow records can be generated on a typical commodity machine. We wanted to test this on a live system that was performing all of the packet capture and processing that a real system with our privacy-preserving capabilities would perform, so that system effects would also be accounted for. To answer this question, we ran live capture experiments for 6 hours each, on subnets of size `/24` (256 hosts), `/22` (1024 hosts), and `/20` (4096 hosts). The machine we ran these experiments on was a Core i7 X980 running at 3.33 GHz, with 24 gigabytes of RAM and a RAID 0 array with three 6 GB/s hard drives dedicated to the partition that we used. However, the motherboard RAID controller, though it uses PCI Express, limits the hard drive bandwidth to 3 gigabits per second so that this is the fastest rate we were able to test for long periods of time, due to the need for a large file so that tcpreplay does not loop too often over the same traffic. The Core i7 X980 has six cores, each with two hardware threads, and supports Advanced Encryption Standard Instructions (AES-NI) on all six cores. For both encryption and statistics generation we used six parallel threads (twelve threads would not have had a significant speedup over six since then pairs of threads would be sharing the AES hardware and memory hierarchies). The test machine was running Linux kernel version 2.6.32. Our live experiments were performed at 3 gigabits per second due to the hard drive bandwidth limitation, but the offline experiments confirm that for the subnet sizes we considered more than 10 gigbaits per second is achieved by our implementation.

We generated a file representing enough realistic traffic to saturate a 3 gigabit link for 1 hour, to be looped six times for a six-hour live test. We then created a tuntap interface, which is a virtual network interface that works in a similar way to a loopback interface, but can be dedicated to just traffic for a particular experiment. Using tcpreplay, we filled this tuntap interface with realistic traffic that was generated based on a statistical profile of real network traffic, in terms of the sizes of flows, the timing, rate, protocols, and other factors. For this purpose, we use AnetTest [3] and modified it to support the subnet sizes we were interested in. We used other tools as references [17] in order to generate more realistic traffic rather than just depending on one traffic generator's statistical model. Our goal was to consider the worst-case scenario in terms of the amount of sessions that could result from real traffic at a given rate.

By looping this 1-hour, 2 terabyte file six times we were able to do live testing of our privacy-preserving network flow recording implementation for six hours for each of three subnet sizes: /24, /22, and /20. The purpose of these tests was to demonstrate that our implementation could sustain a rate of 3 gigabits per second for network flow recording for a long period of time. The purpose of rerunning or looping this 1-hour files six times is for the purpose of eliminating any interference caused by the system and being more accurate in our results. We also wanted to see how well the different parts (encryption and statistics generation) scaled in terms of the amount of data collected in a 5-minute time period.

5.2 Offline experiments

We also sought to answer the question of how fast our implementation could perform for different subnet sizes, without hardware limitations or the inherent limitations of the Linux kernel in capturing network traffic. The reason this is important is that improvements in network traffic recording could lead to network flow recording at 10 gigabit rates, so the encryption and statistics generation for a privacy-preserving network flow implementation would have to keep up with this rate. To answer this question we generated libpcap files of traffic as before and then created the network flow records offline. Then we measured the time that encryption and statistics generation took when computation was the only limiting factor (since network flow records are small and not a significant factor in file system bandwidth). From this we inferred rates that our implementation could perform privacy-preserving network flow recording at for the same three subnet sizes: /24, /22, and /20.

6 Results

Recall that for live experiments our main purpose was to demonstrate that privacy-preserving network flow recording could be performed at the 3 gigabit per second rate on a commodity Linux machine. We answered this in the affirmative, but there are a couple of interesting caveats about the scalability of our system that we describe below.

Also recall that we ran off-line experiments to see if our implementation could achieve a 10 gigabit per second rate, assuming that future improvements to the Linux packet capture infrastructure make it possible to record traffic at this rate. We also answered this in the affirmative.

6.1 Live experiments

Figure 2 shows how encryption and statistics scale compared to the time to record on a /24 subnet. The straight line that ends at 300 seconds is the amount of time that it takes to record the traffic, with 500,000 flow records being recorded in 300 seconds (5 minutes). This time is the same with or without performing any privacy-

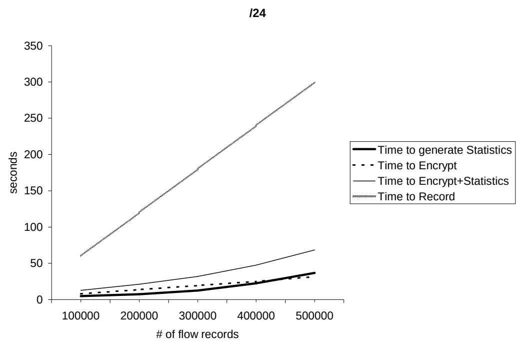

Figure 2: **Timings for /24 Subnets.**

preserving operations such as encryption or statistics generation. Note that our implementation of these operations can be done in parallel to the recording of network flow records, so the times will not be added on but just need to be less than the time to record. The x-axis is the number of flow records recorded up to that point in time. For a fixed bandwidth and subnet size, the worst-case scenario for a network flow recording implementation (that is not sampling) is a large amount of very short sessions. 500,000 network flow records in a five minute period is what we considered to be a worst-case scenario for our experiments. Recall that this is a mix of large and small connections but is more biased towards small connections than we would expect normal traffic to be, to ensure that our tests cover all realistic scenarios by considering the worst case. The y-axis is either the time that has passed for recording the traffic, or the amount of time (out of that total time passed) that has been spent so far on encryption, statistics generation, or both.

The dashed line is the amount of time spent on encryption, including both identity-based encryption and AES. The thick black line is the amount of time spent generating statistics, and the thinner black line is the total time spent on privacy-preserving operations (*i.e.*, the sum of the times for encryption and statistics). Note that this time is cumulative, so linear growth means that the performance of that component is the same throughout. In this graph, it is clear that the privacy-preserving operations can be performed in the amount of time required, meaning that adding privacy-preserving operations to a network flow implementation will not change the rate at which it can perform. For a /24 subnet at 3 gigabits per second, the encryption and statistics only took a fraction of the time allowed for them.

Figure 3 is the same as Figure 2, except that the scale on the y-axis is different to show more details about how encryption and statistics generation scale with respect to

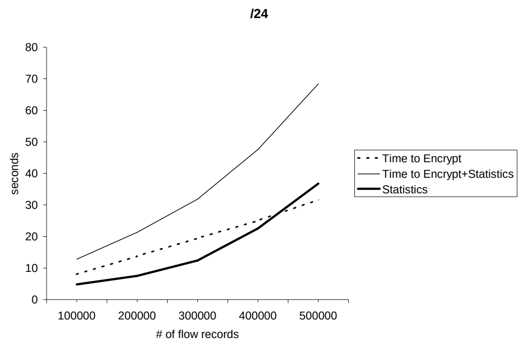

Figure 3: **Tradeoff between time to encrypt and time to import statistical data for** `/24` **subnets.**

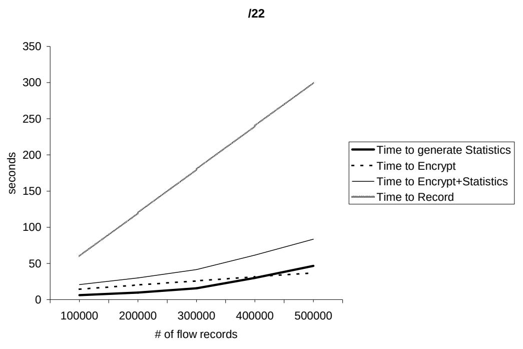

Figure 4: **Timings for** `/22` **subnets.**

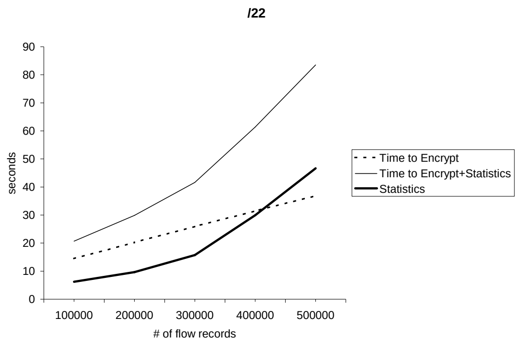

Figure 5: **Tradeoff between time to encrypt and time to import statistical data for** `/22` **subnets.**

each other. We expected that encryption would scale non-linearly with subnet size because of identity-based encryption being required on a per-IP address basis. We were surprised to see that statistics generation grows non-linearly with the number of network flow records, however. The reason for this is that merging statistics records for IP addresses that have already been seen in that five-minute period can be an expensive operation if not implemented carefully. In earlier versions of our implementation, where the dynamically allocated memory for operations such as appending port numbers to a list was not managed explicitly by us because the implementation had been in the Python language, the statistics actually broke the implementation at relatively low rates of traffic, meaning that statistics generation was taking more than five minutes and not keeping up with the network flow recording. We thought that this was due to the size of the hash table for hashing IP addresses that have been seen before in this time period, which does have some effect, but it turned out to be more related to the relatively simple addition and append operations that are performed to merge records for IP addresses that have been seen already in that time period. We found this result, that generating statistics could be more of a performance bottleneck than encryption, to be rather surprising.

Figures 4 and 5, respectively, show the same for `/22` subnets as what Figures 2 and 3 show for /24 subnets. Similarly, Figures 6 and 7 show the same for `/20` subnets. As expected, encryption becomes slightly more dominant in larger subnets since there are more unique IP addresses, as can be seen more clearly in Figure 8. However, we also see that statistics is still the main performance limitation in terms of non-linear behavior with respect to the number of flow records recorded. Since the hash table is reset every five minutes and the time it takes to merge a five-minute statistics record into a 12-hour TP is negligible, we do not need to consider how the statistics scales beyond five minutes.

Our conclusion from the live experiments was that encryption and statistics are scalable and not performance problems compared to how long it takes to record traffic using pcap-based tools. This means that privacy-preserving network flow recording can be done for any realistic rate, traffic mix, and subnet size that regular network flow recording can be done at. However, special attention needs to be paid to the statistics generation, particularly the memory management for this part, to make sure that it scales well over time within a five-minute recording period.

6.2 Offline experiments

Table 1 shows the rates, in gigabits per second, that we were able to achieve in the offline experiments for the three subnet sizes. Clearly, our encryption and statistics generation implementation can be used for 10 gigabit connections at any of these subnet sizes, assuming that the

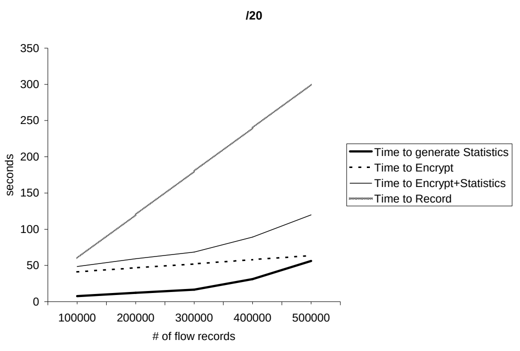

Figure 6: **Timings for `/20` subnets.**

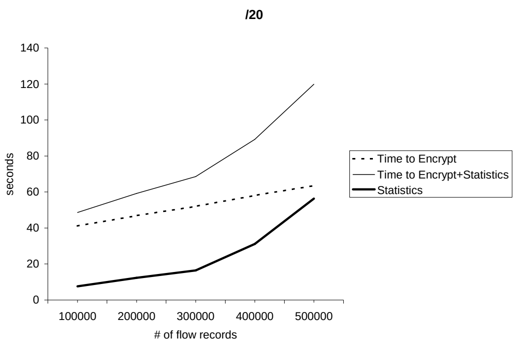

Figure 7: **Tradeoff between time to encrypt and time to import statistical data for `/20` subnets.**

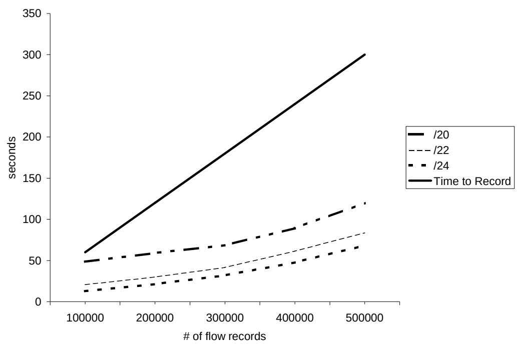

Figure 8: **Timing comparison between `/20`, `/22`, `/24` subnets according to recording time.**

network flow recording itself is able to achieve such a rate.

7 Discussion and future work

We have shown that privacy-preserving network flow recording can be performed at any reasonable rate and subnet size that regular network flow recording can be performed at. However, there are still some open issues for future work. One is to consider other types of queries that network operators may require statistics for. Our existing queries are basic operations, and are all linear. We feel that our current set of queries cover the most common uses of network flow data, however. Another issue that might be considered in future work would be to consider clickstreams and other data that should be kept in a privacy-preserving manner. Since clickstream data is used in different ways compared to network flow data, it may have different requirements for both encryption and statistics.

Subnet size	Maximum rate (Gbps)
`/24`	23
`/22`	18
`/20`	12

Table 1: **Rates that our implementation can achieve for different subnet sizes.**

We considered two threats in this paper: threats against stored network flow data that is stored unencrypted, and threats against statistical data from the network flow recording that is exported to the rest of the organization. We considered these threats separately, and for the second threat we assumed that the statistics were exported only through a database interface. An attacker that can view the encrypted network flow records on the gateway where they are stored cannot violate our protections for the first threat since they are encrypted, but for the second threat many inferences can be made that could not be made through the database interface. The recorded, encrypted network flow data shows every local IP address that communicated in a 5-minute period (IP addresses outside the local network will be encrypted, however). Also, the length of the encrypted records gives some indication of the number of flows during that five-minute period. Since we considered the two threats separately and assumed that for the second threat the attacker would be required to use the database interface, we did not do any padding or attempt to prevent inferences from the encrypted network flow records. This raises an interesting question, however, which is: is it possible to store the encrypted network flow records in a way that prevents the same kinds of statistical inferences if an attacker gains access to the stored network flow records? We plan to explore this in future work.

8 Related Work

Different approaches have been taken in order to protect the confidentiality of web users that are applicable to recorded network flow data, even if network flow recording is not their focus. One main approach that users themselves can employ is to anonymize their traffic through anonymity networks. Tor, the second generation onion router [10], allows you to browse the Internet anonymously. Tor is resistant to many threats against privacy, including many of the threats posed by network flow recording. Tor trades off performance of the network connection for anonymity properties, for the users that choose to use it. In the problem domain we focus on in this paper, we are interested in the threats against the vast majority of users that do not use Tor but whose daily web activities are recorded in the form of network flow records and clickstreams and then stored for weeks, months, or longer.

A research problem that is related to privacy-preserving recording and storage of network flows is the problem of anonymizing network flow data so that it can be shared with others. Foukarakis *et al.* [12] implemented anontool that allows administrators to anonymize network flow data in a highly customizable way. This is more directly related to our statistics generation than to the encryption phase of our implementation. For a discussion of issues concerning external factors in data sanitization, see Bishop *et al.* [4]. Privacy-preserving data aggregation and anonymization have a long history, see, for example, Denning [18]. Related to network flows, there are network trace anonymization attacks that correlate external information with anonymized data and successfully deanonymize objects with distinctive signatures [6]. Moreover, some work has been done to improve Crypto-Pan by designing and integrating it with an efficient passphrase-based key generation algorithm in order to avoid the risk of exposing sensitive information to potential attackers while sharing network logs among security engineers and network operators for the purpose of security research and network measurements [8]. Koukis *et al.* designed and implemented a generic and flexible anonymization framework, due to the current anonymized tools that offer limited flexibility, thus their tool provides extended functionality, covering multiple aspects of anonymization needs and allowing fine-tuning of privacy protection level [15].

We've also seen research related to privacy-preserving forensic attribution architecture primitive where the authors propose a prototype implementation , called Clue, that attributes individual network packets of the sender's machine to protect the privacy of individual senders from serendipitous use, with the ability of revealing the identity of the criminal actors by a trusted authority [1].

Securing log files by cryptographic means was another considered approach towards hiding sensitive information that is stored in certain log files. Schneier and Kelsey [19] had described a computationally cheap method that makes it impossible for the attacker to read, modify, or destroy log file entries by generating log entries prior to the logging compromised machine.

Much previous work has focused on privacy-preserving data publishing. Denning [18] provides a good overview of early work in this area. More recently the notion of differential privacy was introduced by Dwork [11]. The focus is on privacy-preserving analysis of data, where the concept of differential privacy is used to capture the increased risk to one's privacy incurred by participating in a database. Differential privacy is a useful property for many types of proofs and reasoning about privacy. Our work could be more formally analyzed within the framework of differential privacy, but for our current effort we have defined our queries in a way that more closely resembles the inference controls of Denning [18]. Zhou *et al.* [24] target the problem of continuous privacy-preserving publishing of data streams, as opposed to static data. They propose a group of randomized methods and anonymization quality measures used to verify the effectiveness and efficiency of their methods. Our statistical reports could be cast as a streaming privacy-preserving data problem. Horizontally partitioned databases also fall into the category of privacy-preserving data publishing, as studied by Jurczyk and Xiong [14]. They developed a set of decentralized protocols that enable data sharing for horizontally partitioned databases.

To the best of our knowledge, ours is the first work to focus on one of the largest threats to web privacy today: the recording and storage of network flows and the related practice of clickstreams.

9 Conclusion

In this paper, we proposed a privacy-preserving method for recording and storing network flow records, that network operators can use to enforce a privacy policy. A performance analysis of our implementation demonstrates that privacy-preserving network flow recording can be performed at any realistic rate, traffic mix, and subnet size that regular network flow recording can be performed at. We also showed that our implementation can achieve more than 10 gigabits a second on reasonably-sized subnets, as large as `/20`. The main tradeoffs and performance considerations that we identified were:

- The tradeoff between accuracy and privacy for defining privacy-preserving database semantics that allow network operators to view statistics about their network.
- The fact that identity-based encryption in our implementation scales with subnet size and not just the amount of traffic.
- The fact that the memory management of statistics

generation scales non-linearly as the number of IP addresses and records grows.

10 Acknowledgments

This work was supported in part by the U.S. National Science Foundation (CNS-0905177, CNS-0844880). Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.". We would also like to thank the DFRWS anonymous reviewers and others who gave valuable input, like Flocon 2011 attendees. Monzy Merza's conversations with us and help with early coding and testing was very valuable, and we are very grateful for it.

References

[1] M. Afanasyev, T. Kohno, J. Ma, N. Murphy, S. Savage, A. C. Snoeren, and G. M. Voelker. Network support for privacy-preserving forensic attribution. Technical Report CS2009-0940, University of California - San Diego and University of Washington, March 2009.

[2] J. Angwin and T. McGinty. Sites Feed Personal Details To New Tracking Industry. Wall Stree Journal, available at `http://online.wsj.com/`.

[3] Anton Titov, SourceForge. AnetTest. `http://anettest.sourceforge.net/`.

[4] M. Bishop, J. Cummins, S. Peisert, A. Singh, D. Agarwal, D. Frincke, and M. Hogarth. Relationships in Data Sanitization: A Study in Scarlet. In *Proceedings of the 2010 New Security Paradigms Workshop*, Concord, MA, September 21–23 2010.

[5] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[6] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner. The role of network trace anonymization under attack, 2010.

[7] Cisco. Cisco IOS NetFlow. `http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html`.

[8] C.-P. T. Cisco. Network log anonymization: Application of.

[9] CNET News. Data Retention. `http://news.cnet.com/8301-13578_3-9926803-38.html#ixzz0zTejcqJw`.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[11] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[12] M. Foukarakis, D. Antoniades, S. Antonatos, and E. P. Markatos. Flexible and high-performance anonymization of netflow records using anontool.

[13] A. Hintz. Fingerprinting websites using traffic analysis. In *Workshop on Privacy Enhancing Technologies*, 2002.

[14] P. Jurczyk and L. Xiong. Privacy-preserving data publishing for horizontally partitioned databases. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1321–1322, New York, NY, USA, 2008. ACM.

[15] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios. A generic anonymization framework for network traffic. In *IEEE International Conference on Communications*.

[16] Manage Engine. NetFlow Analyzer. `http://www.manageengine.com/products/netflow/cisco-netflow.html`.

[17] pstavirs, Google Project Hosting. ostinato. `http://code.google.com/p/ostinato/`.

[18] D. E. Robling Denning. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[19] B. Schneier and J. Kelsey. Cryptographic support for secure logs on untrusted machines. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, pages 4–4, Berkeley, CA, USA, 1998. USENIX Association.

[20] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[21] Solarwinds. Orion NetFlow Traffic Analyzer. `http://www.solarwinds.com/products/orion/nta/`.

[22] Standford Crypto Group. IBE Secure Email. `http://crypto.stanford.edu/ibe/`.

[23] Vincent Rijmen, Joan Daemen. Advanced Encryption Standard. `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`.

[24] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 648–659, New York, NY, USA, 2009. ACM.