# Flexible and High-Performance Anonymization of NetFlow Records using Anontool

Michalis Foukarakis*, Demetres Antoniades*, Spiros Antonatos*, Evangelos P. Markatos*
*Institute of Computer Science (ICS),
Foundation for Research and Technology Hellas (FORTH),
P.O BOX 1385, Heraklion Crete,
GR7-1110, Greece
{mfukar,danton,antonat,markatos}@ics.forth.gr

***Abstract*—Netflow is a protocol widely adopted by the security and performance measurements community. Nowadays, many distributed applications and architectures base their functionality on Netflow data collected at diverse environments. However, communities and administrators are reluctant to share exported Netflow data for privacy reasons. As a consequence, the effectiveness of distributed approaches is limited due to lack of input data. To overcome this limitation, anonymization on Netflow data is proposed for sharing. However, the available tools are either proprietary or of very limited functionality. Towards this direction, we propose and implement *anontool*, that allow administrators to anonymize Netflow data in a highly customizable way. A comparison of anontool with existing solutions is provided along two dimensions: functionality and performance. Anontool can anonymize traffic even at high bandwidth rates, outperforming most of the tools and having same performance with specialized – but very limited in functionality – approaches.**

## I. Introduction

Network management and security is a distributed process that requires information from various sources, located in multiple points of presence. Network activity log sharing has gained significant popularity nowadays, not only among computer security engineers, but also among researchers, developers and educators. Different groups from diverse communities seek different kinds of information within these logs, each for their own purpose. For example, security engineers try to identify anomalies in the traffic pattern, developers try to spot performance problems of their network applications and many more.

To accommodate this increasingly popular need for sharing information as well as the fundamental lack of trust between different communities, several tools have appeared which provide the means to anonymize the potentially sensitive information contained within these logs. Anonymization tools provide an interface to accessing this information and a variety of algorithms to "hide" it from plain view, depending on the interests and concerns of their users. The tradeoff with anonymization is that when most of the information is altered, the data become less useful and descriptive but when anonymization is performed on a small part, sensitive information may be revealed.

A popular format of network activity logs is the Cisco NetFlow [1] format, which provides important information about network usage and traffic routing. The NetFlow format is based on the concept of flow. In general, a flow is defined as a set of packets which share a common property, although several different types of flows have been proposed. Network activity logs usually adopt the concept of a flow in storing information, and subsequently each tool that processes such logs does so. The most recent evolution of the Netflow format is version 9, which is currently the basis of the IETF [2] standard for information export. In the NetFlow definition, Cisco uses the 5-tuple definition of a flow, where a flow is defined as a unidirectional sequence of packets that share source and destination IP addresses, source and destination port numbers, as well as the IP protocol value.

Various tools and techniques have been proposed and implemented for anonymization purposes. However, most of them provide limited functionality and are customized for specific purposes. Our approach is called `anontool` and it is a general-purpose tool that can anonymize live or stored traffic, Anontool is based on the concept of per-field anonymization, that is, users can define what anonymization function should be applied on each application field. Anontool supports a number of protocols but on this work we focus on the Netflow protocol. Anontool offers a variety of anonymization functions and at the same time achieves high level performance, unlike to other tools of its category. In this work, we present a technical overview of anontool, along with a comparison with similar tools. The comparison is made along two major axis: functionality and performance.

The rest of the paper is organized as follows. In Section II we describe the work on anonymization techniques and tools, Section III presents an overview of anontool and its capabilities and Section IV includes the evaluation of anontool against other Netflow anonymization tools. We summarize and conclude in Section V.

## II. Related work

Several works have been done in the field of anonymizing traces but only a limited number of those works address the issue of anonymizing Netflow data. In this Section, we present a review of related work around anonymization, both on general anonymization approaches as well as Netflow-specific ones.

Tcpdpriv [3] is a well-known anonymization tool that takes as input traces written in tcpdump [4] format and removes sensitive information by operating only on packet headers. TCP and UDP payload is simply removed, while the entire IP payload is discarded for protocols other than TCP or UDP. The tool provides multiple levels of anonymization, from leaving fields unchanged up to performing more strict anonymization, like mapping IP addresses to integers or prefix-preserving anonymization. Tcpdpriv works only on TCP/IP headers, thus it does not provide any functionality for Netflow anonymization. Peuhkuri in [5] addresses the problem of IP address anonymization. Cryptographic algorithms that require small amount of memory are applied in order to provide consistent anonymization across different sessions. Xu, Fan, Ammar et al. in [6], [7] also apply cryptographic algorithms to provide prefix-preserving anonymization. Both works however do not extend to Netflow protocol. Paxson and Pang in [8] introduce a way to anonymize the payload of a packet and remove sensitive information instead of removing the entire payload as the other approaches do. Packets are reconstructed into data stream flows and application level parsers modify the data streams as specified by a policy written in a high-level language. The user can specify the field to be altered using regular expressions and the modification to be done. To the best of our knowledge, this work has not yet been extended to Netflow protocol (currently only HTTP and FTP are supported).

Concerning Netflow-specific work, we can identify several approaches. Prefix-preserving anonymization has also been applied to NetFlow [9]. The Crypto-PAn software has been used and modified in order to generate the cryptographic key that is used from a pass phrase. Anonymization is applied only to IP addresses of flows, while all other fields are left unchanged. The authors have extended their tool in [10] where the users are able to anonymize the eight most common fields of a NetFlow record.

NFDUMP [11] is a set of tools for collection and processing of NetFlow data. The *nfdump* tool among them reads NetFlow log files stored by nfcapd and performs prefix-preserving anonymization on them. It is worth noting that *nfdump* uses the Crypto-PAn module to perform this kind of anonymization, and the key for the cryptographic algorithm is user-supplied. A basic principle of the NFDUMP suite is the separation of the storing process from analyzing the data. As a result, a limitation of NFDUMP is the inability to perform anonymization on live traffic (ie. NetFlow export records as sent by Cisco routers etc.), since it can only process stored log files. The current NFDUMP version is 1.5.2, currently offering support for Cisco NetFlow versions 5, 7 and 9.

FLAIM [12](Framework for Log Anonymization and Information Management) is a general framework, created to support the anonymization of heterogeneous logs to multiple levels. FLAIM was developed by the Log Anonymization and Information Management (LAIM) Working Group [13] to overcome the limitations of other tools, such as CANINE [14], which could not be scripted from the command line, and did not offer support of multiple types of logs. FLAIM includes an anonymization engine containing a broad set of anonymization algorithms for various datatypes, an XML based policy engine which validates and parses users' XML policies against a variety of schemes and finally an API governing how parsing modules can pass records back and forth with FLAIM's anonymization engine. At this time, the FLAIM nfdump module supports anonymization of netflows contained only in NFDUMP version 1.4.x logs and not 1.5.x ones, due to changes in the internal NFDUMP format. As a result, it does not support NetFlow version 9. FLAIM provides several anonymization primitives to choose from, such as prefix-preserving anonymization, random permutations of field values and specialized operations on time-related fields. FLAIM focuses on providing generality rather than performace; we believe that *anontool* can provide the same, if not greater, degree of generality while also achieving the maximum performance, similar to very specialized tools with limited functionality, such as NFDUMP. The latest version of FLAIM is 0.5.2.

Although the reader may be confused by our choice to deal with log anonymizers while we present the implementation of a packet trace anonymizer, these tools are currently the only way of anonymizing NetFlow data, and therefore should be considered in the context of NetFlow data anonymization. It is up to the potential user to decide whether she would prefer to store and manipulate packet traces or log files, yet we offer our opinion on this matter in Section IV.

## III. ANONTOOL

Anontool is command line tool that enables users to anonymize both live and stored traffic. Its functionality is based on the Anonymization API (AAPI), described thoroughly in [15]. AAPI allows a user to write his own anonymization applications. User can define the anonymization function to be applied on any field, having the maximum degree of flexibility in defining her policies. AAPI provides a large number of anonymization functions, from setting fields to zero or constant values, prefix-preserving anonymization, hashing with several different hash functions and block ciphers, including but not being limited to, SHA-1 and SHA-2, MD5, CRC32, 3DES and so forth, mapping (direct and probabilistic) and support for regular expression matching and replacement. AAPI supports a wide variety of protocols, ranging from Ethernet to HTTP and FTP in the application layer. All fields of a protocol are being made available to the user application.

AAPI is implemented as a user-level library in the C language; it provides function calls for creating packet "streams", filtering using BPF filters, and of course applying anonymization functions. It uses *libpcap* [16] for packet capturing and writing packet traces on disk. One of the main design goals was to accomodate extensibility, and potential developers are able to write their own protocol decoder in the framework, similar to those already used for FTP, HTTP, or NetFlow protocols, for a target protocol, such as SMTP for instance,

or new anonymization functions. It is also straight forward to write code that supports new input sources, with few code additions, and apply anonymization as usual using the same notation and anonymization functions AAPI provides. The implementation of AAPI is presented in much greater detail in [15].

Since the emerging use of Netflow data, we decided to extend AAPI with support of the Cisco NetFlow packet export format. Taking advantage of the extensibility feature of AAPI we implemented decoding and anonymization functions for both version 5 and the newly defined version 9 of the NetFlow format. Table I shows all the fields and anonymization primitives available regarding NetFlow v5. Bear in mind the names of the functions are merely indicative, and most are highly configurable with extra parameters.

| Protocol | Fields | Functions |
|---|---|---|
| NETFLOW_V5 | VERSION | UNCHANGED |
| NETFLOW_V9 | FLOWCOUNT | MAP |
| | UPTIME | MAP_DISTRIBUTION |
| | UNIX_SECS | STRIP |
| | UNIX_NSECS | RANDOM |
| | SEQUENCE | HASHED |
| | ENGINE_TYPE | PATTERN_FILL |
| | ENGINE_ID | ZERO |
| | SRCADDR | REPLACE |
| | DSTADDR | PREFIX_PRESERVING |
| | NEXTHOP | PREFIX_PRESERVING_MAP |
| | INPUT | CHECKSUM_ADJUST |
| | OUTPUT | REGEXP |
| | DPKTS | |
| | DOCTETS | |
| | FIRST | |
| | LAST | |
| | SRCPORT | |
| | DSTPORT | |
| | TCP_FLAGS | |
| | PROT | |
| | TOS | |
| | SRC_AS | |
| | DST_AS | |
| | SRC_MASK | |
| | DST_MASK | |

TABLE I
TABLE PRESENTS THE NETFLOW FIELDS THAT AAPI MAKES AVAILABLE FOR ANONYMIZATION AND THE FUNCTIONS WHICH CAN BE APPLIED ON THEM.

Exploiting the template-based nature of the v9 format, it provides the user with complete control of every field made available from information export nodes, be it Cisco routers or network monitoring applications which support the NetFlow export format.

Anontool enables users to select the desired anonymization function per field. It can read traffic either from a live interface or from a tcpdump [4] trace file. The anonymized packets can be written to disk, again in tcpdump format. The choice of tcpdump format was made based on the popularity of the format and the fact that can be given as input to many security and network management applications. Other useful options of anontool is that it can automatically fix checksums of anonymized packets (the checksum will be changed once a field of the packet is altered) and its ability to print packets on screen – in human readable form – for manual inspection. An example invocation of *anontool* is the following:

```
./anontool -i eth0 -t ZERO -c /dev/null
```

The above invocation will open NIC named “eth0” for packet capturing, will zero the TCP port numbers of NetFlow records in the packets captured and recompute checksums, then write each packet to */dev/null*. Alternatively, a user could write:

```
./anontool -i eth0 -a PREFIX \
--NF5_TOS RANDOM -c 42.pcap
```

which would open the interface named “eth0” for capturing, then in every NetFlow datagram captured it would perform prefix-preserving anonymization on source and destination IP addresses and replace the value in the TOS field with a random value, then recompute checksums before writing the packets to a pcap file named *42.pcap*

Anontool is a fairly simple C application that uses the AAPI library to support anonymization of packet traces. It does not implement any anonymization functions in itself; it is much more transparent to a user to put all the anonymization functionality in the AAPI implementation which can be used by any application. What it does do, however, is to provide the user with the choice of protocols, functions to apply, etc. in order to create her anonymization policy for a packet trace. It is worth noting, that to maintain simplicity and not overwhelm the user with a plethora of choices, we have not added support for every primitive AAPI provides. Contrary, we have provided a few preset policies which are commonly used and can be selected by a single command line parameter, and we are exploring the possibility of supporting predefined policies which are stored in files in a general-purpose language such as XML. For instance, a user can invoke a predefined policy which will set IP source and destination addresses’ bits to zero, set the values of the TCP port field into new random ones and replace the values of the Uptime field with a random value, and finally generate new checksums for the NetFlow datagrams before writing them to a file named *anon_trace.pcap* by invoking the tool as follows:

```
./anontool -i eth0 -d anon_trace.pcap
```

On the other hand, it is trivial for any user with knowledge of the C language to add another command line option for the function she desires to use.

## IV. Evaluation

### A. Functionality Comparison

Before proceeding with the performance evaluation of the available tools, we are going to briefly discuss the choices they present to the user who wishes to perform anonymization on NetFlow records using each of these tools.

Before we start, we feel it is essential to define what we consider as "flexibility" in the context of the anonymization process. As mentioned earlier in the paper, AAPI was developed by having flexibility in mind. What this means, is that we feel a potential user, who wishes to perform anonymization on any kind of network data, should have the potential to do so in any way she deems fit. As different organizations and institutions as well as different groups of people like researchers or network engineers tend to have different views and interests over network data, it is most likely that they would wish to "hide" or obfuscate different aspects of their owned network data traces or logs. Therefore, providing them with the ability to do so, is a very important factor an anonymization tool developer should bear in mind. We therefore believe that the maximum degree of flexibility in anonymization policy definitions is when the user has complete control over what primitives she can use over all of the data. This is ensured because *anontool* operates on the granularity of protocol fields, and this is the most fine-grained choice a user can have.

Moreover, we feel that supporting packet traces as a source of network data is important, for two reasons. Firstly, the information in packet traces is complete and does not bear any information loss over logs. During our work with anonymization tools we came along with log formats which, in order to achieve storage and computation efficiency, discarded certain packet contents; we feel this should not be imposed implicitly on a user. Secondly, anonymized packet traces can be further processed by tools meant for accounting, intrusion detection or other tools such as NFDUMP which operate on packet traces, without the need for another application that would reconstruct a packet trace from logs. We feel that this is another factor that gives a user the maximum degree of choice between all the different protocol fields a packet may contain, and this contributes to achieving maximum flexibility as defined in the previous paragraph.

NFDUMP provides the user with the simplest and most rigid anonymization policy of the three tools; prefix-preserving anonymization of all the source and destination IP addresses inside the log file. Remember that this is due to the integration of the Crypto-PAn tool in the *nfdump* application. Regarding the supported formats, NFDUMP handles the collection of NetFlow export packets versions 5 and 7, as well as the newest version 9. The log files it stores, however, are not in the packet export format Cisco has defined. The single user-configurable parameter in this setup is the choice of the key used for the cryptographic algorithm which Crypto-PAn implements. While it may prove useful for specialized applications, NFDUMP offers no flexibility when a user wants to consider alternative anonymization policies.

FLAIM offers support for NetFlow versions 5 and 7. Although its modular nature should make adding support for new protocols or log formats easy, at the moment of this writing, it does not support NFDUMP version 1.5 logs, and therefore cannot process NetFlow v9 records. Note that, when it comes to NetFlow anonymization, FLAIM also operates on NFDUMP log files, and not on the Cisco packet export format. Nevertheless, FLAIM presents the user with choice between all of the fields a NetFlow record contains. The user may then choose the desired primitive to be applied on any field of each record, through the use of XML-based documents which describe her anonymization policy. FLAIM has a wide variety of anonymization primitives for the user to choose; wiping field values clean (Black Marker primitive), truncating fields, several types of permutation of a field, hashing, partitioning and a specialized partitioning for time-based fields called Time Unit Annihilation, and enumeration. While a lot in themselves, FLAIM imposes certain restrictions on the algorithms a user can select to apply on each field. For instance, only the BinaryBlackMarker and Annihilation primitives are valid to apply on the Packets field of a NetFlow structure. It is worth noting, the FLAIM user can change the module schema in order to lift those restrictions, but at the same time she is advised not to do so. We feel only experienced users with FLAIM and XML would be able to perform such changes; such assumptions about a user or anonymization policies should, in our opinion, be avoided. Although it may not seem important, it should essentially be up to the user to decide the optimal anonymization policy to apply in each case, which could certainly vary from sharing of network activity logs, to obfuscating certain parameters of the network which could be inferred from the log, if not anonymized properly.

CANINE supports different kinds of NetFlow formats. Among them, the NetFlow v5 and v7, the NFDUMP format, and two NCSA internal formats derived from them. It can anonymize IP addresses, port and protocol numbers, timestamps and the byte counter on each flow record. The algorithms supported on each field resemble closely the ones used by FLAIM; truncation, random permutations, and prefix-preserving anonymization. For the timestamp, it can annihilate certain parts of it, perform random time shifts, or perform an enumeration. There's also a bilateral classification algorithm available for port numbers. Unfortunately, CANINE was considered non-extensible and difficult to script from the command line, so its developers proceeded with the definition and implementation of FLAIM. Due to these factors, but also because FLAIM is a later tool which addresses these difficulties, we will also not consider CANINE in our performance comparison, as it was indeed quite difficult to evaluate its behavior.

Anontool preserves the basic principle of AAPI, which is bent on being generic and flexible. It offers support for NetFlow version 5, which is the most used version supported on routers, and NetFlow version 9, the latest addition to the series,

which has an extensible design and is currently the IETF standard for information export. We chose not to implement support for NetFlow v7, because its a specialized enhancement which is incompatible with the majority of Cisco routers, and therefore not quite popular. As an application based on AAPI, the *anontool* user has complete control over every field which may be present in a NetFlow packet. We have already mentioned in Section III the available choices of fields a user has, and there are no restrictions regarding the operations which a user may apply on them. Regarding the anonymization operations a user can apply, anontool offers a wide variety of primitives to choose from. Starting with the simplest deletion of a field value, or setting it to a fixed value, a user can also choose mapping a field's values to new ones, which may or may not follow a probability distribution, she can strip certain parts of a field, or replace them with a specified value (binary or string). The popular prefix-preserving algorithms are also supported, and so are various hash functions, cryptographic and not. Also, the user can set fields according to a pattern, and specify regular expressions to match and change a part or whole of a field. This last feature is particularly useful when a user would want to eliminate potentially sensitive information which could appear on the packets of an HTTP transaction, such as part of a URL being requested by a browser.

At this point, we believe that having discussed the capabilities of each tool, anontool presents a user with the maximum amount of flexibility, offering complete control over the NetFlow packet export structure. FLAIM also offers a significant amount of choices to the user, yet it places restrictions which a user may find limiting. NFDUMP offers the least capabilities of the three tools. Also, we argue that since anontool operates on packets using libpcap, its output can be used as input to other tools for network management, monitoring, or accounting, and thus it can be used in conjunction with other tools, including FLAIM and NFDUMP. This is not the case with the NFDUMP log format, unless there are specialized converters which perform this task. Yet the process of conversion takes time and makes the whole process tedious and prone to error.

### *B. Performance comparison*

Recently, NetFlow data are used for security purposes and anomaly detection( [17]–[19]). In the field of computer security, high performance and timely responses to threats are of paramount importance. Therefore, if anonymized network flow data are to be used and shared for security purposes, we should explore how fast the anonymization process can be completed.

In this section, we present performance evaluation of the available tools, described in Section II. In order to perform the performance evaluation we used a real traffic trace collected from a monitoring sensor located at the University of Crete. The trace was collected from 26/03/2007 morning through 27/03/2007 afternoon and contains 7328264 flows presenting total traffic of 94.1 GB. The trace itself was 857 MB large.

To perform our evaluation, we used the most recent versions of the tools available; *1.0* for anontool, *1.5.2* for NFDUMP, and *0.5.2* for FLAIM. All the figures we present are means taken out of 20 iterations.

Since both NFDUMP and FLAIM require the collection of NetFlow data from the network before the actual anonymization process can take place, we used the nfcapd daemon, supplied with the NFDUMP tools, to convert the trace into the NFDUMP format and calculated the sum of user and system time needed for the conversion in the total time needed for the trace anonymization. As this collection/conversion process is required for log processing by the aforementioned tools, we feel this extra cost should be taken into consideration, as the result is a very close approximation of a "direct" comparison.

In this point we would like to argue, once more, that sharing network level traces is more useful than sharing logs, since the user can use the traces for several purposes; i.e. she can translate the trace into flows in the format that is more suitable for the analysis she wants to perform, or she can use the anonymized network trace in order to evaluate NetFlow collection or translation tools.

In our first experiment we choose to follow the NFDUMP anonymization policy, which we implement both with AAPI and FLAIM. NFDUMP deploys prefix preserving anonymization only in the flow source and destination IP addresses. The results are presented in Table II. As we can see the performance of our implementation and NFDUMP is similar but our tool has the ability of deploying anonymization in all fields of the Netflow implementation. FLAIM presents 5 times worse performance, which are attributed to high memory consumption which led to excessive swapping operations (as indicated by *vmstat* tool). This performance problem appears in FLAIM's current release (0.5.2) at the time of this writing as well as in the previous release we tested (0.5.1).

| Tool | User + System Time (secs) | CPU Load (%) |
|---|---|---|
| anontool | 233.76 | 94.2 |
| NFdump | 232.35 | 94.6 |
| FLAIM | 1237.33 | 86.9 |

TABLE II
COMPARISON WHILE DEPLOYING PREFIX PRESERVING IP ADDRESS ANONYMIZATION

In our second experiment we choose to implement a different anonymization policy. We zero both source and destination IP addresses in all Netflow records. Since NFdump has a single anonymization policy we can compare only with FLAIM. As the results from Table III indicate, our implementation is again one order of magnitude faster than FLAIM and also requires half the CPU utilization that FLAIM does. This shows that our tool would be able to anonymize Network Flow data on the fly without any loss even in high bandwidth rates (the 20 seconds needed by anontool are translated to 350 Mbps throughput),

while other tools would require first to capture the data and then follow the anonymization procedure.

| Tool | User + System Time (secs) | CPU Load (%) |
|---|---|---|
| anontool | 19.76 | 46.80 |
| FLAIM | 1168.8 | 99.9 |

TABLE III
COMPARISON WHILE DEPLOYING ZERO IP ANONYMIZATION

Finally, we measured the time it took anontool and FLAIM to execute the predefined anonymization policy we saw at the end of Section III. This policy instructs that source and destination IP addresses are set to zero, the TCP port fields are set to a random value and the Uptime field is also set to a random value. The results appear in table IV.

| Tool | User + System Time (secs) | CPU Load (%) |
|---|---|---|
| anontool | 36.58 | 36.48 |
| FLAIM | 1336.01 | 94.9 |

TABLE IV
COMPARISON WHILE DEPLOYING A PREDEFINED ANONYMIZATION POLICY WITH MULTIPLE FUNCTIONS (ZERO SOURCE AND DESTINATION IP ADDRESSES, RANDOM TCP PORT NUMBERS AND RANDOM UPTIME)

## V. SUMMARY AND CONCLUDING REMARKS

We have presented *anontool*, a tool which provides anonymization capabilities for both stored and live traffic. Anontool is implemented using AAPI, a powerful and flexible anonymization API, and shows that it is possible to provide the means for developers, researchers, and network engineers to anonymize NetFlow data in an efficient and highly customizable manner. We compared it to other tools which were developed for the same purpose along two major axes. The first one was functionality. Anontool provides the widest range of anonymization functions that can be applied to all fields of the Netflow protocol while other tools offer a limited number of functions on a few fields. Furthermore, anontool can work directly on live traffic and uses widely-used format for input and output. The second axis was performance. We have found that *anontool* outperforms similar frameworks, like FLAIM, and has the same performance as very specialized approaches, like NFDUMP, without having to sacrifice flexibility in the anonymization policies a user could define.

## VI. AVAILABILITY

Anontool can be downloaded from http://dcs.ics.forth.gr/Activities/Projects/anontool.html. The application has been installed and tested to Redhat and Debian operating systems.

## REFERENCES


[1] Cisco Systems, Inc, "Netflow Specification." [Online]. Available: http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml

[2] "Ip flow information export (ipfix)." [Online]. Available: http://www.ietf.org/html.charters/ipfix-charter.html

[3] Greg Minshall, "Tcpdpriv." [Online]. Available: http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html

[4] "Tcpdump/libpcap official site." [Online]. Available: http://www.tcpdump.org

[5] M. Peuhkuri, "A method to compress and anonymize packet traces," *Internet Measurement Workshop (San Francisco, California, USA: 2001)*, pp. 257–261, 2001.

[6] J. Xu, J. Fan, M. Ammar, and S. B. Moon, "On the design and performance of prefix-preserving ip traffic trace anonymization," *Internet Measurement Workshop (San Francisco, CA, USA: 2001)*, pp. 263–266, 2001. [Online]. Available: citeseer.nj.nec.com/462352.html

[7] J. Xu, J. Fan, M. Ammar, and S. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," *ICNP 2002*, 2002.

[8] R. Pang and V. Paxson, "A High-Level Programming Environment for Packet Trace Anonymization and Transformation," in *Proceedings of the ACM SIGCOMM Conference*, August 2003.

[9] J. W. A. Slagell and W. Yurcik, "Network log anonymization: Application of crypto-pan to cisco netflows," *NSF/AFRL Workshop on Secure Knowledge Management (SKM)*, 2004.

[10] A. Slagell, Y. Li, and K. Luo, "Sharing network logs for computer forensics: A new tool for the anonymization of netflow records," *Computer Network Forensics Research (CNFR) Workshop*, 2005.

[11] "Nfdump tools collection." [Online]. Available: http://nfdump.sourceforge.net/

[12] A. J. Slagell, K. Lakkaraju, and K. Luo, "Flaim: A multi-level anonymization framework for computer and network logs." in *LISA*, 2006, pp. 63–77.

[13] "Log anonymization and information management working group." [Online]. Available: http://laim.ncsa.uiuc.edu/

[14] Y. Li, A. Slagell, K. Luo, and W. Yurcik, "Canine: A combined converter and anonymizer tool for processing netflows for security," in *Proceedings of the International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM)*, Nov. 2005.

[15] D. Koukis, S. Antonatos, D. Antoniades, P. Trimintzios, and E. Markatos, "A generic anonymization framework for network traffic," in *Proceedings of the IEEE International Conference on Communications (ICC 2006)*, Jun. 2006.

[16] S. McCanne, C. Leres, and V. Jacobson, "libpcap," lawrence Berkeley Laboratory, Berkeley, CA. [Online]. Available: http://www.tcpdump.org/

[17] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," 2004. [Online]. Available: citeseer.ist.psu.edu/715839.html

[18] T. T.Dbendorfer, A.Wagner and B.Plattner, "Flow-level traffic analysis of teh blaster and sobig worm outbreaks in an internet backbone," in *Proceedings of DIMVA 2005, Springer's Lecture Notes in Computer Science*, 2005.

[19] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," 2001. [Online]. Available: citeseer.ist.psu.edu/barford01characteristics.html