High-Speed Prefix-Preserving IP Address Anonymization for Passive Measurement Systems

Ramaswamy Ramaswamy and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
{rramaswa,wolf}@ecs.umass.edu
Technical Report: TR-06-CSE-01

***Abstract*— Passive network measurement and packet header trace collection are vital tools for network operation and research. To protect a user's privacy, it is necessary to anonymize header fields, particularly IP addresses. To preserve the correlation between IP addresses, prefix-preserving anonymization has been proposed. The limitations of this approach for a high-performance measurement system are the need for complex cryptographic computations and potentially large amounts of memory. We propose a new prefix-preserving anonymization algorithm, top-hash subtree-replicated anonymization (TSA), that features three novel improvements: precomputation, replicated subtrees, and top hashing. TSA makes anonymization practical to be implemented on network processors or dedicated logic at Gigabit rates. The performance of TSA is compared with a conventional cryptography based prefix-preserving anonymization scheme which utilizes caching. TSA performs better as it requires no online cryptographic computation and a small number of memory lookups per packet. Our analytic comparison of the susceptibility to attacks between conventional anonymization and our approach shows that TSA performs better for small scale attacks and comparably for medium scale attacks. The processing cost for TSA is reduced by two orders of magnitude and the memory requirements are a few Megabytes. The ability to tune the memory requirements and security level makes TSA ideal for a broad range of network systems with different capabilities.**

I. Introduction

For the operation of networks and networking research it is crucial to easily obtain measurements of network traffic. Passive measurement systems observe all the traffic that crosses a particular node or link and record packet header information. These packet traces are then used to extract statistics and infer network behavior. While network traces provide extremely useful data for network engineers and researchers, they also pose a problem in terms of privacy. Many personal and business transactions are performed over the Internet and it is imperative that the privacy of a network user is maintained. The IP addresses reveal the source and destination of each communication and it is a simple exercise to determine, for example, who is browsing which web servers.

To ensure that no private information is revealed in a network trace, sensitive header fields need to be sanitized. In most cases these fields are the IP source and destination address, which are the focus of this paper. However, IP source and destination addresses cannot simply be removed from the trace, as they are necessary to derive any useful networking statistic from the trace. Instead, IP addresses are "anonymized." The anonymization operation is a one-to-one mapping between the original IP address as seen on the network and the anonymized IP address that is used in the trace. By employing cryptographic hash functions, this mapping cannot be guessed easily and thus is secure in a cryptographic sense. The main constraint on the anonymization algorithm is that it should be "prefix-preserving." This means that if two original IP addresses have a common prefix of length l, then the anonymized IP addresses should also have a common prefix of exactly length l. This ensures that addresses that are "closely" co-located in a network (e.g., in the same subnet) are also closely co-located in the anonymized trace. Thus, some information on network-level characteristics of the measured traffic can be preserved across the anonymization step. The details of such a prefix-preserving mapping for IP addresses is discussed in detail in Section III.

In a measurement system, it is desirable to perform trace anonymization as early in the collection process as possible. By anonymizing header fields on the measurement node itself instead of external post-processing, it is less likely that unanonymized data is leaked. This requires the anonymization process to operate at a speed that can keep up with the link rates of the measurement node. This sort of *online* anonymization, however, cannot be achieved with current prefix-preserving anonymization algorithms. The prefix-preserving anonymization presented in [17] requires up to 32 cryptographic hash (e.g., MD5) computations per IP address. In our measurements in Section VI, we show that this translates into 247,078 RISC instructions on a microprocessor. With link rates in the order of Gigabits per second it is not possible to implement the required processing in a cost-effective way. Even dedicated cryptographic hardware cannot easily achieve this processing rate.

In this paper, we present a novel prefix-preserving anonymization algorithm, called TSA (top-hash subtree-replicated anonymization), that addresses this problem by computing all necessary cryptographic functions offline. In TSA, the anonymization of an IP address only requires 359 instructions and 26 memory lookups. The ability to limit the amount of memory that is necessary for TSA to a few Megabytes makes TSA an ideal algorithm to be implemented on router ports that are equipped with network processors.

We compare the performance of TSA to that of a conventional cryptography based anonymization scheme that caches the anonymization mapping for active IP addresses to avoid recomputation. However, from a network systems point of view, it is not desirable to rely on any form of locality in network traffic. We show that the rate at which new IP addresses appear (i.e. compulsory misses in a cache) is sufficiently large enough to to render caching ineffective.

A major challenge that needs to be addressed in the context of anonymization is the issue of attacks on trace collection systems with the intent to compromise anonymized addresses. By injecting traffic with a particular pattern that can be identified by the attacker in the anonymized trace, the mapping between an original and an anonymized IP addresses can be established. The problem with prefix-preserving anonymization is that even a single compromised address reveals a significant amount of information about other IP address mappings. In Section IV, we develop an analytic model of the security performance of conventional anonymization and TSA under a worst-case attack scenario to evaluate the tradeoffs between the different approaches.

The "active" attack described above, provides a lower bound on the security achievable by any prefix preserving anonymization algorithm. We also investigate the effects of a "passive" attack, in which an attacker, given an anonymized trace, attempts to reveal address mappings in a random order using inference attacks [10] or frequency analysis [17].

TSA is based on three additions to conventional prefix-preserving anonymization, which address the above challenges and makes TSA a practical solution for high-performance network trace collection engines. These additions are:

- **Precomputation:** All necessary computation is done offline. Anonymization is reduced to lookups in a data structure.
- **Subtree Replication:** Precomputation requires a large amount of space to store the results. Subtree replication reduces this space.
- **Top Hashing:** Top hashing limits the prefix preserving properties to areas of the IP address, where it is necessary. This reveals less information to a potential attacker.

In Section II, we discuss existing anonymization methods and other related work. Section III introduces the concept of an anonymization tree and the details of the TSA algorithm. Section IV introduces security metrics and provides an analytic comparison of the resistance to attacks between TSA and a conventional implementation of anonymization. The effect of a random passive attack is also discussed in this section. Section V illustrates the tradeoffs between security and space requirements and identifies the optimal configuration of the TSA algorithm. Section VI discusses implementation issues of TSA and shows processing performance results. Section VII compares the performance of TSA to a conventional anonymization scheme that utilizes caching. Section VIII summarizes and concludes this paper.

II. Related Work

Traditionally, two approaches have been taken towards network measurement: active and passive [2]. In the active approach, a sender and/or receiver measure and record the traffic that they send/receive, obtaining end-to-end (e.g., path) characteristics . NLANR's AMP (Active Measurement Project) [8] and Surveyor [14] are large-scale measurement infrastructures that perform such active measurements. Since active measurement involves only traffic that was injected for the purpose of measurement, no privacy issues arise and packet anonymization is not necessary.

In the passive approach, measurements are taken at a given point in a network and packet headers are observed and collected to determine local characteristics of the network and its traffic. The analyzed traffic is generated by users of the network and the privacy of their communications needs to be protected. For this reason, packet payloads are typically not collected and packet header fields (in particular IP addresses) are anonymized.

A simple way of anonymizing IP addresses is to assign an arbitrary IP address mapping. One common technique is to assign 10.0.0.1 to the first IP address observed in a trace, 10.0.0.2 to the second unique IP address and so on. Recurrences of IP addresses are mapped to the same anonymized address as the first occurrence. This method allows the identification of packets that belong to the same flow, but correlations between flows get lost. From a network management and research point of view it is desirable to identify packets that originate from the same subnet (e.g., to identify correlations in DDoS attacks).

To strike a balance between privacy and usefulness of traces, prefix-preserving anonymization has been proposed, which anonymizes IP addresses while preserving the prefix nature of IP addresses. This has been implemented in *tcpdpriv* [6] and Crypto-PAN [17]. In Crypto-PAN incremental cryptographic hash computations [13][7] are used to determine the address mapping (executing approximately 247,078 RISC instruction to anonymize a single address). In *tcpdpriv*, the anonymization involves pseudorandom functions that cause the address mappings to depend on traffic patterns and differ across traces. The (raw, anonymized) mapping pairs are stored in a table to maintain consistency. To avoid raw addresses being mapped to different anonymized addresses, this table needs to be distributed to all measurement nodes which is cumbersome and makes *tcpdpriv* unsuitable for multinode measurements. Moreover, the memory required by *tcpdriv* depends on the number of entries in the table and grows larger as more raw addresses are seen in the trace. In [10], a high level environment which supports anonymization of both packet headers and payloads using a policy script is introduced. IP addresses are anonymized sequentially using a one-to-one mapping and are not prefix preserving. The methods proposed in this paper are suitable for offline anonymization of packet traces. A cryptography based solution to compress and anonymize packet traces is presented in [11]. However, this method is too computationally intensive to be performed online at a measurement node.

With increasing link speeds, network measurement systems need to collect data at higher and higher traffic rates. High-performance measurement systems have been developed (e.g., Sprint's IPMON project [4], AT&T's Gigascope project [3],

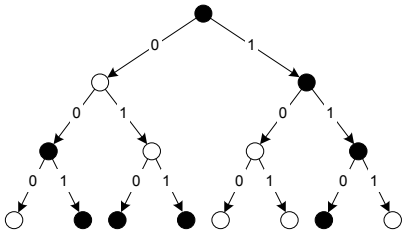

Fig. 1. Anonymization Tree. Black nodes indicate that bits in the original address are flipped to obtain the anonymized address.

and NLANR's passive measurement efforts [9]) to allow continuous monitoring of traffic. This requires that packets can be anonymized *online*. This is currently not possible for high speed links as cryptographic hashing is processing intensive. Instead anonymization is performed offline. In our work, we present an algorithm that allows online prefix-preserving anonymization with very little processing requirement and small space complexity.

III. TSA Algorithm

Before describing the details of our proposed anonymization algorithm, we briefly discuss the concept of an anonymization tree. This is a useful visual representation of the anonymization function and is used to describe TSA below.

In prefix-preserving anonymization, two original addresses IP_1 and IP_2 that share a common prefix of length l are translated by the anonymization function A into different addresses $A(IP_1)$ and $A(IP_2)$. In our terminology, we call the cleartext, unanonymized addresses as "original" addresses and the result of the anonymization as "anonymized" addresses. Due to the prefix-preserving nature of A, the anonymized addresses $A(IP_1)$ and $A(IP_2)$ must also have a common prefix of exactly length l.

Since this property must hold for all pairs of addresses, the anonymization function A, can be seen as a function that takes the prefix tree as input and "flips" some of the nodes. The result of a flip is that the left child becomes the right and vice versa. This function A can then be represented by a binary tree, where a flipped node is illustrated in black and a non-flipped node in white [17]. We call such a tree an "anonymization tree," and an example for addresses with length 3 is shown in Figure 1. Note that any combination of black and white nodes in the tree yields a correct prefix-preserving anonymization function (however simplistic if all nodes are white).

When an IP address is anonymized, a lookup in the tree that represents A is performed using the bit sequence of the original IP address. The sequence of black and white nodes that are encountered during this lookup is recorded. The original IP address is then XOR-ed with the lookup sequence, where black nodes translate into ones and white nodes into zeros. The resulting bit sequence is the anonymized IP address. This process is illustrated in Figure 2. It can be observed that any two addresses with a common prefix of length l share the same anonymization lookup sequence for the first l bits and thus share the same anonymized prefix.

In order to achieve the goal of anonymization, it should not be possible to easily guess the color of some or all of the tree nodes in A. For this purpose, cryptographic hash functions are used to compute if a tree node should be flipped or not. Without the knowledge of the key, the choice of color appears to be random and thus cannot be guessed. If the colors of the nodes of the anonymization tree can be revealed through an attack on the anonymization system, then a simple bitwise repetition of the XOR operation shown in Figure 2 on the anonymized address reveals the original IP address.

In [17], the computation of the node color is done incrementally by using the color of the previous nodes and the address prefix as inputs. This requires 32 successive cryptographic computations and is computationally expensive. We address these problems by using precomputation and subtree replication. With precomputation, the cryptographic processing can be done efficiently offline and the per-address computation is reduced to several table lookups. Due to the space explosion caused by precomputation, we use subtree replication to keep space requirements tractable. This comes at the cost of an increased vulnerability to attacks on the privacy, which is analyzed in detail in Section IV.

Another important observation is that nodes that are closer to the root of the anonymization tree are more valuable to a potential attacker. If the attacker can determine the color of the root node, then the first bit of all addresses in the address space is revealed. The problem is that if the anonymization tree lookup sequence for even a single address is known the color of the root node is known. This leads to the problem that 20% of all address bits can be revealed by cracking just 22 such mappings. To alleviate this problem, we propose the use of top hashing.

The details of our proposed top-hashed subtree-replicated anonymization (TSA) algorithm are discussed in the following subsections.

A. Precomputation

In [17] and [6], anonymization mappings are calculated whenever new addresses are encountered. The motivation for this approach is to do as few cryptographic computations as necessary. For each address that is anonymized, the color of the nodes in the anonymization tree are computed and stored. This builds the anonymization tree incrementally and is based on the addresses that are actually observed on the network (illustrated in Figure 3). As we have previously mentioned, these cryptographic operations are so complex that they cannot be performed online on a high-speed measurement node.

The only way to avoid *online* cryptographic processing is to precalculate the entire anonymization function A. In practice a measurement node can be expected to observe IP addresses from only a small fraction of the total address space. It is still necessary to precalculate the anonymization mapping for all addresses in the address space as traffic patterns are not known beforehand. This implies that the mapping for all 4 billion possible IPv4 addresses needs to be calculated and stored (illustrated in Figure 4). For 32-bit addresses, this yields a tree with $2^{32} - 1$ nodes of color white or black. The minimum amount of memory for such a data structure is 4Gbits = 512MB where each node is represented as a single

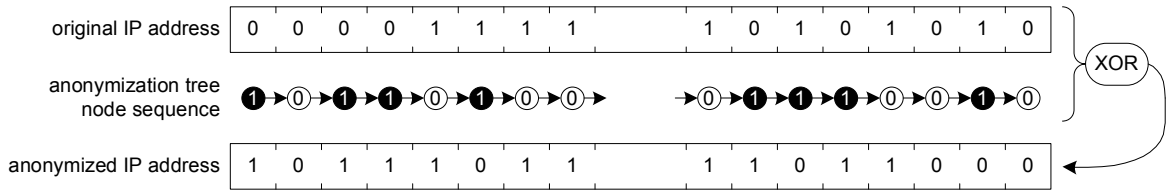

Fig. 2. Illustration of Anonymization Process. The original address is XOR-ed with the lookup sequence in the anonymization tree. Black nodes represent a '1' causing bit flips to obtain the anonymized address.

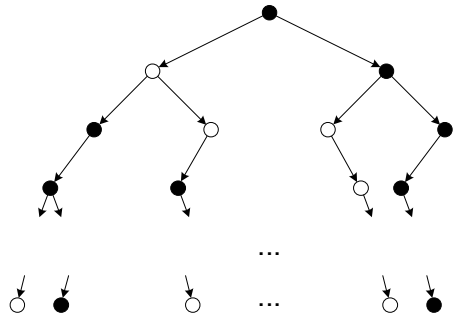

Fig. 3. Incremental Online Computation of Anonymization Tree. Tree nodes are added for addresses that are anonymized.

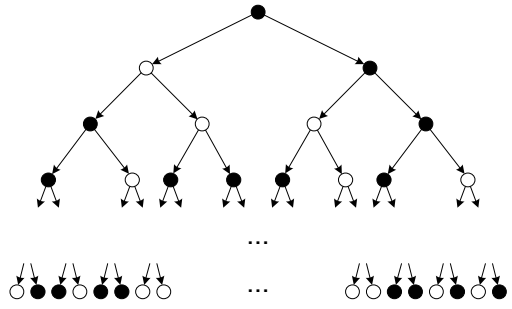

Fig. 4. Precomputation of Entire Anonymization Tree. Tree nodes are computed offline for any possible address.

bit. Pointers between nodes are not necessary because the binary tree is complete. Conventional compression algorithms cannot be employed to reduce the memory requirement due to the pseudo-random nature of cryptographic functions that are used to determine the node colors.

With a precomputed anonymization tree, an address mapping can be determined by performing 32 lookups in the binary tree. No cryptographic computations are necessary, but the required memory is very large. While it is conceivable that next-generation measurement nodes could afford to allocate half a Gigabyte of memory to store the precomputed anonymization tree, it is still desirable to reduce the memory requirements. This is particularly important for high-performance routers with measurement capabilities because a copy of the anonymization tree is required on each port and needs to be implemented in faster, more expensive SRAM.

We address the issue of memory requirements by replicating subtrees in the anonymization tree.

B. Subtree Replication

The major drawback of precomputation is the amount of memory necessary to store the entire anonymization tree with $2^{32} - 1$ nodes. It is necessary to have the entire tree available because the IP addresses of the network traffic are unknown *a priori*. At the same time, most network traffic is limited in the number of distinct addresses that can be observed. This leads to the idea of replicating anonymization subtrees instead of having a distinct anonymization subtree for each prefix.

The subtree replication scheme used in TSA is illustrated in Figure 5. Subtrees are enclosed by triangles and are complete binary trees as described above. Instead of having 2^l unique subtrees for level l, we create only a small number of unique subtrees. These are virtually replicated and act as subtrees for all 2^l nodes in this level. In Figure 5, there are 8 subtrees of which 2 are unique. The remaining 6 subtrees are copies of the 2 unique subtrees (these are shaded in Figure 5). This significantly reduces the memory requirement as only the unique subtrees need to be stored. The choice of which subtree to use to anonymize an IP address is determined by performing a mod operation on a portion of the original IP address which returns a value between one and the number of unique subtrees.

The drawback of subtree replication is that this anonymization mapping can be revealed more easily. If the color of nodes in a subtree are revealed *and* the attacker knows which subtrees are copies of each other then one revealed address can cause a large amount of anonymization information to be revealed.

To make it harder to identify which subtrees are duplicates of each other, we consider a modification to subtree replication, called subtree replication with hashed mapping. In this scheme, the choice of which subtree to use is determined by a cryptographic hash function which is also precomputed for all possible unanonymized IP addresses and stored in a table. A lookup operation is performed on this table using a portion of the unanonymized IP address as an index to determine which subtree to use for anonymizing the IP address. This makes it more difficult to attack the anonymization algorithm, but also requires more memory to store the table containing the precomputed hash values.

The impact of the choice of size and number of unique subtrees and the use of hashed mapping is analyzed in detail in Section IV. We show that subtree replication can be configured to be reasonably secure while requiring significantly less memory than a complete anonymization tree.

To alleviate the decrease in security due to subtree replication, we discuss next a method of increasing security for small scale attacks by protecting the nodes close to the root of the anonymization tree.

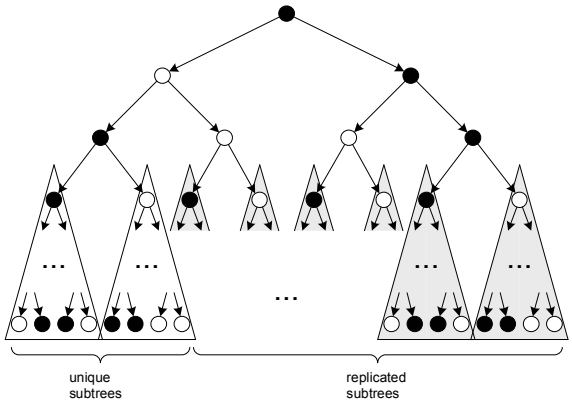

Fig. 5. Subtree Replication in Anonymization Tree. Reuse of subtrees reduces memory requirements.

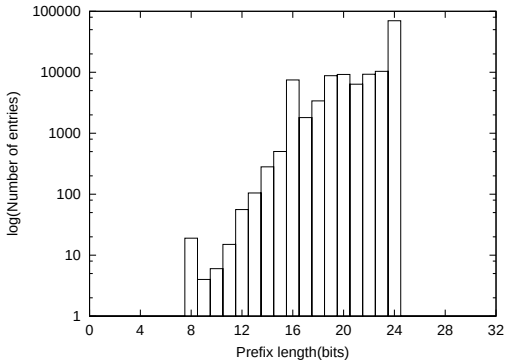

Fig. 6. IPv4 Prefix Length Distribution.

C. Top Hashing

From a security point of view, one of the major drawbacks of prefix preserving anonymization is that there is a correlation between the original IP addresses and the anonymized addresses. If two addresses share a prefix of a certain length, they share a prefix of the same length in the anonymized trace. This of course is desirable from a networking research point of view as the prefix relation gives clues of network traffic properties.

There are two key observations that lead to our proposed top hashing improvement. First, the nodes closest to the root in the anonymization tree reveal most information about other addresses if their color is revealed. They represent short prefixes that are shared by a large number of addresses. If the root node is revealed, the first bit of *all* anonymized IP addresses is known. Second, in the Internet the prefix nature of IP addresses is only relevant for the lower 24 bits of an IPv4 address. This is due to the historical assignment of IP addresses. There are no address blocks larger than 2^{24} addresses (formerly Class A). Thus, if two addresses share a prefix that is shorter than 8 bits, they are not any "closer" in networking terms than two addresses that do not share a common prefix. Figure 6 shows the prefixes of a BGP table obtained from the Route Views Project [1] (AT&T, ID 7018, RIB date 01/23/04) and supports the observation that the shortest common prefixes occurring in the Internet are 8 bits of length.

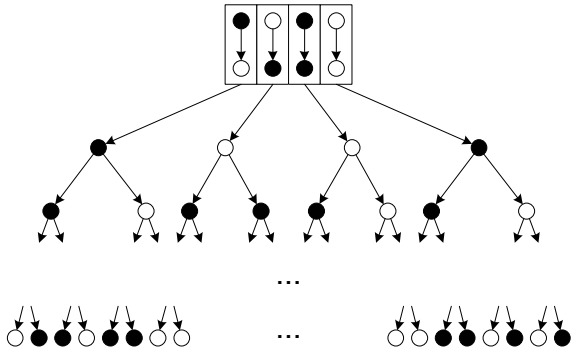

Fig. 7. Top Hashing of Anonymization Tree. Hashing of initial bits limits prefix preserving property to subtrees and increases resilience to attacks.

The top hashing in TSA exploits this absence of short prefixes to improve the resistance of the anonymization algorithm to attack. Instead of using an anonymization tree for the most significant bits of an address, a cryptographic hash function is used to anonymize these bits. The remaining bits are anonymized with the conventional anonymization tree. This is illustrated in Figure 7. The benefit of this approach is that the hash function removes all correlation between the prefixes and thus a compromised address does not reveal information about other address prefixes. The cost of this approach is that the prefix nature of the most significant bits is not preserved, but as shown above, this has no practical impact on IPv4 network traces.

Top hashing is an improvement that is not specific to TSA. It is applicable to other anonymization algorithms (such as [17]). We show in Section VI that top hashing improves performance by reducing processing requirements while increasing security for both TSA and other conventional anonymization algorithms.

IV. Security Analysis

The proposed TSA algorithm makes several changes to conventional prefix-preserving anonymization using an anonymization tree. These changes improve the performance of the anonymization (no online cryptographic computation, reduction of space requirement due to subtree replication), but also impact the "security" of the anonymization process. We use the term "security" to mean the ability of the anonymization algorithm to resist an attack that intends to reveal mappings between original and anonymized IP addresses. We do this by determining how easily an attacker can determine the colors of nodes in the anonymization tree (or a hash table) and how much information is revealed about other address mappings that share the same common prefix.

To understand the impact of using top hashing and subtree replication, we derive expressions for various security metrics and compare them to conventional prefix-preserving anonymization. To obtain general results, we consider a range of possible parameter configurations, which are shown in Figure 8. Top hashing is performed on the top t most significant bits of the IP address. Subtree replication uses r unique subtrees of height s.

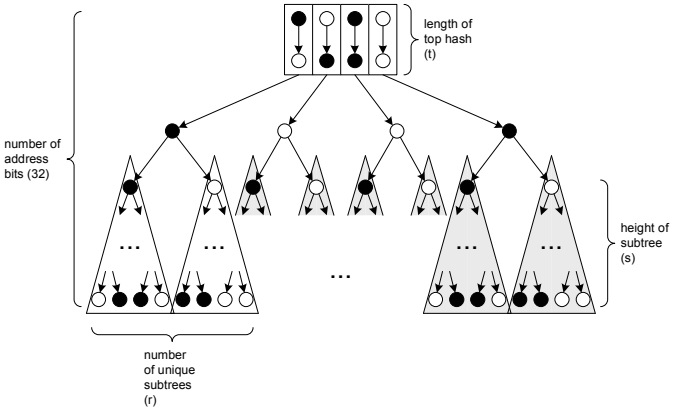

Fig. 8. Configurable Parameters in TSA.

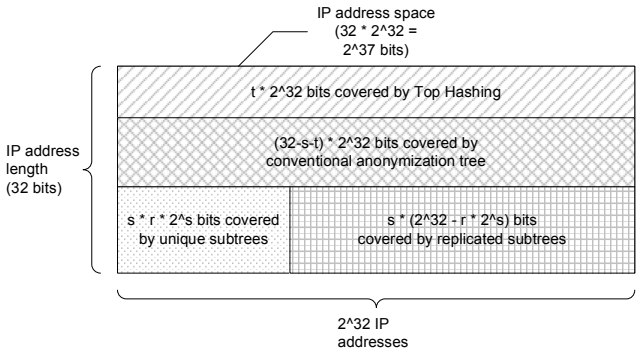

Fig. 9. Illustration of Relation Between IP Address Space and TSA Components.

A. Attack Model

The anonymization mechanism is under attack when the mapping of original IP addresses to anonymized IP addresses is revealed. There are two ways by which an attacker can compromise the anonymization mapping of an IP address: active and passive attacks.

In an active attack, traffic of a certain pattern with carefully selected IP addresses is injected into the network and then identified in the anonymized trace. For example, one could send a sequence of n packets with packet lengths of $l_1 \dots l_n$. If the sequence is chosen randomly and n is sufficiently large, the pattern of packet lengths is unique in the anonymized trace. Then the attacker knows the IP addresses used for the original traffic as well as the IP addresses that were generated by the modification. Using a simple XOR computation as shown in Figure 2, the color of the nodes in the anonymization tree can be determined. With each IP address mapping that is compromised in such a way, the attacker gains more information about the anonymization tree.

In a passive attack, the attacker can only observe the anonymized traffic, but not actively inject traffic. This is the case when an attack is aimed at a recorded trace. Instead of injecting traffic, statistical methods and heuristics can be employed to identify frequently used IP addresses (e.g., popular web servers) with the goal of mapping them to frequently occurring anonymized addresses. This makes passive attacks more complex to analyze than active attacks. We have simulated a passive attack on an anonymized trace. The results are presented in Section IV-G.

We are interested in analyzing the performance of the anonymization algorithms under a worst case attack model. This assumes that the attacker can inject arbitrary traffic and easily identify this traffic in the anonymized trace. The key question is how much information can be revealed in such an attack given that the attacker can compromise i IP address mappings of his/her choice. It is always assumed that the attacker chooses the IP addresses in such a way as to maximize the amount of information that can be revealed. Typically, this means that the IP addresses are chosen to reveal different paths in the anonymization tree that are as distinct as possible. This is called a semantic attack in [17].

B. Security Metrics

In this security analysis, we use two metrics that have been introduced in [17] to evaluate the security of prefix preserving anonymization schemes against semantic attacks. These metrics are:

- The number of unknown non-leaf nodes in the anonymization function, C: The value of C represents the number of non-leaf nodes in the anonymization tree whose color (i.e., if they flip bits or not) is unknown. This metric is useful for understanding which parts of the anonymization tree have been revealed. It however does not capture the impact of revealing different nodes (e.g., revealing the root node has an impact on 2^{32} addresses whereas a leaf node impacts only two addresses).
- The number of unknown address bits, U: This is the total number of bits that are unknown in all 2^{32} addresses of the complete address space. When an address is compromised, all bits in the address are revealed. Additionally, certain bits in other anonymized addresses may also be revealed due to the nature of prefix-preserving anonymization. Figure 9 illustrates the $32 \cdot 2^{32}$ bits of all IP addresses that are represented in U. Different components of the anonymization scheme "protect" these bits from becoming compromised.

We determine C and U as a function of i, which is the number of addresses that have been compromised by the attacker (where $0 \leq i \leq 2^{32}$). The larger the values of C and U for a given i, the more resistant an anonymization scheme is to attacks. As explained above, we assume a worst case attack model where the attacker chooses the sequence of addresses so as to reduce C and U as much as possible.

C. Security Analysis for Conventional Anonymization

Conventional anonymization (CA) represents the algorithm that uses the anonymization tree as shown in Figure 3. This is equivalent to precomputation and it makes no difference in terms of security if the tree node colors are computed online or offline. Thus, we assume a complete binary tree with 32 levels for the anonymization function. The attacker uses the worst case attack pattern that uses distinct paths in the anonymization tree and reveals the nodes closest to the root as quickly as possible.

Initially, C_0^{CA}, the number of unknown nodes in the anonymization tree when no addresses have been compromised, is simply the number of non-leaf nodes in the anonymization tree. Note that C_0 depends on the algorithm that is used (CA in this case) and varies between different anonymization schemes. Since the number of non-leaf nodes in a complete binary tree of height, h is $2^h - 1$, it follows that:

$$C_0^{CA} = 2^{32} - 1. \tag{1}$$

As each address is compromised, a certain number of internal nodes in the anonymization function are revealed. We represent this quantity as $\overline{C_i^{CA}}$, the number of *known* nodes in the anonymization function as i addresses are compromised. When one address is compromised, we reveal the root node of the anonymization tree and one node in each of the remaining 31 levels. Under a worst case attack, the next address that is compromised reveals (again) the root node, and different nodes on the remaining 31 levels. Thus, for $i = 2$, the root node plus two nodes on each of the next 31 levels are known. The maximum number of nodes known in each level l is 2^l. The total number of nodes revealed after i addresses have been compromised is:

$$\overline{C_i^{CA}} = \sum_{l=0}^{31} min(i, 2^l), \tag{2}$$

where l, represents the level of the binary tree on which a particular node is present ($0 \leq l \leq 31$).

Finally, C_i is given by:

$$C_i^{CA} = C_0^{CA} - \overline{C_i^{CA}} \qquad i = 1, 2, \ldots, 2^{32}, \tag{3}$$

where C_0^{CA} and $\overline{C_i^{CA}}$ are given by Equations (1) and (2) respectively.

The metric U_i^{CA}, which weighs the impact of a revealed node on the entire address space, can be expressed in a similar way. Since we have 2^{32} addresses and each address is 32 bits long, U_0, the number of unknown address bits when no addresses are compromised, is given by:

$$U_0 = 2^{32} \cdot 32 = 2^{37}. \tag{4}$$

U_0 is independent of the particular anonymization algorithm used as it pertains to the IP address space. As i addresses are compromised, the number of address bits that are revealed in the entire address space is represented by $\overline{U_i^{CA}}$. When the root node is revealed, the most significant bit for 2^{32} addresses is revealed. When one node on the next level ($l = 1$) is revealed, the second bit is revealed for 2^{31} addresses. When both the nodes in the next level ($l = 1$) are revealed, the second bit is revealed for all 2^{32} addresses. $\overline{U_i^{CA}}$ is the product of the number of nodes revealed in the anonymization tree and the number of bits revealed per node after i addresses have been compromised. This is given by:

$$\overline{U_i^{CA}} = \sum_{l=0}^{31} min(i, 2^l) \cdot 2^{32-l}. \tag{5}$$

The first portion is the same expression as $\overline{C_i^{CA}}$, and 2^{32-l} represents the number of bits revealed per compromised node on level l. Finally, U_i^{CA} is given by:

$$U_i^{CA} = U_0 - \overline{U_i^{CA}} \qquad i = 1, 2, \ldots, 2^{32}, \tag{6}$$

where U_0 and $\overline{U_i^{CA}}$ are given by Equations (4) and (5) respectively.

D. Security Analysis for Top Hashing

For top hashing (TH), we derive similar expressions for U_i^{TH} and C_i^{TH}. We denote t as the number of most significant bits of the address which are chosen to be hashed. Once again, we assume a complete anonymization tree with 32 levels and a worst case attack scenario. Each individual t-bit hash is considered as a single node in the anonymization tree.

C_0^{TH}, is the number of non-leaf nodes in the anonymization tree when no addresses have been compromised. The top-hash of the tree consists of 2^t nodes. Each node consists of a complete binary tree with $32 - t$ levels. C_0^{TH} is given by:

$$C_0^{TH} = (2^{32-t} - 1) \cdot 2^t + 2^t = 2^{32}. \tag{7}$$

We also assume that the attacker will first attempt to reveal the top hash table before proceeding to reveal internal nodes in the subtrees of the top hash. To reveal the top hash, 2^t addresses will be required. For the i^{th} compromised address, $min(i, 2^t)$ nodes will be revealed in the top hash table (we cannot reveal more than 2^t entries in the top-hash). The number of nodes revealed in the subtree is derived in a manner similar to that of Equation (2). The only difference here is that the summation extends from level t to level 31, (since the first t levels are protected by the top-hash). $\overline{C_i^{TH}}$ is given by:

$$\overline{C_i^{TH}} = min(i, 2^t) + \sum_{l=t}^{31} min(i, 2^l). \tag{8}$$

The expression for C_i^{TH} is the same as Equation (3) for C_i^{CA}.

U_i^{TH} can also be computed as shown in the previous section. We can derive an expression for $\overline{U_i^{TH}}$, the number of address bits revealed when i addresses are compromised, by multiplying Equation (8) with the number of bits revealed as a node in the anonymization tree is compromised. In the top-hash, t bits are revealed for 2^{32-t} addresses when a single hash is discovered. The product of these quantities with the first term of Equation (8) gives the number of bits revealed for the top-hash alone when i addresses are compromised. Similarly, 2^{32-l} bits are revealed for each node that is discovered in the l^{th} level ($t \leq l \leq 31$) of the address tree. The product of this quantity with the second term of Equation (8) gives us the number of bits revealed in the remainder of the address space. So $\overline{U_i^{TH}}$ is given by:

$$\overline{U_i^{TH}} = t \cdot 2^{32-t} \cdot min(i, 2^t) + \sum_{l=t}^{31} min(i, 2^l) \cdot 2^{32-l}. \tag{9}$$

The expression for U_i^{TH} is the same as Equation (6). U_0 is given by Equation (4).

A comparison between the number of revealed address bits under attack, U_i, for conventional anonymization (CA) and our proposed top hashing (TH) scheme is shown in Figure 10. U_i

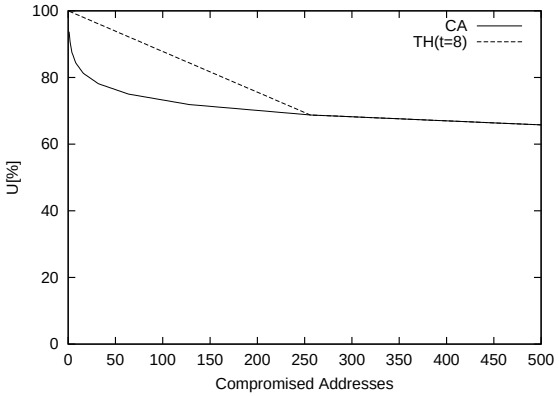

Fig. 10. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Top Hashing (TH) as a Function of the Number of Compromised Addresses.

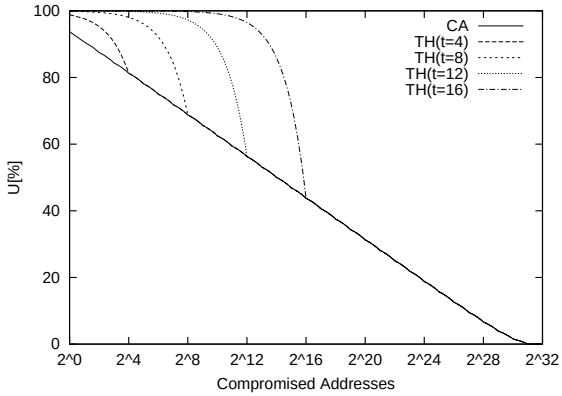

Fig. 11. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Top Hashing (TH) for Different Values of t as a Function of the Number of Compromised Addresses.

is expressed as a percentage of U_0. It can be observed that for the initial 256 compromised addresses top hashing reveals less information. In particular, with only one compromised address, $1/16^{th}$ of the address space in conventional anonymization is revealed, causing an immediate drop of U_1^{CA} to 93.75%. The slower decrease of U_i^{TH} in top hashing is a direct result of using a hash instead of a prefix-preserving mapping for the most significant 8 bits. However, once all 256 hash mappings of these 8 bits have been discovered, top hashing behaves the same way as conventional anonymization.

To show the impact of the parameter t, which specifies the size of the top hash, Figure 11 shows U_i^{TH} for a range of values for t. Note that the x-axis shows the number of compromised addresses on a logarithmic scale. U_i^{CA} and U_i^{TH} reach zero at 2^{31} compromised addresses, because all the information of the anonymization function has been discovered. Due to the prefix-preserving nature, the remaining 2^{31} addresses can be obtained by flipping the least significant bit. For larger values of t, top hashing performs very well and does not reveal very much information until the entire hash function has been discovered. The IP prefix distribution shown in Figure 6 indicates however that $t > 8$ would cause a loss of prefix-sharing relation in the anonymized trace. In cases where this is acceptable, larger values of t can provide better overall security.

E. Security Analysis for Subtree Replication

In this section, we derive expressions for U_i and C_i for both subtree replication (SR) and subtree replication with hashed mapping (SR-H). In SR, the replicated subtree is simply determined using a $mod(r)$ function. In SR-H, a cryptographic hash function is used to determine this mapping. Both analyses also use top hashing and are based on the results from the previous section.

As shown in Figure 8, the s lower bits of the address are considered for replication. Additionally, the subtree replication factor, r, gives the number of unique subtrees that are replicated. For example, $s = 8$ and $r = 256$ means that the 8 bit deep subtrees at the end of the address tree are replicated. There are 2^{24} such subtrees, out of which the first 256 subtrees are unique. Thus, each subtree is replicated 2^{16} times.

An expression for C_i^{SR} can be obtained as done previously by computing $\overline{C_i^{SR}}$, the number of nodes of the anonymization tree that are revealed as i addresses are compromised. This is given by:

$$\overline{C_i^{SR}} = min(i, 2^t) + \sum_{l=t}^{31-s} min(i, 2^l) + \sum_{l=32-s}^{31} min(i, r \cdot 2^{l-32+s}) \frac{2^{32-s}}{r}, \quad (10)$$

where the first term represents the number of nodes revealed in the top-hash portion and the second term represents the number of nodes revealed in the main portion of the anonymization tree. Note that the second term is summed from t to $(31 - s)$ since $(t \leq l \leq 31 - s)$ for the middle level. The last term in the equation expresses the number of nodes revealed in the subtree portion of the anonymization tree. For r unique subtrees, a total of $\sum_{l=32-s}^{31} min(i, r \cdot 2^{l-32+s})$ nodes are revealed. This causes the same number of nodes to be revealed in the remaining $\frac{2^{32-s}}{r}$ subtrees as well since the same subtree is reused. The expression for C_i^{SR} is the same as Equation (3). C_0^{SR} is specified by Equation (7) (replicated nodes are counted individually).

U_i^{SR} can be obtained from Equation (6). U_0 is the same as the previous cases and is given by Equation (4). To compute $\overline{U_i^{SR}}$, we can use Equation (10) and multiply each term of the equation with the number of bits revealed by compromising the nodes. For the top hash, the number of bits compromised is $t \cdot 2^{32-t}$. For the remainder of the anonymization tree, the number of bits revealed is 2^{32-l}. The expression for $\overline{U_i^{SR}}$ is:

$$\overline{U_i^{SR}} = t \cdot 2^{32-t} \cdot min(i, 2^t) + \sum_{l=t}^{31-s} min(i, 2^l) \cdot 2^{32-l} + \sum_{l=32-s}^{31} min(i, r \cdot 2^{l-32+s}) \cdot 2^{32-l} \frac{2^{32-s}}{r} \quad (11)$$

Figure 12 compares U_i for subtree replication (including top hashing) and conventional anonymization. The height of the

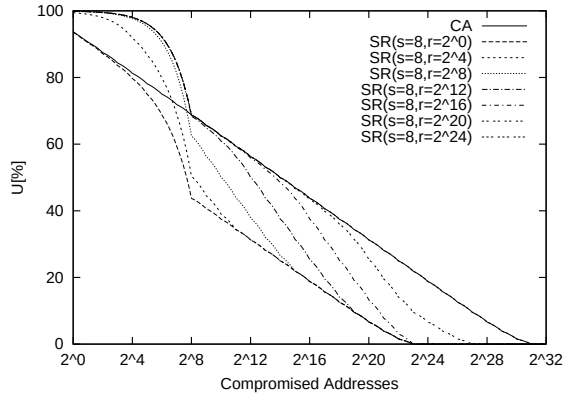

Fig. 12. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Subtree replication with top hashing (SR) for Different Values of Unique Subtrees r as a Function of the Number of Compromised Addresses. Top hashing is set to $t = 8$ and the subtree height is $s = 8$.

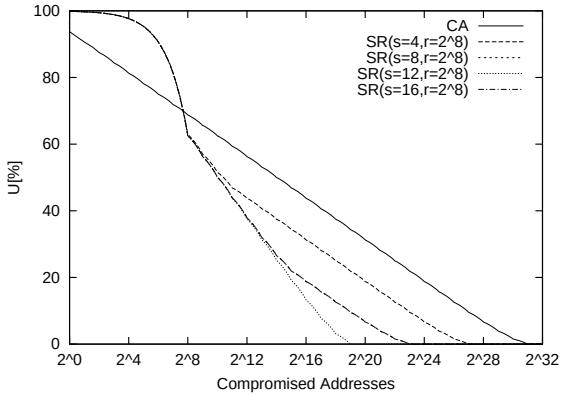

Fig. 13. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Subtree Replication with Top Hashing (SR) for Different Values of the Subtree Height s as a Function of the Number of Compromised Addresses. Top hashing is set to $t = 8$ and the subtree replication is $r = 2^8$.

subtrees is $s = 8$. The replication parameter r that determines the number of unique subtrees is varied from a single subtree ($r = 2^0$) to all unique subtrees ($r = 2^{24}$). The single unique subtree is the lower bound of the performance and performs strictly worse than CA. The larger the value of r, the closer SR comes to CA (for $i > 2^t$) and TH shown in Figure 10. For $r = 2^{16}$, SR shows comparable security performance to CA and TH for up to 2^{16} compromised addresses while using only $1/256^{th}$ of the number of subtrees of TH.

The impact of the subtree height, s, is shown in Figure 13. The height is varied from 4 to 16 bits with a constant replication of $r = 256$. Smaller subtrees reveal less information due to replication and thus perform better. The effect to top hashing on SR can be identified by the "hump" in the plots of Figure 12 and Figure 13. This shows that top hashing increases security by reducing U at a slower rate when compared to conventional anonymization. If top hashing were to be removed from SR, then the reduction in U for all SR plots will either keep up with or be less than the reduction in U for the CA plot until 256 addresses are compromised (since $t = 8$). When more addresses are compromised, the graphs will be identical to those shown in Figure 12 and Figure 13.

F. Security Analysis for Subtree Replication with Hashed Mapping

The major drawback of SR is that once the unique subtree has been completely compromised (by compromising its 2^s address mappings), all copies of the subtree are also compromised. Subtree Replication with Hashed Mapping (SR-H) addresses this problem. Instead of having a simple $mod(r)$ function to determine which subtree is used when traversing the anonymization tree, SR-H uses a cryptographic hash to choose the subtree. This makes it more difficult for an attacker to determine all address mappings.

The worst case attack for SR-H requires that each unique subtree is revealed (assuming the attacker can guess which these are) by compromising all its address mappings. Then the attacker tries to compromise the duplicate subtrees with the following method. By compromising a single address in a duplicated subtree, the attacker obtains a mapping between one original and one anonymized address. Since all unique subtrees are already revealed, a simple comparison of the obtained mapping with all subtrees yields the unique subtree that is used. It is unlikely (but possible) that there are a large number of different subtrees that contain the same mapping. If this is the case, the attacker can compromise more addresses to obtain the unique subtree. Since we consider the worst case attack, we assume the subtree can be identified by the first compromised address.

Thus, with SR-H, the attacker needs one extra compromised address per subtree to reveal the entire data structure. While this might seem easily achievable, it does contribute to a significant effort for configurations where the number of subtrees is large.

The expression for $\overline{U_i^{SR-H}}$ can be derived from Equation (11) and is given by:

$$\begin{aligned}\overline{U_i^{SR-H}} \quad = \quad & t \cdot 2^{32-t} \cdot min(i, 2^t) + \sum_{l=t}^{31-s} min(i, 2^l) \cdot 2^{32-l} + \\ & \sum_{l=32-s}^{31} min(i, r \cdot 2^{l-32+s}) \cdot 2^{32-l} + \\ & min(max(s \cdot 2^s \cdot (i - r \cdot 2^{s-1}), 0), \\ & s \cdot (2^{32} - 2^s \cdot r)) \end{aligned} \tag{12}$$

The third term represents the number of bits revealed by revealing nodes in the subtree. Since a hash is used, bits in the duplicated subtrees are not revealed when nodes in a particular subtree are compromised. The third term is not multiplied by $\frac{2^{32-s}}{r}$ (which is the number of repeated subtrees) to account for this fact. The fourth term accounts for the fact that an additional address needs to be compromised in order to reveal a duplicated subtree. We assume that the attacker is capable of revealing the r unique subtrees before compromising the remaining duplicated subtrees. This means that the attacker will need the first $r \cdot 2^{s-1}$ addresses to reveal the unique subtrees. Subsequently, the attacker can reveal one entire subtree ($s \cdot 2^s$ bits) with a single address. The max

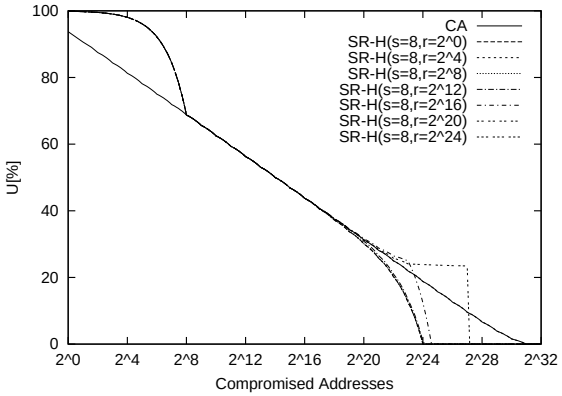

Fig. 14. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Subtree Replication with Hashed Mapping and Top Hashing (SRH) for Different Values of Unique Subtrees r as a Function of the Number of Compromised Addresses. Top hashing is set to $t = 8$ and the subtree height is $s = 8$.

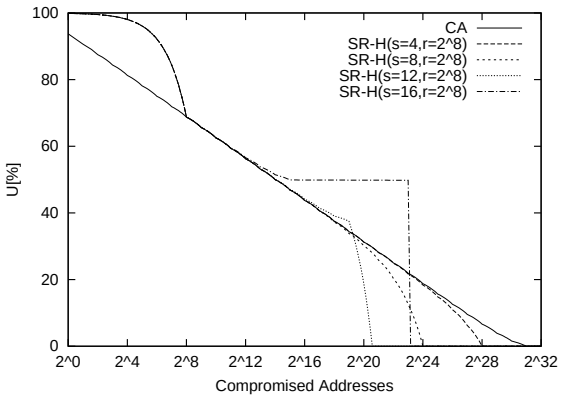

Fig. 15. Percentage of Unknown Bits in Address Space (U_i) for Conventional Anonymization (CA) and Subtree Replication with Hashed Mapping and Top Hashing (SRH) for Different Values of the Subtree Height s as a Function of the Number of Compromised Addresses. Top hashing is set to $t = 8$ and the subtree replication is $r = 2^8$.

function is required since the number of bits revealed will be negative when $i < r \cdot 2^{s-1}$. We have a total of $s \cdot 2^{32}$ bits in the subtrees out of which $2^s \cdot r \cdot s$ bits have been revealed in the unique subtrees. The min function is needed to ensure that the number of bits revealed by the fourth term does not exceed the maximum number of bits that can be revealed in the repeated subtrees which is $(s \cdot 2^{32}) - (2^s \cdot r \cdot s)$.

The results for U_i^{SR-H} with $s = 8$ are shown in Figure 14 as compared to CA. It can be observed that SR-H now tracks the security performance of CA (for $i > 2^t$) and TH for up to 2^{20} compromised addresses. For $i > 2^{20}$, whole subtrees can be compromised quickly with the attack described above. A similar trend can be seen in Figure 15, where the subtree height is varied. For certain configurations (e.g., $s = 16$ and $r = 2^8$), U_i^{SR-H} is *above* U_i^{TH} for large i (which is not a result of top hashing). This is due to a different attack strategy in SR-H than in CA. In SR-H, it is more desirable to first compromise the unique subtrees *completely* (the flat part of the plot in Figure 15), and then discover the duplicate subtrees quickly. This leads to U_i^{SR-H} leveling off and then dropping steeply.

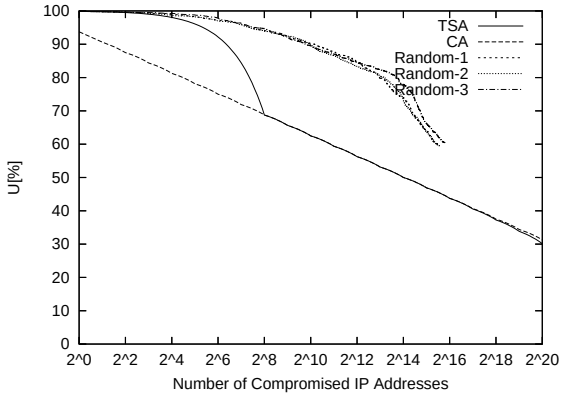

Fig. 16. Percentage of Unknown Bits in Address Space(U_i) as a Function of the Number of Compromised Addresses for Subtree Replication with Hashed Mapping and Top Hashing (TSA), Conventional Anonymization (CA), and 3 Random Passive Attacks. Top hashing is set to $t = 8$, subtree height is $s = 8$ and subtree replication is $r = 2^8$.

G. Passive Attacks

In a passive attack, the attacker is assumed to have obtained a trace in which the IP addresses have been anonymized in a prefix preserving manner. The attacker attempts to reveal mappings between the raw and anonymized IP addresses, but is limited to the addresses that actually occur in the trace. Thus, a worst case attack is not possible. This type of attack is quite complex to analyze. Instead, we have simulated such an attack on a real trace obtained from the main Internet access link at the University of Massachusetts.

Figure 16 shows the results of such an attack. The worst case shows the decrease in U_i for the active attack analyzed previously. This graph was obtained from Figure 14 with subtree replication and top hashing (SRH) and with parameters $t = 8$, $s = 8$, and $r = 2^8$. Three random attacks are simulated. In each attack, addresses to compromise are chosen at random from the pool of available addresses in the trace. It can be seen that the decrease in U for all 3 random attacks is less than the worst case indicating that passive attacks are significantly less effective than worst case attacks. Eventually, as more IP addresses are compromised, we would expect both graphs to merge. We were unable to simulate the attack till this point due to a lack of unique IP addresses in the anonymized trace.

H. Analysis Summary

The security analysis for the different anonymization algorithms yields several important observations:

- Top hashing improves the security of the anonymization for small scale attacks ($i < 2^t$) and never decreases the security below that of CA.
- Subtree replication causes a decrease in security because a compromised address reveals bits in all duplicated subtrees, too. For larger numbers of unique subtrees, subtree replication approximates the performance of TH and CA.
- Subtree replication with hashed mapping performs similar to TH and CA even for smaller numbers of unique

subtrees. Only for very large-scale attacks ($i > 2^{20}$), address bit mappings are revealed more quickly.
- Passive random attacks do not provide as much information as the worst case active attacks. A passive attack requires more IP addresses to be compromised to reveal the same amount of information.

This shows that SR-H achieves the performance of TH and CA while using less memory in most cases. The following section addresses the issue of memory requirements and how a good balance between security and memory consumption can be found.

V. Security-Space Tradeoff

There is an inherent tradeoff between the amount of memory that is required to store the anonymization data structure and how resistant the anonymization algorithm is against attacks. In TSA, there are three parameters that can be adjusted for this purpose: top hash size t, subtree size s, and the number of unique subtrees r. In this section, we derive an expression for the memory size required for a given set of parameters and define a metric that describes the security performance. We explore the tradeoffs and identify the best combination of parameters.

A. Space Requirements

The space requirement, S, for different anonymization algorithms is determined by the number of nodes in the anonymization trees and the number and size of the hash tables used. Since each node only stores the information if the corresponding address bit should be flipped or not, a single bit of information is sufficient. A complete binary tree of height h has $2^h - 1$ internal nodes and thus requires that many bits of storage. Hash tables that map 2^n entries to 2^m values require $2^n \cdot m$ bits of storage.

For conventional anonymization, the space requirement is $S^{CA} = 0$ because all values are computed online. If they were to be stored for reuse to reduce the overall processing cost then the maximum amount of storage would be a complete tree:

$$S^{CA}_{max} = 2^{32} - 1. \tag{13}$$

This is equivalent to a precomputed anonymization tree. Top hashing replaces the top t levels of the tree with a hash function, which increases the overall space consumption to:

$$S^{TH} = 2^t \cdot t + 2^{32} - 2^t \tag{14}$$

Thus the overhead from top hashing is $S^{TH} - S^{CA}_{max} = 2^t \cdot (t-1) + 1$. When subtree replication is added, the space requirement consists of the smaller anonymization tree of height $31 - s$ (plus top hashing overhead) and the r unique subtrees of height $s - 1$:

$$S^{SR} = 2^t \cdot (t-1) + 2^{32-s} + r \cdot (2^s - 1). \tag{15}$$

This does not consider hashed mapping, which requires an additional hash table to map between the 2^{32-s} nodes and the r unique subtrees. Thus, the space requirement for subtree replication with hashed mapping is:

$$S^{SR-H} = 2^t \cdot (t-1) + 2^{32-s} + r \cdot (2^s - 1) + (2^{32-s}) \cdot \lceil \log_2 r \rceil. \tag{16}$$

S^{SR} in MB		s					
		4	8	12	16	20	24
r	2^0	32.00	2.00	0.13	0.02	0.13	2.00
	2^4	32.00	2.00	0.13	0.13	2.00	32.00
	2^8	32.00	2.01	0.25	2.01	32.00	512.00
	2^{12}	32.01	2.12	2.12	32.01	512.00	–
	2^{16}	32.12	3.99	32.12	512.00	–	–
	2^{20}	33.88	33.88	512.00	–	–	–
	2^{24}	62.00	512.00	–	–	–	–
	2^{28}	512.00	–	–	–	–	–

TABLE I
Memory Requirement for Subtree Replication (incl. Top Hash) in MB. Invalid parameter combinations are indicated by '–'.

S^{SR-H} in MB		s					
		4	8	12	16	20	24
r	2^0	32.00	2.00	0.13	0.02	0.13	2.00
	2^4	160.00	10.00	0.63	0.16	2.00	32.00
	2^8	288.00	18.01	1.25	2.07	32.00	512.00
	2^{12}	416.01	26.12	3.62	32.10	512.01	–
	2^{16}	544.12	35.99	34.12	512.13	–	–
	2^{20}	673.88	73.88	514.50	–	–	–
	2^{24}	830.00	560.00	–	–	–	–
	2^{28}	1408.00	–	–	–	–	–

TABLE II
Memory Requirement for Subtree Replication with Hashed Mapping (incl. Top Hash) in MB. Invalid parameter combinations are indicated by '–'.

Tables I and II show the memory requirements for SR and SRH which were calculated using the above formulae. As discussed previously, t should be as large as possible, but cannot be larger than $t = 8$ due to the prefix distribution in the current Internet. For large values of r, the memory consumption of SRH exceeds that of SR due to the size of the tables required for hashed mapping. For the subtree height parameter, s, the memory requirement is smallest around $s = 12$, because the size between the original anonymization tree and the subtrees is balanced. For smaller s, the original subtree requires more memory, and for larger s, the replicated subtrees require more memory. The parameter r has little impact on the space requirement for $r < 2^s$. This is important to note as in this range r has significant impact on the security of the anonymization algorithm (see Figure 12). This leads to the question on which parameter combination of r and s yields the best security for a given size of memory.

B. Resistance to Attack

To make the comparison between different parameter combinations tractable, we introduce a security metric, R_p, that reduces the functions of U_i shown in Section IV to a single value. We call R_p "resistance to attack" and define it as the number of addresses that need to be compromised to reduce U_i to p percent of U_0:

$$R_p = min(i) | \frac{U_i}{U_0} \leq p. \tag{17}$$

The parameter p specifies the desired "security level." If p is large (e.g., $p = 90\%$), then $R_{90\%}$ is small, because only few addresses need to be compromised to reduce U_i to 90% of U_0.

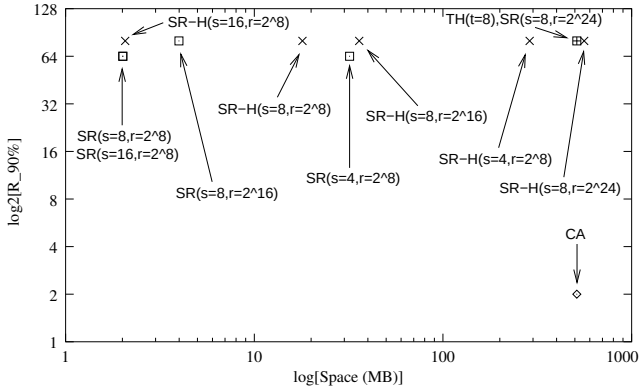

Fig. 17. Space-Security Tradeoff for Different Anonymization Algorithms (for $R_{90\%}$).

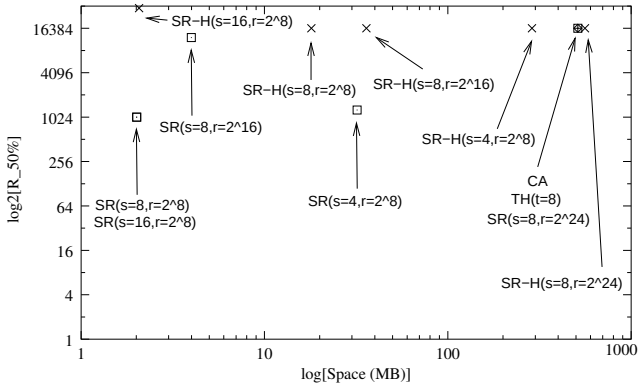

Fig. 18. Space-Security Tradeoff for Different Anonymization Algorithms (for $R_{50\%}$).

C. Optimization

The tradeoff between space and security can be explored by comparing S and R. For R, a security level needs to be specified and for this evaluation we are considering $R_{90\%}$ and $R_{50\%}$. $R_{90\%}$ corresponds to a very high security level, where only 10% of compromised address bits are tolerable. $R_{50\%}$ is a low security level, where half the address space is revealed.

Figures 17 and 18 show the results for a variety of parameters of r and s. For $R_{90\%}$, conventional anonymization performs very poorly, because two compromised addresses reveal more than 10% of U. All TSA configurations perform significantly better due to the use of top hashing. The space consumption for convention anonymization is assumed to be S^{CA}_{max}, but could be also be considered to be $S^{CA} = 0$ (which cannot be shown on this graph). SR and SR-H perform comparably, with SR($s = 8$, $r = 2^8$), SR($s = 16$, $r = 2^8$), and SR-H($s = 16$, $r = 2^8$) performing best. All three configurations require only 2MB of memory and show a resistance to attack of $R_{90\%} > 64$.

For $R_{50\%}$, CA shows high resistance to attack (with the same space constraint as discussed before). SR-H($s = 16$, $r = 2^8$) shows a slightly higher resistance, which is due to the different attack strategy against hashed mapping discussed in Section IV. The best configurations for TSA are SR-H($s = 16$, $r = 2^8$) with 2MB memory and ($s = 8$, $r = 2^{16}$) with 4MB memory. For both algorithms, $R_{50\%} > 10,000$.

The results show that TSA performs better than existing anonymization algorithms for high security levels and performs comparably at lower security levels. TSA requires only 2–4MB of memory to achieve this performance. For conventional anonymization the amount of memory (if precomputation is used) can be as large as 512MB.

VI. Implementation

We have implemented prototypes of TSA and conventional anonymization to evaluate the processing performance aspects of these algorithms in detail. For conventional anonymization we have implemented both an online version (as performed in [17]) and a precomputed version (with a fully precomputed anonymization tree). For TSA, we have implemented versions using SR (both with and without top hashing) and versions using SR-H (both with and without top hashing). To evaluate processing cost, we used the PacketBench system [12]. PacketBench simulates the functionality of a network processor while providing an easy to use environment for implementing packet processing functionality.

We compare the number of instructions executed to anonymize a single IPv4 address and the number of memory accesses necessary for all the algorithms. The results are shown in Table III. Conventional anonymization (CA) either has a high processing cost (for the online versions), or requires a large amount of memory to store the anonymization data structures (the precomputed version). When computed online, CA requires several hundred thousands instructions to process a packet. This is due to the computational complexity of performing numerous cryptographic operations (MD5 computations in this case). This significant overhead for MD5 computations has been observed previously in the context of IPv6 authentication [16] and deemed unsuitable for packet-level processing at high line speeds.

In contrast, all the configurations of TSA in Table III require less than 500 instructions per address. This reduction of processing cost by three orders of magnitude makes it possible to implement TSA in high-performance measurement nodes. The memory accesses in TSA are between 25 or 35 depending on the configuration. These can be performed in parallel, because the location of all nodes in the tree are known beforehand.

Top hashing has three effects on all algorithms. It reduces processing cost since the computation for the bits of the IP address covered by the top hash are replaced by a single lookup to the top hash table. It increases memory requirements by a small amount since we need to store the top hash table. Finally, it increases the security of all algorithms particularly for $R_{99\%}$ and $R_{90\%}$ levels.

Overall, Table III shows that TSA is able to solve both problems with conventional anonymization - (1) the high processing cost of online conventional anonymization and (2) the large memory requirements of precomputed conventional anonymization, while maintaining comparable security levels.

One concern of high-performance processing is the growing gap between processor clock speed and memory access time. Accesses to SRAM and SDRAM can take a significant amount

Algorithm	Instructions per IP Address	Memory Accesses per IP Address	Memory in MB	Security		
				$R_{99\%}$	$R_{90\%}$	$R_{50\%}$
CA (online, no top hashing)	247,078	0	0	1	2	16384
CA (online, with top hashing)	185,313	1	<0.01	8	80	16384
CA (precomputed, no top hashing)	460	32	512	1	2	16384
CA (precomputed, with top hashing)	348	25	512	8	80	16384
TSA SR($s = 8$,$r = 2^{16}$, no top hashing)	463	32	4.1	1	2	12288
TSA SR($s = 8$,$r = 2^{16}$, with top hashing)	357	25	4.1	8	80	12288
TSA SR-H($s = 16$,$r = 2^{8}$, no top hashing)	464	33	2.1	1	2	30720
TSA SR-H($s = 16$,$r = 2^{8}$, with top hashing)	359	26	2.1	8	80	30720

TABLE III

COMPARISON OF ALL PERFORMANCE ASPECTS OF TSA WITH CONVENTIONAL ANONYMIZATION (CA). TOP HASHING IS SET TO 8 BITS.

of time which can increase the overall processing cost of TSA. We have used a multibit trie implementation [15] to reduce the number of memory accesses necessary to traverse the anonymization tree. Our current prototype runs on an Intel IXP 2400 network processor [5] and requires 4 accesses to the anonymization data structures instead of 26. This results in a significant performance boost. The drawback is that a multibit trie requires more memory than a simple binary tree as some of the information is stored redundantly.

The final implementation issue is how to initialize the anonymization data structure that is used by TSA. This is done offline and only once (unless the anonymization keys are changed). The memory representing the anonymization tree can be filled with random bits since any combination of node colors yields a correct prefix preserving anonymization mapping. The top hash table needs to be filled with a sequence of bits that represent a one-to-one mapping of the t input bits. This can be achieved by enumerating all 2^t bit combinations and permutating t-bit chunks. The hash table for subtree replication with hashed mapping can be filled with pseudo-random bits as it represents a many-to-one mapping. In order to make the anonymization reproducible and consistent across multiple measurement nodes, all bit sequences for initializing the data structures are derived from keyed hash functions. The cryptographic strength of the entire anonymization algorithm depends on the cryptographic strength of these hash functions.

VII. EFFECT OF CACHING

We compare the performance of conventional anonymization (CA) with that of TSA. We assume that conventional anonymization is augmented with a caching scheme to avoid recomputation of anonymization mappings. For the performance of CA to match TSA, the following relation must be satisfied:

$$instructions_{CA} \cdot miss\_rate = instructions_{TSA} \qquad (18)$$

$instructions_{CA}$ and $instructions_{TSA}$ are the per packet processing cost in instructions for conventional anonymization and TSA. $miss\_rate$ is the total miss rate for the cache.

From Table III, we can see that conventional anonymization takes 247,078 instructions per address. We need to perform two anonymization operations per packet for a total of 494,156 instructions. TSA costs 359 instructions per anonymization operation for a total of 718 instructions per packet. This implies that $miss\_rate$ must be 0.00145 (or 0.145%) in order for caching to be effective.

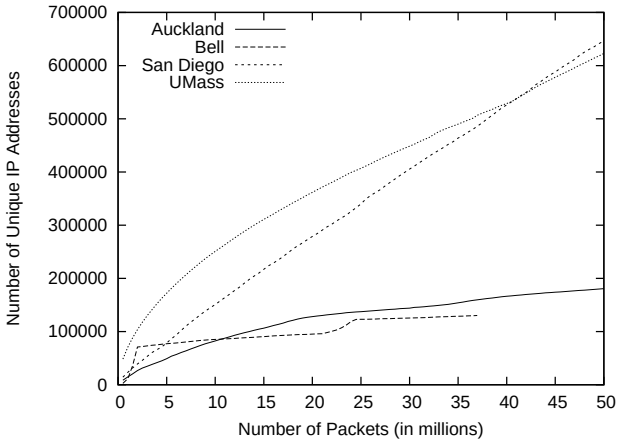

Fig. 19. Number of Unique IP Addresses Seen as a Function of Packets Encountered.

Figure 19 shows the variation in the number of unique IP addresses seen in the trace as a function of the number of packets encountered. This function is plotted for 4 traces. The San Diego, Auckland and Bell traces were obtained from [9]. The UMass trace was obtained from the main Internet access link at the University of Massachusetts. The anonymization mapping has to be calculated for all these addresses irrespective of the caching scheme used. The slope of these graphs represents the compulsory miss rate.

Figure 20 plots the variation in total miss rate for the San Diego trace. It can be seen that the miss rate is always greater than 0.00145 for different cache sizes varying from 16k entries to 1M entries. Even if we had an infinitely large cache, the miss rate is still not low enough to compensate for the high processing cost of conventional anonymization. The graph for an infinite entry cache represents the slope of the graphs shown in Figure 19. In Figure 19, it is worthwhile to note that for both the Bell and Auckland traces, the rate at which unique IP addresses appear is not very large and caching of anonymization mappings may work in such a situation. It is very clear that this is not the case for the San Diego and UMass traces.

From a systems point of view, it is desirable for new IP addresses to arrive at a constant rate. A burst of unique IP addresses will require a series of anonymization computations

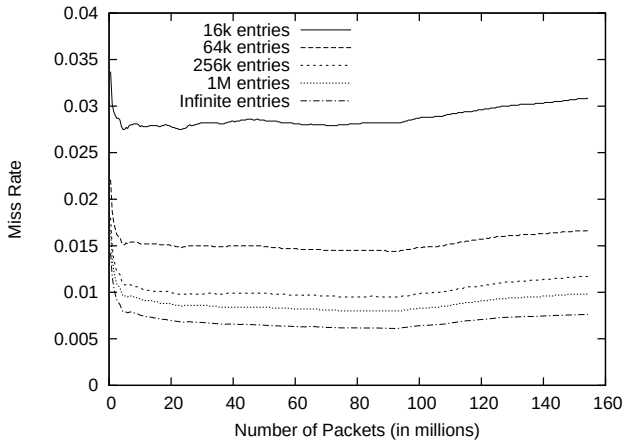

Fig. 20. Variation in Miss Rate as a Function of the Number of Packets Encountered for Different Cache Sizes. The San Diego trace was chosen for this analysis.

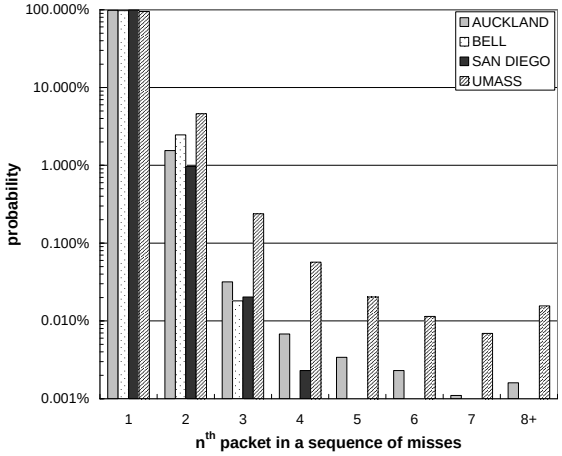

Fig. 21. Probability of a Miss being the n^{th} in a Sequence of Continuous Misses for Various Traces.

to be performed which can possibly reduce the ability of the measurement node to process packets at link speed. Figure 21 shows the probability of a new IP address (a miss in the cache) being the n^{th} in a sequence of continuous unique IP addresses (a sequence of misses in the cache) for all the four traces analyzed previously. Although a majority of the IP addresses appear individually, the UMass trace exhibits some bursty behavior. Due to the observed miss rates and burstiness, TSA presents a better choice than conventional anonymization with caching.

VIII. Summary and Conclusion

In this paper, we have proposed and analyzed a novel prefix-preserving anonymization algorithm called TSA. The key characteristics are precomputation, subtree replication with hashed mapping, and top hashing. The proposed algorithm outperforms the conventional anonymization mechanism (even while using caching) in terms of computation speed while providing comparable levels of security to attacks. The memory requirements for our algorithm are small enough to efficiently implement it on measurement systems. The various performance aspects of the two best configurations of TSA are shown in Table III and contrasted to conventional prefix-preserving anonymization. The significantly lower processing cost per IP address is the main criteria that makes TSA suitable for *online* anonymization of network traces and eliminates the need for post-processing. This is an important step towards a practical deployment of large-scale continuous measurement systems and ensuring that the privacy of network users is protected.

IX. Acknowledgments

This work was supported in part by NSF grant award ITR-CNS-0325868. The UMass trace was provided by Sharad Jaiswal and Yong Liu from the Department of Computer Science at the University of Massachusetts at Amherst. All other traces were obtained the from the NLANR Measurement and Network Analysis Group (NLANR/MNA) under NSF Cooperative Agreement No. ANI-0129677 (2002) and ANI-9807479 (1998).

References

[1] Advanced Network Technology Center, University of Oregon. *Route Views Project Page*, 2003. http://www.routeviews.org/.

[2] S. Bhattacharyya and S. Moon. Network monitoring and measurements: Techniques and experience. In *Tutorial at ACM Sigmetrics 2002*, Marina Del Rey, CA, June 2002.

[3] C. Cranor, Y. Gao, T. Johnson, V. Shkapenyuk, and O. Spatscheck. Gigascope: High performance network monitoring with an SQL interface. In *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*, page 623, Madison, WI, June 2002.

[4] C. Fraleigh, C. Diot, B. Lyles, S. B. Moon, P. Owezarski, D. Papagiannaki, and F. A. Tobagi. Design and deployment of a passive monitoring infrastructure. In *Passive and Active Measurement Workshop (PAM2001)*, Amsterdam, Netherlands, Apr. 2001.

[5] Intel Corporation. *Intel IXP2400 Network Processor*, 2004.

[6] G. Minshall. *TCPDPRIV*. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html, 1.1.10 edition, Aug. 1997.

[7] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*, Nov. 2001. FIPS 197.

[8] National Laboratory for Applied Network Research. *Active Measurement Project*, 2005. http://watt.nlanr.net/.

[9] National Laboratory for Applied Network Research - Passive Measurement and Analysis. *Passive Measurement and Analysis*, 2003. http://pma.nlanr.net/PMA/.

[10] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the ACM SIGCOMM Conference*, pages 339–351, Karlsruhe, Germany, August 2003.

[11] M. Peuhkuri. A method to compress and anonymize packet traces. In *Proc. of First ACM SIGCOMM Internet Measurement Workshop*, pages 257–260, San Francisco, USA, November 2001.

[12] R. Ramaswamy and T. Wolf. PacketBench: A tool for workload characterization of network processing. In *Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6)*, pages 42–50, Austin, TX, Oct. 2003.

[13] R. L. Rivest. The MD5 message digest algorithm. RFC 1321, Network Working Group, Apr. 1992.

[14] Surveyor Home Page. http://www.advanced.org/surveyor/.

[15] D. E. Taylor, J. W. Lockwood, T. Sproull, J. S. Turner, and D. B. Parlour. Scalable ip lookup for programmable routers. In *Proc. of the Twenty-First IEEE Conference on Computer Communications (INFOCOM)*, pages 562–571, New York, NY, jun 2002.

[16] J. D. Touch. Performance analysis of MD5. In *Proc. of ACM SIGCOMM 95*, pages 77–86, Cambridge, MA, aug 1995.

[17] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proc. of 10th IEEE International Conference on Network Protocols (ICNP'02)*, pages 280–289, Paris, France, Nov. 2002.