Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme*

Jun Xu   Jinliang Fan   Mostafa H. Ammar
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{jx,jlfan,ammar}@cc.gatech.edu

Sue B. Moon
Sprint ATL
1 Adrian Court
Burlingame, CA 94010
sbmoon@sprintlabs.com

August 18, 2002

Abstract– Real-world traffic traces are crucial for Internet research, but only a very small percentage of traces collected are made public. One major reason why traffic trace owners hesitate to make the traces publicly available is the concern that confidential and private information may be inferred from the trace. In this paper we focus on the problem of anonymizing IP addresses in a trace. More specifically, we are interested in *prefix-preserving anonymization* in which the prefix relationship among IP addresses is preserved in the anonymized trace, making such a trace usable in situations where prefix relationships are important. The goal of our work is two fold. First, we develop a cryptography-based, prefix-preserving anonymization technique that is provably as secure as the existing well-known TCPdpriv scheme, and unlike TCPdpriv, provides consistent prefix-preservation in large scale distributed setting. Second, we evaluate the security properties inherent in all prefix-preserving IP address anonymization schemes (including TCPdpriv). Through the analysis of Internet backbone traffic traces, we investigate the effect of some types of attacks on the security of any prefix-preserving anonymization algorithm. We also derive results for the optimum manner in which an attack should proceed, which provides a bound on the effectiveness of attacks in general.

Keywords– prefix-preserving address anonymization, TCPdpriv, cryptography-based anonymization, traffic measurement, security, privacy.

1 Introduction

Real-world Internet traffic traces are crucial for network research such as workload characterization, traffic engineering, web performance, and more generally network performance analysis and simula-

*A preliminary version of the paper will be presented at the 10th IEEE International Conference on Network Protocols, to be held in Paris, France, from November 12th to 15th, 2002.

tion. However, only a tiny percentage of traffic traces collected are made public (e.g., by NLANR/MOAT Network Analysis Infrastructure (NAI) project [1] and ACM ITA project [2]) for research purposes. One major reason why ISPs or other traffic trace owners hesitate to make the traces publicly available is the concern that the confidential (commercial) and private (personal) information regarding the senders and receivers of packets may be inferred from the trace. In cases where a trace has been made publicly available, the trace is typically subjected to an anonymization process before being released.

A straightforward approach to anonymizing a packet trace is to map each distinct IP address appearing in the trace to a random 32-bit "address". The only requirement is that this mapping be one-to-one. Anonymity of the IP addresses in the original trace is achieved by not revealing the random one-to-one mapping used in anonymizing a trace. Such anonymization, however, results in the loss of the prefix relationships among the IP addresses and renders the trace unusable in situations where such relationship is important (e.g., routing performance analysis, or clustering of end-systems [3]). It is, therefore, highly desirable for the address anonymization to be *prefix preserving*. That is, if two original IP addresses share a k-bit prefix, their anonymized mappings will also share a k-bit prefix. One approach to such prefix preserving anonymization is adopted in *TCPdpriv* developed by Greg Minshall [4].

In this work we first formally characterize prefix-preserving anonymization functions by showing that the set of such functions follow a canonical form. TCPdpriv can be viewed as a table-based approach that generates a function randomly from this set. It may produce inconsistent prefix-preserving anonymization (i.e., same original prefix mapped into different anonymized prefixes) when used independently on more than one trace. We develop an alternative cryptography-based, prefix-preserving anonymization technique to address this issue, and prove rigorously that the proposed technique maintains the same level of anonymity as TCPdpriv.

Second, we are interested in analyzing the security properties inherent in prefix-preserving IP address anonymization in general (whether using TCPdpriv or the proposed scheme). We aim to understand its susceptibility to attacks that may reveal some IP address mappings (e.g., [5]). Through analysis of real-world IP traffic traces, we investigate the effect of some types of attacks on the security of the prefix-preserving anonymization process. In the process, we derive some results pertaining to the optimum manner in which an attack should proceed with the goal of understanding the bounds on the performance of attacks in general.

Although our results can be used to analyze the effect of attacks on an anonymized trace, we believe that it is outside the scope of our work to make any conclusions regarding how "safe" it is to release an anonymized trace. We stress that our work constitutes a scientific endeavor, intended to explore the potential and limits of prefix-preserving anonymization as a way to simultaneously satisfy the needs of network researchers and the concerns of trace owners. We realize that the decision to release data, even in anonymized form, is affected by many non-technical issues. Our role is to provide a technical foundation for such decision making.

The rest of this paper is organized as follows. In Section 2 we introduce our result regarding the canonical form of a prefix-preserving anonymization scheme. We also describe the operation of TCPdpriv and present our own cryptography-based scheme. In Section 3 we describe cryptographic and semantic attacks; two forms of attacks that may potentially be used to defeat an anonymization scheme. Section 4 proves the immunity of our cryptography-based anonymization scheme from cryptographic attacks. In Section 5, we develop a framework for evaluating the effects of semantic attacks on anonymization schemes in general (including TCPdpriv and our cryptography-based scheme). We then use the framework to derive numerical results demonstrating the effects of certain attacks on real-world traces. The paper is concluded in Section 6.

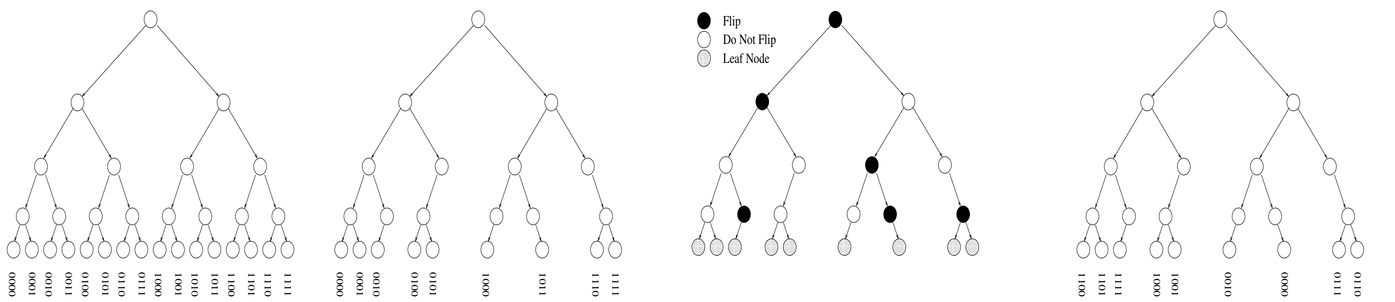
2 Prefix-Preserving Anonymization Schemes

We begin this section with a formal definition of prefix-preserving anonymization.

Definition 1 (Prefix-preserving Anonymization[1]) *We say that two IP addresses $a = a_1a_2\cdots a_n$ and $b = b_1b_2\cdots b_n$ share a k-bit prefix ($0 \leq k \leq n$), if $a_1a_2\cdots a_k = b_1b_2\cdots b_k$, and $a_{k+1} \neq b_{k+1}$ when $k < n$[2]. An anonymization function F is defined as a one-to-one function from $\{0,1\}^n$ to $\{0,1\}^n$. An anonymization function F is said to be prefix-preserving, if, given two IP addresses a and b that share a k-bit prefix, $F(a)$ and $F(b)$ also share a k-bit prefix.*

It is useful for our future analysis to consider a geometric interpretation of this form of anonymization. We first note that the entire set of possible distinct IPv4 addresses can be represented by a complete binary tree of height 32. The set of distinct addresses present in an unanonymized trace can be represented by a subtree of this complete binary tree where each address is represented by a leaf. We call

[2]For all known packet traces, $n = 32$, as an IPv4 address has four bytes.



(a) address space (b) original address tree (c) anonymization function (d) anonymized address tree

Figure 1: Address Trees and Anonymization Function

this the *original address tree*. Each node in this original address tree (excluding the root node) corresponds to a bit position, indicated by the height of the node, and a bit value, indicated by the direction of the branch from its parent node. Figure 1(a) shows a complete binary tree (using 4-bit addresses for simplicity) and Figure 1(b) shows an original address tree.

A prefix-preserving anonymization function can be viewed as specifying a binary variable for each non-leaf node (including the root node) of the original address tree. This variable specifies whether the anonymization function “flips” this bit or not. Applying the anonymization function results in the rearrangement of the original address tree into an *anonymized address tree*. Figure 1(d) shows the anonymized address tree resulting from the anonymization function shown in Figure 1(c). Note that an anonymization function will, therefore, consist of at least I binary variables if the original address tree has I non-leaf nodes.

Although what we have presented is clearly a method for prefix-preserving anonymization, it is not immediately obvious that this is the only method. In the following theorem, we prove that this is indeed the only method.

Theorem 1 (Canonical Form Theorem) *Let f_i be a function from $\{0,1\}^i$ to $\{0,1\}$, for $i = 1, 2, \cdots, n-1$ and f_0 is a constant function. Let F be a function from $\{0,1\}^n$ to $\{0,1\}^n$ defined as follows. Given $a = a_1 a_2 \cdots a_n$, let*

$$F(a) := a'_1 a'_2 \cdots a'_n \tag{1}$$

where $a'_i = a_i \oplus f_{i-1}(a_1, a_2, \cdots, a_{i-1})$, and $\oplus$ stand for the exclusive-or operation, for $i = 1, 2, \cdots, n$. We claim that *(a) F is a prefix-preserving anonymization function and*

(b) A prefix-preserving anonymization function necessarily takes this form.

Proof: (a) Suppose two raw addresses $a = a_1a_2 \cdots a_n$ and $b = b_1b_2 \cdots b_n$ share a k-bit prefix; that is, $a_1a_2 \cdots a_k = b_1b_2 \cdots b_k$, and, if $k < n$, $a_{k+1} = \overline{b_{k+1}}$ (or equivalently $\overline{a_{k+1}} = b_{k+1}$). Then for $i = 1, 2, \cdots, k$,

$$
\begin{aligned}
a'_i &= a_i \oplus f_{i-1}(a_1, a_2, \cdots, a_{i-1}) \\
&= b_i \oplus f_{i-1}(b_1, b_2, \cdots, b_{i-1}) \\
&= b'_i
\end{aligned}
$$

and, if $k < n$,

$$
\begin{aligned}
a'_{k+1} &= a_{k+1} \oplus f_k(a_1, a_2, \cdots, a_k) \\
&= \overline{b_{k+1}} \oplus f_k(b_1, b_2, \cdots, b_k) \\
&= \overline{b'_{k+1}}
\end{aligned}
$$

(b) This is equivalent to proving that given any prefix preserving function F, we can find corresponding $f_i, i = 0, 1, \cdots, n-1$ in the above form. Given any F and any $i, 0 \leq i \leq n-1$, we define f_i as follows. Given any i-bit sequence $a_1a_2 \cdots a_i$, we append an arbitrary $n - i$ bit sequence $a_{i+1}a_{i+2} \cdots a_n$ to it. Then we define $f_i(a_1, a_2, \cdots, a_i) := c$, where c is the $(i+1)$-th bit of $F(a_1a_2 \cdots a_n) \oplus a_{i+1}$. It remains to show that f_i is well-defined: different choices of $a_{i+1}, a_{i+2}, \cdots, a_n$ lead to the same c value. Given another sequence $b_{i+1}b_{i+2}, \cdots, b_n$, we show $c = c'$, where c' is computed as $(i+1)$-th bit of $F(a_1a_2 \cdots a_ib_{i+1} \cdots b_n) \oplus b_{i+1}$. We only need to discuss following two cases:

(1) When $a_{i+1} = b_{i+1}$, $F(a_1a_2 \cdots a_ia_{i+1} \cdots a_n)$ and $F(a_1a_2 \cdots a_ib_{i+1} \cdots b_n)$ should have the same $(i+1)$-th bit (denoted as d) since F is prefix-preserving. So $c = d \oplus a_{i+1} = d \oplus b_{i+1} = c'$.

(2) Similarly, we can show $c = c'$ when $a_{i+1} = \overline{b_{i+1}}$. $\square$

Remark: Note that there is a natural one-to-one mapping between the canonical form of a prefix-preserving anonymization function and its graphical representation. Each node in an anonymization tree (see figure 1), as represented by its prefix $a_1a_2 \cdots a_k$, will be labeled "flip" or "no flip", when $f(a_1a_2 \cdots a_k) = 1$ or 0, respectively.

In the following, we describe TCPdpriv, an existing traffic anonymization tool that, among other things, allows the prefix-preservation anonymization of IP addresses. We describe how TCPdpriv implements prefix-preserving anonymization and identify its properties. We then discuss our cryptography-based prefix-preserving anonymization algorithm that possesses additional functionality. Finally, we define metrics for the level of security that is constrained by the prefix-preserving requirement and show that both TCPdpriv and our scheme achieve this same level of security.

2.1 TCPdpriv and Its Properties

TCPdpriv's implementation of the prefix-preserving translation of IP addresses is table-based: it stores a set of <raw, anonymized> binding pairs of IP addresses to maintain the consistency of the anonymization. When a new raw IP address a needs to be anonymized, it is first compared with all the the raw IP addresses inside the stored binding pairs for the longest prefix match. Suppose the binding pair whose raw address has longest prefix match with $a = a_1a_2...a_n$ is <x, y> (let $x = x_1x_2\cdots x_n$ and $y = y_1y_2\cdots y_n$), and the length of the match is k. Suppose a is anonymized to $b = b_1b_2\cdots b_n$. Then $b_1b_2\cdots b_kb_{k+1} = y_1y_2\cdots y_k\overline{y_{k+1}}$ and $y_{k+2}y_{k+3}\cdots y_n = rand(0, 2^{n-k-1} - 1)$, where $rand(x, y)$ generates a pseudorandom (not required to be cryptographically strong) number between x and y. If $a \neq x$, <a,b> will be then added to the binding table.

When a binary trie data structure is used, the search for the longest prefix match has the cost of $O(n)$, where n is the number of bits in the address. The memory requirement of the algorithm is $O(M)$, where M is the number of binding pairs stored. We refer readers to the source code of TCPdpriv [4] for the actual data structure and algorithm.

Despite the elegance and simplicity of the TCPdpriv implementation, it does not facilitate the parallel and distributed (yet consistent) anonymization of traffic traces:

- TCPdpriv does not allow distributed processing of different traces simultaneously. Like other prefix-preserving anonymization functions, TCPdpriv can be mapped to the canonical form shown in Theorem 1. For TCPdpriv, the functions $\{f_i\}_{0\leq i\leq n-1}$ in the canonical form are trace-dependent: they are determined by the raw IP addresses and the relative order in which they appear in a trace. Therefore, a raw address appearing in different traces may be mapped to different anonymized addresses by TCPdpriv, hence the inconsistency[3]. However, there is a real need for simultaneous (yet consistent) anonymization of traffic traces in different sites, e.g., for taking a snapshot of the Internet. It would be very cumbersome if hundreds of traces have to be gathered first and then anonymized in sequence.

- A large trace (e.g., terabytes) may be collected for a high-speed link for a long period of time. For the same reason discussed above, TCPdpriv does not allow a large trace file to be broken down into pieces and processed in parallel consistently.

2.2 A Cryptography-based Scheme

We have designed an algorithm that addresses the aforementioned properties of TCPdpriv by deterministicly mapping raw addresses to anonymized addresses based on a relatively small key (compared to the M-entry binding table), which facilitates distributed and parallel anonymization of traffic traces. We show that the algorithm is provably secure up to the level of security a prefix-preserving anonymization could possibly deliver.

Based on the canonical form in Theorem 1, our cryptography-based scheme is defined as instantiating functions f_i in (1) with cryptographically strong stream ciphers or block ciphers as follows:

$$f_i(a_1 a_2 \cdots a_i) := \mathcal{L}(\mathcal{R}(\mathcal{P}(a_1 a_2 \cdots a_i), \kappa)) \tag{2}$$

where $i = 0, 1, \cdots, n-1$ and $\mathcal{L}$ returns the "least significant bit". Note that we are able to specify this scheme in such a succinct way thanks to the formulation and proof of Theorem 1. Here $\mathcal{R}$ is a pseudorandom function or a pseudorandom permutation (i.e., a block cipher) such as Rijndael [6], and $\mathcal{P}$ is a padding function that expands $a_1 a_2 \cdots a_i$ into a longer string that matches the block size of $\mathcal{R}$. κ is the cryptographic key used in the pseudorandom function $\mathcal{R}$. Its length should follow the guideline (e.g., between 128 and 256 bits in 32-bit steps in Rijndael) specified for the pseudorandom function that is actually adopted.

As we can see from (2), the cryptography-based anonymization function is uniquely determined

[3]TCPdpriv may be modified to allow the binding table used in one anonymization session to be saved and used in another session for consistent anonymization. However, the binding table is large and can be cumbersome for distribution. Also the anonymization process still has to be serialized.

by κ. In other words, an unanonymized address appearing in two different traces will be mapped to the same anonymized address if the same key is used to anonymize both traces. So, for consistent distributed anonymization of multiple traces, the κ needs to be distributed to various hosts or sites where the anonymization will occur. A secure key distribution scheme (such as [7, 8, 9]) suitable for the specific requirements (e.g., scalability) of an organization can be used for this purpose.

The new scheme is designed to be generic: any secure stream and block ciphers, which can be modeled as pseudorandom functions (PRF) or pseudorandom permutations (PRP), may be used in place of $\mathcal{R}$. In the following section, we characterize the best possible security level of F and show that it is provably secure (up to that level) based on the assumption that $\mathcal{R}$ is a PRF (PRP is a special case of PRF).

We implemented our scheme by instantiating $\mathcal{R}$ with Rijndael, a secure block cipher that has been adopted by NIST as AES [6]. As a block cipher, Rijndael can be modeled as strong pseudorandom permutation [10, 11, 12], which is the base assumption for provable security of our scheme. We found that the scheme can process 10,000 packets per second on a 800 MHz Intel Pentium III processor, fast enough for practical purposes. This speed can be doubled if the scheme precomputes and stores the anonymization result for the first 16 bits, costing 128 KB[4]. Note that since this cache is deterministicly generated from the key, it will not interfere with parallel and distributed execution of the scheme.

3 Attacking Prefix-Preserving Anonymization

In this section, we discuss two possible ways in which our scheme may be attacked. An intruder is assumed to have compromised (gain full knowledge to) the bindings between certain number of raw and anonymized address pairs through means other than compromising the key (i.e., the known plaintext attack model). We identify the following two types of the attacks: the first affects only our scheme and the second affects TCPdpriv and our scheme to the same extent.

- **Cryptographic Attack.** Aided by the knowledge of the compromised raw-anonymized address pairs, the intruder tries to infer the cryptographic key used in the anonymization algorithm (κ in

[4]Storing such intermediate results in a software cache with appropriate replacement policies (e.g., LRU) may result in even higher improvement on the overall anonymization speed, when there is a decent amount of locality [13] in the trace. However, such improvement can be highly trace-dependent.

(2)) using all possible cryptanalysis techniques. TCPdpriv is not susceptible to this attack.

- **Semantic Attack.** Without compromising the cryptographic keys, the attacker may still be able to infer a part of (typically a prefix) or even whole unanonymized addresses from an anonymized address by exploiting the semantics of prefix-preserving and traditional cryptanalysis techniques such as frequency analysis. This process can again be aided by the knowledge of the compromised addresses. Note that the semantic attack is inherent with the prefix-preserving anonymization scheme: all prefix-preserving schemes (including TCPdpriv and our scheme) are subject to this type of attack to the same degree.

We will prove that the security of our scheme against cryptographic attack depends solely on the strength of the pseudorandom function used in its construction ($\mathcal{R}$ in (2)). It is not dependent on the data that is anonymized. The robustness of our scheme against semantic attack, on the other hand, is dependent on certain "entropy" property that may vary from trace to trace. Therefore, it is assessed by measuring such properties on specific traces.

In the sequel we study both attacks in detail. In Section 4 we show that our scheme is provably secure against cryptographic attack. In Section 5, we investigate the effectiveness of semantic attacks through measurements on real unanonymized packet traces.

4 Security Analysis of Cryptographic Attack

In this section, we prove that our scheme defined by (1) and (2) achieves the highest level of security achievable by prefix-preserving schemes when the adversaries are assumed to be computationally bounded. In stating the theorems and the proofs, we follow the standard notions of security and proof techniques in the provable security literature [14, 15].

We first characterize the highest level of security achievable by any prefix-preserving anonymization scheme. Suppose that a set of N anonymized addresses S have been compromised. Given an *arbitrary* anonymized address b (fixed after it is chosen), suppose k is the longest prefix match between b and the elements in S. Then, due to the prefix-preserving nature of the anonymization algorithm, the first $(k+1)$ bits of the corresponding raw address, referred to as a, are revealed as mentioned before. The highest level of security that can be achieved is then to ensure that the remaining $(n-k-1)$ bits are indistinguishable from random bits to adversaries.

In order to formalize this concept we first introduce the following definitions:

Definition 2 *(adapted from [16]) Suppose p_0 and p_1 are two probability distributions on the set $\{0,1\}^l$, bit strings of length l. Let $A : \{0,1\}^l \rightarrow \{0,1\}$ be a probabilistic (randomized) algorithm. Let $\epsilon > 0$ and two random variables X_0 and X_1 have distributions p_0 and p_1 respectively. We say that A is an ϵ-distinguisher of p_0 and p_1 provided that $|Pr(A(X_0) = 1) - Pr(A(X_1) = 1)| \geq \epsilon$. We say that p_0 and p_1 are ϵ-distinguishable if there exists an ϵ-distinguisher of p_0 and p_1.*

Definition 3 *We call a function F: $U \rightarrow V$ to be (q,t,ϵ)-pseudorandom, when there is no algorithm A that, given any $x \in U$ at A's choice, can be an ϵ-distinguisher between the uniform distribution on V and the distribution of $F(x)$. Here A is allowed to use F as an oracle on q points of its choice different from x and spends no more than t computation time. Note here that the distribution of $F(x)$ is induced by the distribution of F in function space. So, equivalently, we can say that the function F is ϵ-indistinguishable from a* ***random function****, which can be viewed as a random variable uniformly distributed in the set of all functions from U to V.*

With the above definitions in mind a prefix preserving scheme can be said to attain its highest level of security if the algorithm F is indistinguishable from a *random prefix-preserving function*, a function uniformly chosen from the set of all prefix-preserving functions.

We prove in Theorem 2 that the cryptography-based scheme achieves the aforementioned level of security when the adversaries are assumed to be computationally bounded. In contrast, in TCPdpriv, this indistinguishability is achieved in the information-theoretical sense: the adversary does not need to be computationally bounded. This, however, comes at the cost of maintaining a large binding table (essentially a one-way pad).

The notations (S, N, b, a, and k) introduced in the paragraph before the last will be used throughout this section, and $a = a_1 a_2 \cdots a_n$. Given (S, N, b, a, k) as defined above, we define $\tilde{F} : \{0,1\}^{n-k-1} \rightarrow \{0,1\}^{n-k-1}$ in which $\tilde{F}(x)$ is defined as the last $(n-k-1)$ bits of $F(a_1 a_2 \cdots a_{k+1} || x)$. Here F is defined as in (1), $x \in \{0,1\}^{n-k-1}$, and "$||$" represents concatenation.

Theorem 2 *Given the knowledge of compromised addresses S, if the function $\mathcal{R}$ in (2) is a (32*(N+1), t, $\frac{\epsilon}{2^n * n}$)-pseudorandom function, then $\tilde{F}^{-1}$ is a (0,t,ϵ)-pseudorandom function. In other words, given* *any $y \in \{0,1\}^{n-k-1}$, the distribution of $\tilde{F}^{-1}(y)$ is not ϵ-distinguishable from uniform distribution on $\{0,1\}^{n-k-1}$ for all algorithms A that runs for no more than t time.*

Proof: Since $\mathcal{R}$ is a (32*(N+1), t, $\frac{\epsilon}{2^n * n}$)-pseudorandom function, by Lemma 1, $\tilde{F}$ is a (0,t,$\frac{\epsilon}{2^n}$)-pseudorandom function. Then by Lemma 2, this implies that $\tilde{F}^{-1}$ is a (0,t,ϵ)-pseudorandom function. □

For better continuity of text, we state without the proof the lemmas used in proving Theorem 2. Their detailed proofs are in Appendix A.

Lemma 1 *If $\mathcal{R}$ is a (32*(N+1), t, $\frac{\epsilon}{n}$)-pseudorandom function, then $\tilde{F}$ is a (0,t,ϵ)-pseudorandom function even with the knowledge of S.*

Lemma 2 *If a permutation $G : V \to V$ is a (0, t, ϵ)-pseudorandom function, then G^{-1} is a (0, t, $\epsilon|V|$)-pseudorandom function.*

Remark: If the only assumption about G is that it is a pseudorandom function, then this bound of ($\epsilon|V|$) for G^{-1} is indeed tight. An instance where this bound is tight is shown in the remark after the proof of Lemma in Appendix A.

In this section, we formally characterize the notion of provable security (Definition 2 and 3) in our context: indistinguishability between our anonymization function and a random prefix-preserving function to a computationally constrained adversary (with no more than t computation time). We prove rigorously that our scheme is secure against cryptographic attacks based on this notion of provable security. In the next section, we proceed to explore the security of our scheme and TCPdpriv against semantic attacks.

5 Evaluation of the Effects of Semantic Attacks

In this section, we study the security of prefix-preserving anonymization against semantic attacks. Since the risk of semantic attacks is inherent with all prefix-preserving anonymization schemes, the findings of this study apply to all schemes, including ours and TCPdpriv. Our goal here is to provide a framework for evaluating the privacy risks in releasing an anonymized trace so that trace owners may be better equipped to make informed decisions about releasing anonymized traces.

The security implications of prefix-preserving anonymization of traffic traces using TCPdpriv [4] are briefly studied in [5] and [17]. Here we offer a more formal approach to characterize the security of prefix-preserving anonymized traces against semantic attacks. Our contribution is summarized as following:

1. We provide a framework (including a set of metrics) for evaluating the effect of attacks on anonymized traces. The framework assumes that an attack is characterized by the number of address mappings that are compromised and by properties that compromised addresses may have (e.g., random, all DNS addresses or frequently-occurring addresses).

2. We show that, for the traces we examine, an attacker that compromises a random set of addresses causes approximately the same damage as a more careful attacker which compromises the same number of addresses optimally. This seems to indicate that compromising addresses at random is a good means of attacking prefix-preserving anonymization.

3. We show that the damage caused by an attack can be trace-specific. This means that no blanket statements can be made regarding the safety of releasing traces but rather each case needs to be evaluated on its own merits.

4. We show that two "obvious" attacks: using frequency analysis and compromising all DNS server addresses, yield as much damage as a small number of randomly compromised addresses. We also discuss the feasibility and likely effectiveness of other more elaborate attacks.

5.1 Metrics to Measure Effect of Attacks

When we study the security of an anonymized trace, we would like to measure the amount of information that is leaked from or kept untouched in the whole trace as a consequence of compromising some address mappings. Note that the specifics of the attack by which address mappings have been compromised is not important, and what ultimately matters is the result of the attack, i.e., the number of compromised addresses and their properties.

In this section we define three metrics to measure the effect of attacks on anonymized traces. Each measure reflects a different security concern.

The number of unknown compressed bits, C: When some address mappings are compromised, the states of some nodes of the *anonymization function* (see figure 1) are revealed and the anonymization function is partially compromised. This leads to the definition of C as the total number of nodes in the anonymization function whose states are *not* known. Note that C corresponds to the entropy of the anonymization function after the attack.

The number of unknown uncompressed bits, U: Another concern is the security state of the anonymized addresses. When some address mappings are compromised, all the bits in the compromised addresses are revealed. In addition and due to the prefix-preserving nature of the anonymization algorithm, certain bits in other addresses are also revealed. This leads to the definition of U as the total number of bits that are *not* known, in all addresses.

The number of addresses with exactly i known most significant bits, F_i: Neither C nor U describe exactly where bits have been revealed. We, therefore, measure F_i defined as the total number of addresses that has *exactly* i most significant bits known, where $0 \leq i \leq 32$.

5.2 An Evaluation of the Effect of Attacks on Real Traces

Recall that we model the effect of an attack by the number of compromised addresses and the properties associated with them. In this section we consider the effect of compromising N addresses chosen either randomly or according to a greedy algorithm (which we prove is optimal for some measures).

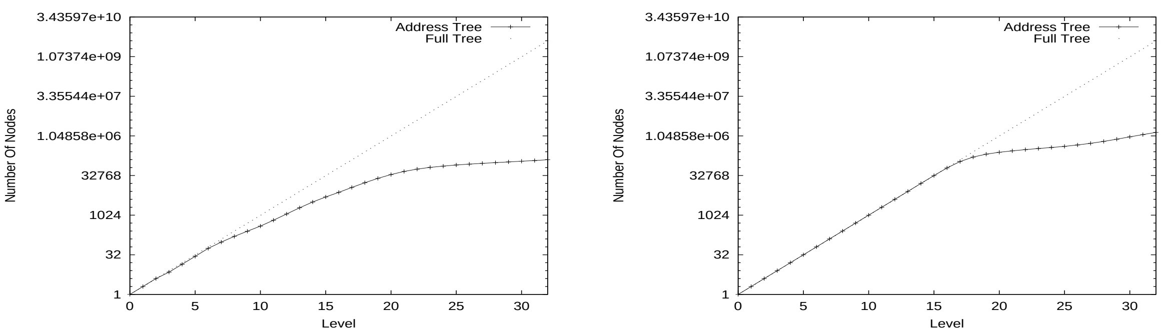
We present results based on a trace from Tier-1 ISP link and a publicly available one from NLANR [18]. Note that both traces contain real (unanonymized) IP addresses[5]. The properties of the two traces are shown in Table 1.

Figure 2(a) shows the number of nodes in each level of the original address tree built from the NLANR trace. The figure shows that the number of nodes increases when the level increases. It also shows that the tree is quite dense on the top but becomes sparser as it progresses towards the leaves representing the IP addresses. Similar figures are obtained from the Tier-1 ISP trace and are shown in figure 2(b).

[5] The NLANR trace is an destination-IP-address-only trace.

	Tier-1 ISP	NLANR [18]
Type	Full Header	Destination IP Only
Location	Packet-Over-SONET OC3 link	N/A
Start Time	09:56 PDT 8/9/2000	N/A
End Time	19:56 PDT 8/9/2000	N/A
Size	50GB Binary	930MB ASCII
Number of Packets	567,680,718	31,518,464
Number of Distinct Addresses	1,423,937	130,163

Table 1: Example Traces



(a) NLANR IP Address Trace

(b) Tier-1 ISP IP Header Trace

Figure 2: Shape of Address Trees

Effect of Compromising Random Addresses We first consider the effect of compromising a random number of addresses. Figures 3(a),(b) and (c) are simulation results on the NLANR trace and show how U and C decrease as the number of compromised IP addresses increases. The results are obtained by randomly choosing a certain number of addresses from the NLANR trace and evaluating the C and U measures assuming they are compromised. This is repeated 10 times and the graphs represent the average of the experiments. Figures 3(b) and (c) magnify the portion of 3(a) when the number of compromised IP addresses ranges from 0 to 3000 and 0 to 300, respectively.

We can see in the graphs that the value of C drops almost linearly with respect to the number of compromised IP addresses, which means the anonymization function is quite resistant to the attacks. The value of U drops very fast initially and flattens out. This implies that an ordinary address has a

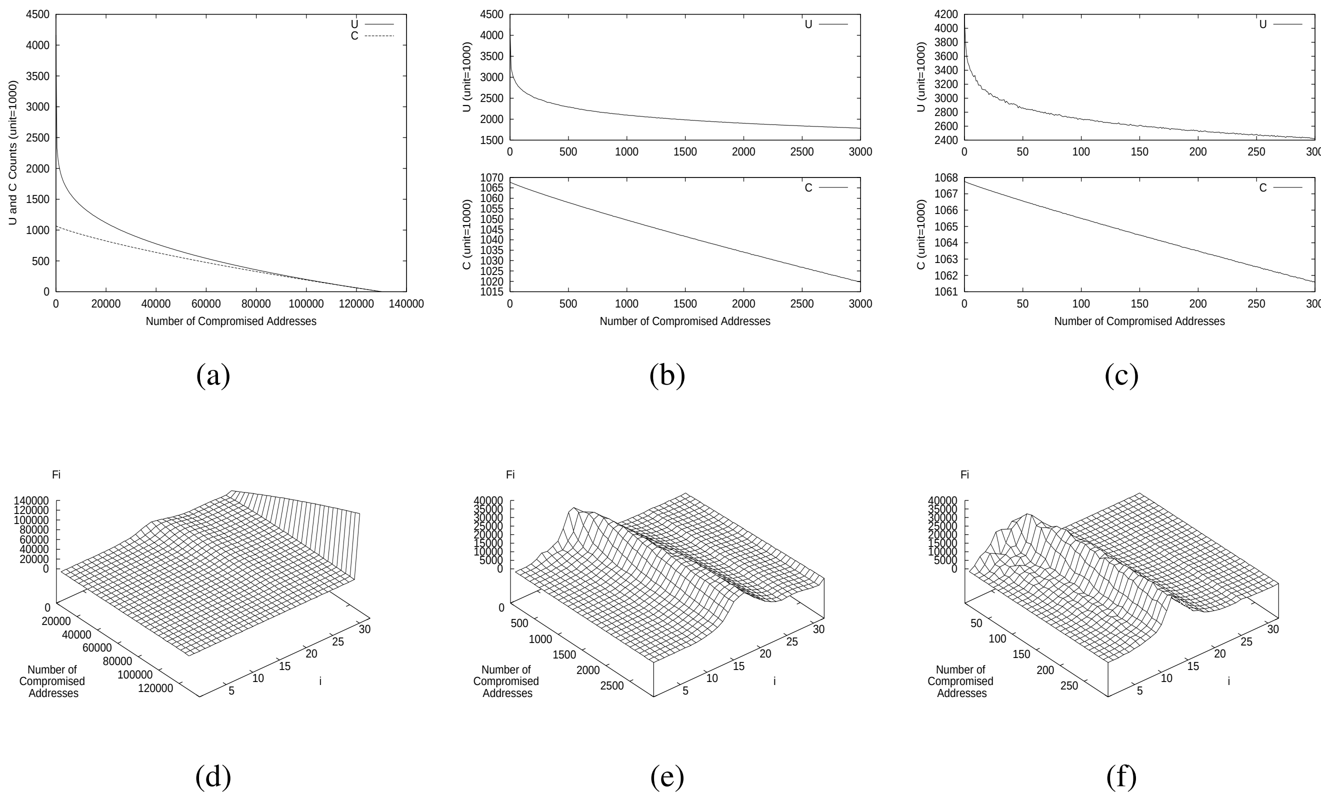


Figure 3: Measurement of U,C and F after attacks on the NLANR trace resulting in randomly chosen compromised addresses. Figures (a),(b) and (c) are measurements of U and C; (b) and (c) magnify (a) at the portion of x range 0 to 3000 and 0 to 300, respectively. Figures (d), (e) and (f) are the measurements of F; (e) and (f) magnify (d) at the portion of x range 1 to 3000 and 1 to 300, respectively.

very high probability to have several of its 32 bits revealed (prefix bits) but a low probability to have a large number of them revealed.

Figures 3(d), (e) and (f) are simulation results on the NLANR trace and show F_i, the number of addresses who have had exactly i most significant bits revealed, for $i = 0, ..., 32$ and various values of N. Figures 3(e) and (f) magnify the portion of 3(d) when the number of compromised IP addresses ranges from 1 to 3000 and 1 to 300, respectively. The ridge in figure 3(d) shows that the effect of the attack is relatively low when the total number of compromised address mappings is a small proportion of the total number of addresses, e.g., no more than 20,000 out of 130,163. The ridge in figure 3(d) shows that most addresses have around 16 bits compromised when there are approximately 2000 addresses compromised. This could mean that privacy is preserved in situations where the least significant 16

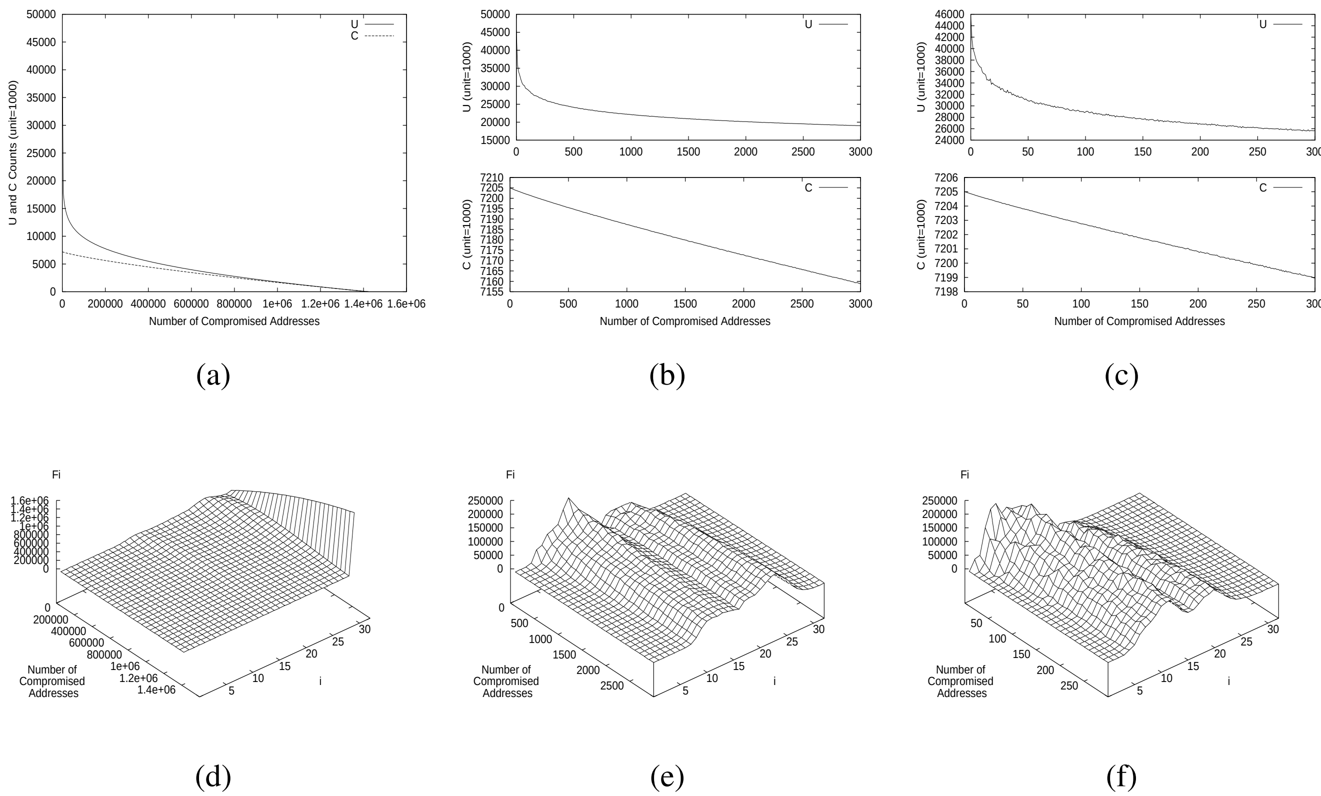


Figure 4: Measurement of U,C and F after attacks on the Tier-1 ISP trace resulting in randomly chosen compromised addresses. Figures (a),(b) and (c) are measurements of U and C; (b) and (c) magnify (a) at the portion of x range 0 to 3000 and 0 to 300, respectively. Figures (d), (e) and (f) are the measurements of F; (e) and (f) magnify (d) at the portion of x range 1 to 3000 and 1 to 300, respectively.

bits are more important for personal privacy than the most significant 16 bits. Similarly, the ridge in figure 3(f) is centered around the 12-bit line.

The Tier-1 ISP trace contains many more distinct addresses than the NLANR trace does, the simulation on the Tier-1 ISP trace, however, exhibits similar trends as shown in figure 4. In figure 4, the U curve drops faster and the F ridge spreads wider up along y-axis than they do in figure 3. This suggests that the Tier-1 ISP trace is not as resistant to semantic attacks as the NLANR trace is.

Effect of Compromising Greedily-Generated Addresses A surprising result is that a *greedy algorithm*, which chooses at each step an address that causes the greatest *single-step* reduction in U or C value, actually generates the optimal sequence of compromised addresses. That is, for any $N > 0$, a sequence of N addresses generated by the greedy algorithm cause the maximum reduction in U (or C)

among all sets of N compromised addresses. Since the formal formulation of the greedy algorithm and its optimality proof is very involved, for better continuity of text, we move it to Appendix B.

Figure 5 shows simulation results demonstrating the effect of an attack on the Tier-1 ISP trace as a function of the number of addresses compromised and assuming the attacker can choose these addresses to minimize U.

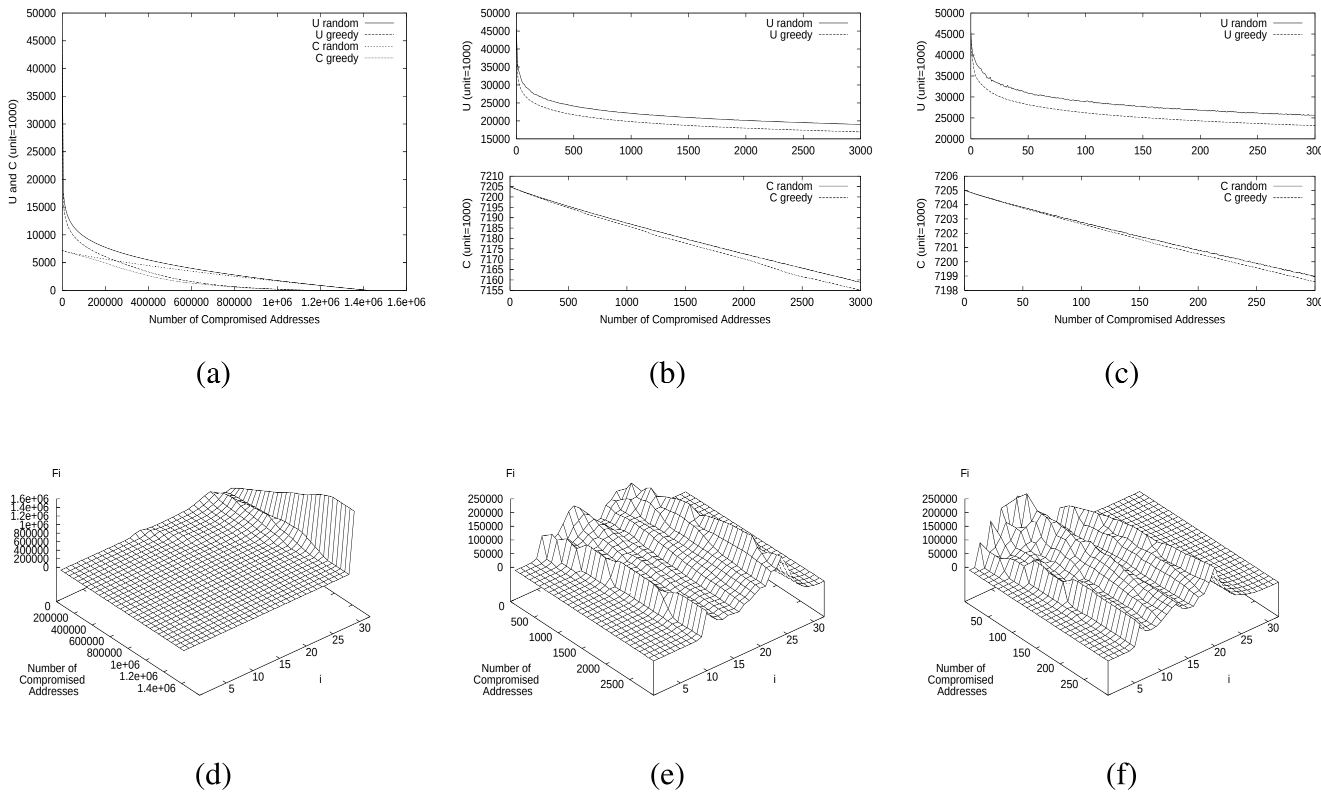
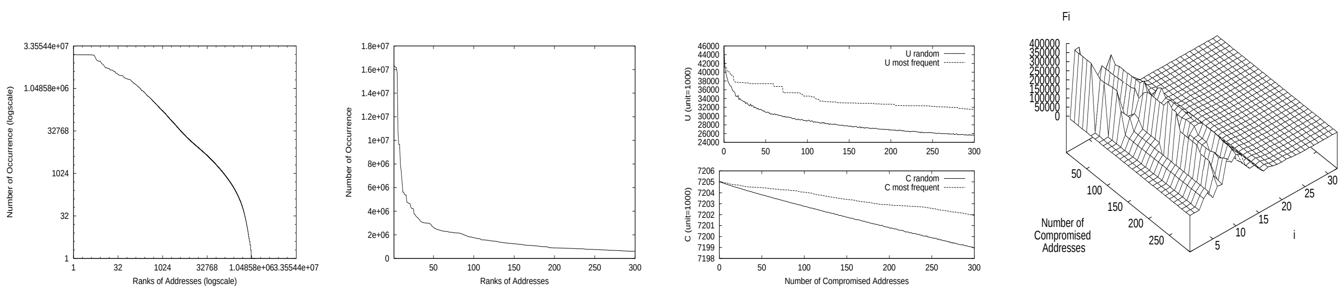


Figure 5: Measurement of U,C and F after attacks on the Tier-1 ISP trace resulting in U-optimal greedy set of compromised addresses. Figures (a),(b) and (c) are measurements of U and C; (b) and (c) magnify (a) at the portion of x range 0 to 3000 and 0 to 300, respectively. Figures (d), (e) and (f) are the measurements of F; (e) and (f) magnify (d) at the portion of x range 1 to 3000 and 1 to 300, respectively.

Comparing the figures in figure 5 with those in figure 4 we see that for the U and C measures, the effect of compromising some number of addresses randomly is similar to the effect of compromising an optimally chosen set of addresses. For this trace, this seems to indicate that compromising addresses at random is a good means of attacking prefix-preserving anonymization.

5.3 Results on Two Specific Attacks

As mentioned earlier, we have chosen to characterize attacks by the number and property of the addresses they reveal. We have consider randomly-chosen and optimally-chosen addresses. We now consider what happens when the set of addresses have properties that derive from a specific attack. We consider two types of attacks that have been mentioned in the literature [5]: frequency analysis and DNS server tracing.



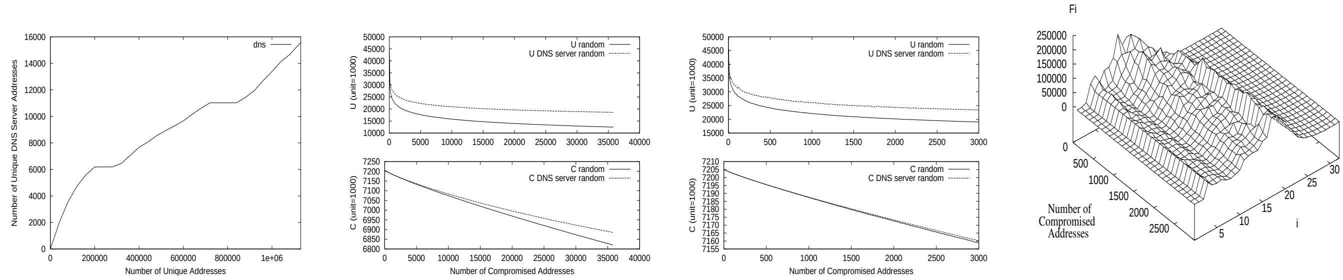
(a) (b) (c) (d)

Figure 6: Effect of Frequency Analysis Attack On Tier-1 ISP Trace

Frequency Analysis: IP addresses of popular sites can be inferred from their high frequency of occurrence in an anonymized trace. Figure 6(a) shows the frequency that different addresses occur in the Tier-1 ISP trace. Addresses are sorted by their frequency of occurrence from left to right. Figure 6(b) magnifies the portion of 6(a) for the 300 most frequent addresses. These figures show that only a small number of addresses (in the tens) are actually distinguishable from others by their frequency of occurrence.

In figures 6(c) and (d), we show the U, C and F values assuming the N most-frequently-occurring addresses are compromised as N varies. Figure 6(c) shows that compromising the most frequent 300 addresses has the same effect on U as compromising about 40 randomly chosen addresses. Figure 6(d) shows that compromising frequently occurring addresses has a more localized effect, that is, affecting mainly the most significant bits (compare with figure 4). According to these results, frequency analysis, by itself, does not appear to be a serious threat to this Tier-1 ISP trace.

DNS Server Address Tracing: The IP addresses of DNS servers may be inferred from the hierarchical relationship among them. Starting with a root DNS server, an attacker can trace down the DNS



(a) (b) (c) (d)

Figure 7: Effect of DNS Server Tracing Attack on Tier-1 ISP Header Trace

server hierarchy based on their protocol-defined relationship in the anonymized trace, assuming the attacker has enough knowledge about the DNS server hierarchy.[6]

Figure 7(a) shows the number of DNS server addresses that appear in a portion of the Tier-1 ISP trace as a function of the number of distinct addresses, as we consider more and more records in the trace. This figure shows that a proportion in the range of $1/15$ to $1/40$ of distinct addresses in the Tier-1 ISP trace are DNS server addresses, depending on where the trace is cut. Referring back to figure 4, it can be seen that compromising this many random addresses represents a significant risk to the anonymization process. This might lead one to conclude that an attack that reveals the mapping of all DNS server addresses in the trace will essentially "break" the anonymization process. But this is somewhat misleading since the DNS server addresses are not really random. We investigate this matter further in figure 7(b) which also derives from the Tier-1 ISP trace. In the figure we show the value of U and C as a function of the number of compromised addresses when these compromised addresses are drawn at random from the set of all addresses and when they are drawn at random from the set of DNS server addresses. The figure shows that for the same number of compromised addresses the attacker can reveal more "bits" if the addresses were chosen at random from the entire set of addresses as opposed to the set of DNS server addresses. In fact, compromising all 35,903 DNS server addresses is equivalent to compromising a set of only approximately 3,500 random addresses. More close-up

[6]This assumption is quite questionable though. Not all DNS servers allow listing of their downstream servers for security reasons. This makes it difficult to get the topology of the DNS hierarchy and we have not seen any such topology publicly available.

results are shown in figure 7(c) and (d), which are the U, C, F curves for 0 to 3000 randomly-chosen compromised DNS server address mappings. These results suggest that, for this trace, an attack that reveals DNS server addresses is perhaps not as serious as one would expect and that in fact an attack that can reveal much fewer random addresses would be more effective.

5.4 Miscellaneous attacks

In addition to the frequency analysis and DNS tracing attacks we discussed in the previous section, we also study other types of attacks that may pose threats to our prefix-preserving anonymization scheme. Note, however, that our results here are preliminary and still a topic of further research.

- **Active attacks:** This type of attack also affect non-prefix-preserving address anonymization schemes. In this attack, an intruder simply injects some "probing packets" into the network, and hopefully gets them recorded and anonymized in the trace. Assume that the intruder keeps a copy of the injected packets, he/she will be able to recover bindings between unanonymized and anonymized addresses later when the trace is released. This type of attack is very hard to counter, since it can be made highly robust: the destination (victim) IP address can be encoded into fields such as port numbers and packet length. Detection of this attack is also tricky since to a certain extent it can be viewed as a covert channel problem [19]. Keeping information such as when and where the trace will be gathered secret seems to be the best defense against such attacks. This, however, still can not thwart an intruder that performs probing continuously over a long period of time.

- **Port scanning:** Port scanning is the standard technique for an intruder to identify an Internet host for potential break-in. It does so by "scanning" a subnet for a specific service (port number) that is vulnerable to intrusion. The IP addresses to be scanned often advance in a step of 1 (i.e., $A, A+1, A+2, \ldots$). Though such an attack does not target our anonymization process, it may still pose a serious threat. That is, if the intruder recognizes port scan in the trace, and if $A's$ anonymized version is compromised from the trace, $A+1, A+2, \ldots$ will also be revealed. Fortunately, intrusion detection software (e.g., Snort [20]) for detecting port scanning is available. The unanonymized trace can be first filtered by such software before being anonymized. We are currently measuring the amount (percentage) of port scanning traffic that is contained in our traces to understand the effect of filtering such traffic from a trace.

- **Routing table inference:** In some routing performance research, trace data and a relevant routing table may need to be released together. This can be done by anonymizing the IP prefixes in the routing table using the same key as trace anonymization. Note that prefix-preserving anonymization can be applied to IP prefixes of any length. In this case, it is very important that the plaintext routing table (also routing tables of "nearby" routers, which can be similar) to be kept secret[7]. We also note that we only need to anonymize and release the routing table entries that the traffic trace has actually accessed, which will make it even harder for the intruder to infer useful information from the anonymized routing table. We experimented our 50 GB Tier-1 ISP trace, and found that our trace matches only 2,988 out of 45,008 prefixes in the routing table that came with it.

6 Summary of Our Work

Our work mainly consists of two parts. In the first part, we characterize the prefix-preserving IP address anonymization using a canonical form, and propose a new cryptography-based scheme. Unlike TCPdpriv, our scheme is suitable for (consistent) parallel and distributed anonymization of traffic traces. We prove rigorously that our scheme is secure up to the level a prefix-preserving scheme could possibly deliver. We implemented the scheme and evaluated its performance on real traffic traces (10,000 packets per second using Rijndael). In the second part of our work, we first propose a framework (including a set of metrics) for evaluating the effect of attacks on anonymized traces. Using this framework, we study the effect of two well-known attacks, frequency analysis and DNS tracing, on two real-world traffic traces. We also formally characterize the optimal fashion (greedy algorithm) in which an attacker should compromise a subset of anonymized addresses. We show that compromising an optimal set of N addresses is almost as effective as randomly compromising N addresses. Finally, we found that the damage caused by an attack can be very much trace specific and no blanket statements can be made regarding the safety of releasing traces but rather each case needs to be evaluated on its own merits.

[7]Otherwise, it is straightforward for an intruder to infer the true identifies of a large portion of IP prefixes by studying their length or common-prefix relationship. In addition, routing tables of "nearby" routers should also be kept secret, since they can be similar.

References

[1] Tony McGregor, Hans-Werner Braun, and Jeff Brown. The NLANR network analysis infrastructure. *IEEE Communications Magazine*, 38(5):122–128, May 2000.

[2] The Internet traffic archive. http://ita.ee.lbl.gov/, April 2000.

[3] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *Proc. ACM Sigcomm 2000*, pages 97–110, September 2000.

[4] Greg Minshall. *TCPdpriv Command Manual*, 1996.

[5] T. Ylonen. Thoughts on how to mount an attack on tpcpdriv's "-a50" option ... In *TCPpdpriv source distribution*, 1996.

[6] J. Daemen and V. Rijmen. AES proposal: Rijndael. Technical report, Computer Security Resource Center, National Institute of Standards and Technology, http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, Feb 2001.

[7] Neuman and Ts'o. Kerberos: An authentication service for computer networks, from IEEE communications magazine, september, 1994. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*.

[8] R. Ganesan. Yaksha: Augmenting kerberos with public-key cryptography, 1995.

[9] Michael K. Reiter, Matthew K. Franklin, John B. Lacy, and Rebecca N. Wright. The omega key management service. In *ACM Conference on Computer and Communications Security*, pages 38–47, 1996.

[10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[11] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[12] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of Cipher Block Chaining. In Yvo G. Desmedt, editor, *Advances in Cryptology - Crypto 94*, number 839, pages 341–358. Springer Verlag, 1994.

[13] M. Á. Ruiz-Sánchez, E. W. Biersack, and W. Dabbous. Survey and taxonomy of ip address lookup algorithms. *IEEE Network*, 15(2):8–23. Mar/April, 2001.

[14] M. Bellare. Practice-oriented provable-security. In *First International Workshop on Information Security(ISW97)*, Boston, Massachusetts, 1998. Springer-Verlag. Lecture Notes in Computer Science No. 1396.

[15] S. Goldwasser and M. Bellare. Lecture notes on cryptography. available online from http://www-cse.ucsd.edu/users/mihir/papers/gb.html.

[16] D. Stinson. *Cryptography, Theory and Practice*. CRC Press, 1995.

[17] K. Cho, K. Mitsuya, and A. Kato. Traffic data repository at the wide project. In *Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*, San Diego, CA, June 2000.

[18] NLANR. File 'sdc-964451101.tstamp+plen+destip' included with NLANR network traffic packet header traces, 2000.

[19] B. Lampson. A note on the confinement problem. *CACM*, 16(10).

[20] Snort, the open source network intrusion detection system, 2001.

[21] A. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proc. of IEEE FOCS'82*, pages 80–91, November 1982.

Appendix A. Proofs of Lemma 1 and Lemma 2

In this appendix, we offer detailed proofs of Lemma 1 and Lemma 2 (introduced in Section 4). For simplicity of discussion, we use U^i to denote uniform distribution on $\{0,1\}^i$. As a convention, random variables and algorithms will be denoted by capital letters and fixed values by lower-case letters. We use "$=_{dist}$" to denote that two random variables are equal in distribution. Again, recall that n denotes the number of bits in an IP address.

Proof of Lemma 1: We prove the contrapositive. Suppose $\tilde{F}$ is not a (0,t,ϵ)-pseudorandom function. Then there is an algorithm A, which picks an $s \in \{0,1\}^{n-k-1}$ at its choice, can be an ϵ-distinguisher between $\tilde{F}(s)$ and the uniform distribution on $\{0,1\}^{n-k-1}$. Also, A uses no more than t computation time. We need to show if such A exists, then $\mathcal{R}$ is not a (32*(N+1), t, ϵ)-pseudorandom function.

We define $\tilde{F}_i : \{0,1\}^{n-k-1} \to \{0,1\}$ in which $\tilde{F}_i(x)$ is the i_{th} bit of $\tilde{F}(x)$ for any x, $1 \le i \le n-k-1$. Let $U_i, i = 1,2,\cdots,n-k-1$ be random variables with uniform distributions on $\{0,1\}^i, i = 1,2,\cdots,n-k-1$ respectively. We define random variables $Y_i, 0 \le i \le n-k-1$. For each Y_i, given an outcome ω in the probability space, $Y_i(\omega) := \tilde{F}_1(s)(\omega) \mid\mid \tilde{F}_2(s)(\omega) \mid\mid \cdots \tilde{F}_i(s)(\omega) \mid\mid U_{n-k-1-l}(\omega)$, $i = 1,2,\cdots,n-k-1$. Then Lemma 3 shows that there exists $l, 1 \le l \le n-k-1$ such that there is a distinguisher algorithm B that satisfies $|Pr(B(Y_{l-1}) = 1) - Pr(B(Y_l) = 1)| \ge \frac{\epsilon}{n-k-1}$. However, as we will show next, this will imply that $\mathcal{R}$ can not be a (32*(N+1), t, ϵ)-pseudorandom function.

Recall that $\tilde{F}(x)$ is defined as the last $n-k-1$ bits of $F(a_1a_2\cdots a_{k+1}||x)$, where $a := a_1a_2\cdots a_n$. We construct an algorithm C that picks $D := PAD(a_1a_2\cdots a_{k+1}s_1s_2\cdots s_l)$ and tries to distinguish the distribution of $\mathcal{R}(D, key)$ from U^{128} (uniform distribution on the range of $\mathcal{R}$). Given an input

$X \in \{0,1\}^{128}$, C first uses $\mathcal{R}(*, key)$ as an oracle $32*N$ times to obtain S (the N pairs of compromised IP addresses) using (2). Then C uses $\mathcal{R}$ as an oracle $l-1$ more times to obtain $\tilde{F}_i(s), i = 1, 2, \cdots, l-1$. Then C constructs a random variable Y on $\{0,1\}^{128}$ as follows. Given an outcome ω in the probability space, $Y(\omega)'s$ first $l-1$ bits are $\tilde{F}_1(s)\ \tilde{F}_2(s) \cdots \tilde{F}_{l-1}(s)$, its l_{th} bit is $LSB(X)$, and its last $n-k-1-l$ bits are $U_{n-k-1-l}(\omega)$. Finally, C returns $B(Y)$ as the result. It is not hard to verify that (a) if X has the distribution $\mathcal{R}(D, key)$ then Y has the distribution of Y_l and (b) if X has the distribution U^{128} then Y has the distribution of Y_{l-1}. Therefore $|Pr(C(\mathcal{R}(D, key)) = 1) - Pr(C(U^{128}) = 1)|$ = $|Pr(B(Y_l) = 1) - Pr(B(Y_{l-1}) = 1)| \geq \frac{\epsilon}{n-k-1} > \frac{\epsilon}{n}$. This shows that C is an $\frac{\epsilon}{n}$-distinguisher between the distribution of $\mathcal{R}(D, key)$ and U^{128}. Also, C has made no more than $32*(N+1)$ oracle calls and uses no more than t time (evaluating $B(Y)$). This contradicts the assumption that $\mathcal{R}$ is a (32*(N+1), t, $\frac{\epsilon}{n}$)-pseudorandom function. □

In the following we introduce a variation of a standard lemma used in developing the concept of pseudorandom number generator [16]. The original idea behind this proof is attributed to Yao [21]. Let p_0 and p_1 be two probability distributions on $\{0,1\}^m$. Let X_0 and X_1 be two random variables of distribution p_0 and p_1 respectively. We define $m+1$ random variables $Y_i, i = 0, 1, \cdots, m$ on the set $\{0,1\}^m$ as follows. Given an outcome ω in probability space, $Y_i(\omega)$:= (the first i bits of $X_0(\omega)$) || (the last $m-i$ bits of $X_1(\omega)$).

Lemma 3 (Hybrid Argument) *If p_0 and p_1 are ϵ-distinguishable, then there exists $l, 1 \leq l \leq m$ such that the distribution of Y_{j-1} and the distribution of Y_j are $\frac{\epsilon}{m}$-distinguishable.*

Proof: It is not hard to verify that $Y_0 =_{dist} X_1$ and $Y_m =_{dist} X_0$. Suppose A is a ϵ-distinguisher between X_0 and X_1. Then $|Pr(A(X_0) = 1) - Pr(A(X_1) = 1)| \geq \epsilon$. However, $Pr(A(X_0) = 1) - Pr(A(X_1) = 1)$ = $\Sigma_{i=1}^m (Pr(A(Y_{i-1}) = 1) - Pr(A(Y_i) = 1))$. So according to the triangle inequality, $|Pr(A(X_0) = 1) - Pr(A(X_1) = 1)| \leq \Sigma_{i=1}^m |(Pr(A(Y_{i-1}) = 1) - Pr(A(Y_i) = 1))|$ (*). Therefore, there must exist $j, 1 \leq j \leq m$ such that $|Pr(A(Y_{j-1}) = 1) - Pr(A(Y_j) = 1)| \geq \frac{\epsilon}{m}$, since otherwise (*) will not hold. □

Proof of Lemma 2: We prove the following contrapositive. Suppose G^{-1} is not a (0, t, $\epsilon|V|$)-pseudorandom function. Let U^V denote the uniform distribution on V. Then there is an algorithm A, which picks a y_0 at its choice, such that $Pr(A(G^{-1}(y_0)) = 1) - Pr(A(U^V) = 1) \geq \epsilon|V|$. Here A

executes no more than t time. We construct an algorithm $B(x, y)$ such that

$$B(x,y) = \begin{cases} A(x) & : \quad y = y_0 \\ 0 & : \quad otherwise \end{cases} \tag{3}$$

Then we construct C such that it can pick an x at its choice and let $Pr(C(G(x)) = 1) - Pr(C(U^V) = 1) \geq \epsilon$. C works as follows. With every execution, C first picks X_0 uniformly randomly from V. Then given any input y, $C(y) := B(X_0, y)$. It remains to show that C is an ϵ-distinguisher. First, we can see that $Pr(C(y) = 1)$=$Pr(B(X_0, y_0) = 1) * Pr(y = y_0)$. So $Pr(C(G(X_0)) = 1)$ = $Pr(B(X_0, G(X_0)) = 1)$ = $\Sigma_{\alpha \in V} Pr(B(\alpha, G(\alpha)) = 1) * Pr(X_0 = \alpha)$ = $\Sigma_{\beta \in V} Pr(B(G^{-1}(\beta), \beta) = 1) * \frac{1}{|V|}$ = $\frac{1}{|V|} Pr(B(G^{-1}(y_0), y_0) = 1)$ = $Pr(A(G^{-1}(y_0)) = 1) * \frac{1}{|V|}$. Also $Pr(C(U^V) = 1)$ = $Pr(B(U^V, G(X_0)) = 1)$ = $\frac{1}{|V|} Pr(B(U^V, y_0) = 1)$ = $\frac{1}{|V|} Pr(A(U^V) = 1)$. Therefore $|Pr(C(G(X_0)) = 1) - Pr(C(U^V))|$ = $\frac{1}{|V|} |Pr(A(G^{-1}(y_0)) = 1) - Pr(A(U^V) = 1)| \geq \frac{1}{|V|} * (|V|\epsilon)$ = ϵ. $\Box$

Remark: If the only assumption about G is that it is a pseudorandom function, then this bound of $|V|\epsilon$ for G^{-1} is "almost" tight. To see this, let G be the following (randomized) function. We choose a fixed element $y_0 \in V$ and any subset S of V such that $|S|$ = $\lfloor \frac{|V|}{2} \rfloor$. We also pick a random value x_0 uniformly distributed on S. Then we let $G(x_0) := y_0$, and G restricted on the domain $V - X(\omega)$ be a one-to-one random function from $V - \{x\}$ to $V - \{y\}$. Then it can be shown that G is a (0, ∞, $\frac{1}{|S|}$)-pseudorandom function as $G^{-1}(y_0)$ can be any element in S. However, we will show that G^{-1} is a (0, ∞, $\frac{1}{2}$)-pseudorandom function as follows. An adversary first picks y_0. Then the uniform distribution on S and the uniform distribution on V can be distinguished by the following algorithm A. Given an input x, A outputs 1 if $x \in S$ and 0 otherwise. Let U^V be the uniform distribution on V. Then $Pr(A(G^{-1}(y_0)) = 1)$ = 1 and $Pr(A(U^V))$ = $\frac{|S|}{|V|} \leq \frac{1}{2}$. So $|Pr(A(G^{-1}(y_0)) = 1) - Pr(A(U^V)) \geq \frac{1}{2}$. So when G is a (0, ∞, $\frac{1}{S}$)-pseudorandom function, G^{-1} is not even a (0, ∞, $\frac{1}{2} - \delta$) for any positive δ.

Appendix B. Greedy Algorithm and Its Optimality Proof

In this section, we formally define the greedy algorithm and state the optimality of the greedy algorithm in reducing the U value. Since the greedy algorithm for C value can be similarly formulated and proved, in the interest of space, we omit that part. In the following, we first introduce the notations and definitions we are going to use in the proof of our main theorem that the greedy algorithm is optimal.

1. Greedy(tree x, int N)
2. $S := \emptyset$
3. $R := 0$
4.   for i := 1 to N do
5.   choose $y \in LF(x) - S$ that maximizes $P(S, y, x)$
6. $R := R + P(S, y, x)$
7. $S := S \bigcup \{y\}$

Figure 8: Our Greedy Algorithm.

Notation 1 *Whenever there is no ambiguity, we denote an address tree (defined in Section 2) or a subtree by its root node. We denote the set of leaf nodes on the tree x as $LF(x)$. Given a tree node x, we define $x.left$ and $x.right$ as its left and right child. Also, we denote the cardinality of a set S by $|S|$. Throughout the rest of this paper, n always denotes the number of bits in an IP address.*

Definition 4 *Given a tree x, and a set of leaves S. We refer to $S \bigcap LF(x.left)$ as the left set of S, denoted as $LS(S)$, and $S \bigcap LF(x.right)$ as the right set of S, denoted as $RS(S)$.*

Definition 5 *Given a tree x of height h, we define $R(S, x)$ as the number of address bits, among the last h bits of all leaf nodes (IP addresses), which are revealed when the set S ($S \subseteq LF(x)$) of IP addresses are compromised. Note here that, in computing $R(S, x)$, the first $n - h$ bits are ignored (n is the number of bits in an IP address). $R(S, NULL)$ is 0 by definition. We define $P(S, y, x) = R(S \bigcup \{y\}, x) - R(S, x)$, which is the the number of bits that will be newly compromised when y is added to the compromised address set S.*

Definition 6 *Given a tree x, a set S of leaves is called the* optimal set *defined on x, if given any other leaf set S' of the same cardinality, $R(S', x) \leq R(S, x)$.*

Given a tree x, the greedy algorithm to choose m IP addresses to compromise is shown in Fig. 8. Note that it is a randomized algorithm, since when there are more than one $y's$ that maximizes $P(S, y, x)$ (line 5), they will be randomly picked with equal probability. In the program, the variable S is the set of IP addresses compromised and R has the value of $R(S, x)$.

Definition 7 *Given a tree x, a leaf set S is called a* greedy set *defined on x, when the greedy algorithm (shown in Fig. 8) with input x will generate a sequence of $|S|$ leaves that are exactly elements of S with nonzero probability.*

In the following, we state and prove the following theorem, which implies that the greedy algorithm indeed generates an optimal set. Lemmas used in the proof will be stated and proved after the theorem.

Theorem 3 *Any greedy set is also an optimal set.*

Proof: Given a tree x, we only need to prove the following: given any $N > 0$, and any optimal set O and greedy set S of the same cardinality N, $R(S, x) \geq R(O, x)$. This is trivially true when $N = 1$. So in the following, we only consider $N > 1$. We induct on the height h of the tree x. The conclusion trivially holds for $h = 1$ (a single node tree). Suppose the conclusion also holds for $h = k$.

We now prove the theorem for $h = k + 1$ and $N > 1$. Since the elements of set S is a possible sequence generated by the greedy algorithm, they can be ordered according to the greedy order they appear in the sequence. We denote $S_l = l_1, l_2, \cdots, l_i$ as elements of $LS(S)$ in the greedy order, and $S_r = r_1, r_2, \cdots, \cdots, r_{N-i}$ as elements of $RS(S)$ in the greedy order. Similarly, we denote $LS(O)$ as $O_l = l'_1, l'_2, \cdots, l'_j$ and $RS(O)$ as $O_r = r'_1, r'_2, \cdots, r'_{N-j}$ (no order is assumed in this case). According to Lemma 4, the sequences S_l and S_r are greedy sequences in the trees $x.left$ and $x.right$, respectively.

When $i = j$, according to induction hypothesis, $R(O_l, x.left) \leq R(S_l, x.left)$ and $R(O_r, x.right) \leq R(S_r, x.right)$, since the height $x.left$ and $x.right$ are both k. Since $N > 0$, $R(O, x) = R(O_l, x.left) + R(O_r, x.right) + |LF(x)| \leq R(S_l, x.left) + R(S_r, x.right) + |LF(x)| = R(S, x)$ according to Lemma 5.

Now we only need to consider the case where $i \neq j$. WLOG, we assume that $i > j$. Then S_r can be extended to a longer greedy sequence $Z = r_1, r_2, \cdots, r_{N-j}$, where $r_{N-i+1}, r_{N-i+2}, \cdots, r_{N-j}$ are drawn from $x.right$. Also, $Y = l_1, l_2, \cdots, \cdots, l_j$, being a subsequence of S_l (a greedy sequence in the tree $x.left$ by Lemma 4), is also a greedy sequence in the tree $x.left$. Then according to the induction hypothesis, $R(Z, x.right) \geq R(O_r, x.right)$ since $|Z| = |O_r| = N - j$ and $R(Y, x.left) \geq R(O_l, x.left)$ since $|Y| = |O_l| = j$. Define $X = Y \bigcup Z$. Then $R(X, x) \geq R(O, x)$ according to Lemma 5. Now to prove $R(S, x) \geq R(O, x)$, our final step is to prove $R(S, x) \geq R(X, x)$. We define $Y_m = l_1, l_2, \cdots, l_m$, $Z_m = r_1, r_2, \cdots, r_{N-m}$, and $X_m = Y_m \bigcup Z_m$, where $j \leq m \leq i$. Note that $S = X_i$ and $X = X_j$. What we need to show is $R(X_i, x) \geq R(X_j, x)$.

We claim that $R(X_{m+1}, x) \geq R(X_m, x)$, where $j \leq m \leq i-1$. We first prove the case where $m = j$. This is equivalent to prove that $P(X_j - \{r_{N-j}\}, l_{j+1}, x) \geq P(X_j - \{r_{N-j}\}, r_{N-j}, x)$. Let G be the set of elements that has already been in the greedy sequence right before the element l_{j+1} is added, when the greedy algorithm is executed to generate S. By the semantics of the greedy algorithm, $G \subseteq Y_j \bigcup Z_i$. So $G \subseteq Y_j \bigcup Z_i \subseteq X_j - \{r_{N-j}\}$. Also $LS(G) = LS(X_j - \{r_{N-j}\}) = Y_j$. Then, since $N > 1$, according to Lemma 6(a), $P(G, l_{j+1}, x) = P(X_j - \{r_{N-j}\}, l_{j+1}, x)$, and according to 6(b), $P(G, r_{N-j}, x) \geq P(X_j - \{r_{N-j}\}, r_{N-j}, x)$. Also $P(G, l_{j+1}, x) \geq P(G, r_{N-j}, x)$, since the greedy algorithm chooses l_{j+1} over r_{N-j}. Consequently, $P(X_j - \{r_{N-j}\}, l_{j+1}, x) \geq P(X_j - \{r_{N-j}\}, r_{N-j}, x)$. This proves $R(X_{m+1}, x) \geq R(X_m, x)$ where $m = j$. Using similar arguments, we can prove $R(X_{m+1}, x) \geq R(X_m, x)$ for $m = j+1, j+2, \cdots, i-1$. Then we get our desired result $R(X_i, x) \geq R(X_j, x)$. □

Lemma 4 *If S is a greedy set defined on the tree x, then $LS(S)$ is a greedy set defined on the tree $x.left$, and $RS(S)$ is a greedy set defined on the tree $x.right$.*

Proof: Suppose that $LS(S)$ consists of k nodes $y_1, y_2, \cdots, y_k$ in the sequence it is generated by the greedy algorithm on S. Let G_i be the set of nodes that have been generated by the greedy algorithm right before the node y_i is generated. Obviously $G_{i-1} \subset G_i$. For any i and any $z \in LS(LF(x)) - G_i$, we know that (a) $P(G_i, z, x) \leq P(G_i, y_i, x)$ since otherwise the algorithm running on tree x would not have chosen y_i over z. It remains to show (b) $P(LS(G_i), z, x.left) \leq P(LS(G_i), y_i, x.left)$. That is, the algorithm running on tree $x.left$ (with parameter k) would have a nonzero probability to generate the sequence $y_1, y_2, \cdots, y_k$. We need to discuss two cases. The first case is when $G_i = \emptyset$. In this case, obviously $i = 1$ and y_1 is the first node inserted. Then according to the Lemma 7(a), $P(G_1, z, x.left) = P(G_1, z, x) - |LF(x)|$ and $P(G_1, y_1, x.left) = P(G_1, y_1, x) - |LF(x)|$. Then (b) follows from (a). The second case is when G_i is nonempty (either $i \neq 1$ or G_1 is not empty). In this case, according to Lemma 7(b) $P(LS(G_i), z, x.left) = P(G_i, z, x)$ and $P(LS(G_i), y_i, x.left) = P(G_i, y_i, x)$, (b) also follows from (a). □

Lemma 5 *Given a tree x, when $|S| > 0$, $R(S, x) = R(LS(S), x.left) + R(RS(S), x.right) + |LF(x)|$.*

Proof: Since $|S| > 0$, the height h of the tree x is at least 1. When the first node in S is introduced, the $(n-h+1)_{th}$ bit of every address under tree x is compromised and there are $|LF(x)|$ of them. So

$R(S,x) = R(S, x.left) + R.(S, x.right) + |LF(x)|$. However, the last $h-1$ bits of all leaf nodes under $x.left$ and $x.right$ are only affected by $LS(S)$ and $LS(R)$ respectively. That is $R(S, x.left) = R(LS(S), x.left)$ and $R(S, x.right) = R(RS(S), x.right)$. The result follows. □

Lemma 6 *Given a tree x and two nonempty leaf sets S and T, and a leaf y of x, the following are true:*

- *(a) If $LS(S) = LS(T)$ and $y \in LS(LF(x))$ (in the left subtree), then $P(S,y,x) = P(T,y,x)$. Similarly if $RS(S) = RS(T)$ and $y \in RS(LF(x))$, then $P(S,y,x) = P(T,y,x)$*
- *(b) If $S \subseteq T$, then $P(S,y,x) \geq P(T,y,x)$.*

Proof: (a) We only prove the first part since the second part follows from the symmetry. Suppose $y \in LS(LF(x))$, then according to Lemma 7(b), $P(S, y, x.left)$ = $P(LS(S), y, x.left)$ $P(T, y, x.left)$ = $P(LS(T), y, x.left)$ since S and T are both nonempty. Then the result follows from the assumption that $LS(S) = LS(T)$. (b) Since $S \subseteq T$, when a new node y is compromised, the set of new bits that are compromised as a consequence in the case when S has been compromised, is a superset of in the case when T has been compromised. The result follows. □

Lemma 7 *Given a tree x, if $y \in LS(LF(x))$, then (a) $P(\emptyset, y, x)$ = $P(\emptyset, y, x.left)$ + $|LF(x)|$ and (b) for any nonempty set S, $P(S,y,x) = P(S, y, x.left) = P(LS(S), y, x.left)$. Similarly, if $y \in RS(LF(x))$, then (c) $P(\emptyset, y, x)$ = $P(\emptyset, y, x.right)$ + $|LF(x)|$ and (c) for any nonempty set S, $P(S,y,x)$ = $P(S, y, x.right)$ = $P(RS(S), y, x.left)$.*

Proof: We only prove (a) and (b) since (c) and (d) follows from the symmetry. Suppose the height of the tree x is h. First recall that n is the number of bits in an IP address. (a) When y is compromised, the $(n-h+1)_{th}$ bit of every address under tree x is compromised, and there are $|LF(x)|$ of them. Also, for any $z \in RS(LF(x))$, its last $h-1$ bits are not compromised because the length of the common prefix between y and z is no longer than $n-h$. Also, the number of bits (among the last $h-1$ bits) in nodes of $LS(LF(x))$ that will be affected by $y's$ being compromised is fully accounted in $P(\emptyset, y, x.left)$. Therefore, $P(\emptyset, y, x)$ = $P(\emptyset, y, x.left)$ + $|LF(x)|$. (b) This case is similar to (a) except that when S is nonempty, the $(n-h+1)_{th}$ bit of every address under tree x has already been compromised before y is compromised. So the term $|LF(x)|$ does not exist in (b). For the second equality, note that none of the leaf nodes in $RS(S)$ will affect the last $h-1$ bits of the leaf nodes under tree $x.left$. □