The Zenith Attack: Vulnerabilities and Countermeasures

Richard Skowyra
Boston University
111 Cummington St
Boston, MA 02215
Email: rskowyra@cs.bu.edu

Azer Bestavros
Boston University
111 Cummington St
Boston, MA 02215
Email: best@bu.edu

Sharon Goldberg
Boston University
111 Cummington St
Boston, MA 02215
Email: goldbe@cs.bu.edu

Abstract—**In this paper we identify and define Zenith attacks, a new class of attacks on content-distribution systems, which seek to expose the popularity (i.e. access frequency) of individual items of content. As the access pattern to most real-world content exhibits Zipf-like characteristics [2], there is a small set of dominating items which account for the majority of accesses. Identifying such items enables an adversary to perform follow up adversarial actions targeting these items, including mounting denial of service attacks, deploying censorship mechanisms, and eavesdropping on or prosecution of the host or recipient. We instantiate a Zenith attack on the Kademlia and Chord structured overlay networks and quantify the cost of such an attack. As a countermeasure to these attacks we propose Crypsis, a system to conceal the lookup frequency of individual keys through aggregation over ranges of the keyspace. Crypsis provides provable security guarentees for concealment of lookup frequency while maintaining logarithmic routing and state bounds.**

I. Introduction

Peer-to-peer systems have emerged as one of the most popular methods for distributing content over the Internet, from largely static movies, music, and software to cached web objects and multicast streams. Due to the global reach of such systems and their lack of central accountability, peer-to-peer networks are useful venues for the publishing and distribution of sensitive material which may be censored, classified, or banned under certain nations' laws. To this end, a significant number of networks and protocols have been proposed (and implemented) to provide some level of anonymity to users. Tor [6] is a well-known example, which relies on onion routing and plausible deniability to conceal the source of traffic originating in its network.

Structured overlay networks in particular have gained popularity in recent years and are deployed in a wide variety of settings, including the trackerless torrent system in Bittorrent [8], the Coral CDN [9], Freenet [4], OpenDHT [22], and Dynamo [11]. Often used as distributed hash tables (DHTs) to index content, these networks provide logarithmic bounds on routing and network state, as well as resilience to high levels of churn and transient node failures. There are a number of anonymizing protocols for structured overlays([19], [27], [17], [20], [28], [24], [10]), which generally provide some combination of origin anonymity and destination k-anonymity.

Content Popularity We argue that origin and destination anonymity are not enough. Anonymity protocols address threat models in which an adversary is seeking to discover the identity of network participants. Consider, however, the case where users are ignored entirely and an adversary directly targets the content that they are accessing with denial of service attacks. Or the case in which anonymity protocols fail (either due to a strong adversary or improper implementation) and identities are exposed. In these cases the popularity of content may become critical information; the impact of denial of service attacks can be maximized and the extent of a censored document's distribution (and perhaps the resulting punishment to the leaker) can be gauged. Furthermore, if popularity can be measured over time, access frequency could potentially be correlated with external information to partially identify users despite the presence of anonymizing protocols. Peak access frequencies may, for example, fall within a few probable time zones, hinting at geographic or national interest in a particular item of content.

Zenith Attacks A second line of defense is needed. Content popularity must itself be obfuscated in order to mitigate the damage described above. We call attacks that reveal content popularity Zenith attacks, and describe them in more detail below. To demonstrate their practicality we instantiate a Zenith attack on the popular Chord and Kademlia structured overlay protocols. Furthermore, we show that the resources (quantified in the number of adversarial nodes) required to mount even the most naive Zenith attacks are extremely low, and well within the capabilities of almost any real-world attacker. This is largely due to the deterministic routing which characterizes structured overlays; given the distance between two nodes, it is trivial to calculate the expected proportion of traffic destined for one node that will be seen by the other.

Crypsis Motivated by the ease with which an adversary can determine popular content, we propose Crypsis: a system which conceals individual key lookup frequencies in structured overlay networks by aggregating all lookups over a segment of the total keyspace. Crypsis completely divides the keyspace into a number of nonoverlapping segments and uses standard iterative lookups to contact a randomly chosen node in the segment hosting a desired key. Within that segment a separate query protocol locates the key itself, without revealing information on the key being looked up to any nodes but the host for that key. We define and prove Crypsis' security properties in the presence of a weak global adversary, and show that it maintains (within a small constant factor) the logarithmic routing and state bounds of the original structured overlay on which it is built.

Organization The remainder of this paper is organized as follows. Section II describes the Zenith attack, and presents experimental results on the vulnerability of Chord and Kademlia to naive versions of the attack. Section III describes our threat model in detail. Section IV describes Crypsis' architecture and analyzes its security properties. Section V discusses what an adversary can conclude from information that Crypsis does not conceal. Section VI discusses related work. Finally, Section VII presents our conclusions and future work.

II. The Zenith Attack

We define Zenith attacks as a general class of attacks against content distribution systems which exploit routing and lookup information to reveal content popularity. The specific mechanisms that a Zenith attack employs are dependent on the target system. For example, in unstructured overlay networks like Gnutella an attack may be mounted by monitoring flooded searches or queries throughout the network, or by monitoring the cache lifetime and replication rate of content on adversarial nodes over time. In this paper, however, we focus on an instance of the Zenith attack against structured overlay networks. These overlays are one of the most likely victims of such an attack, as they often form the backbone of the distribution systems of choice for content which needs to be distributed anonymously and/or discreetly (e.g. Freenet, Nisan, Bittorrent, I2P, etc.)

The Naive Attack Zenith attacks on structured overlays exploit the fact that routing is key-based, and that keys have a one-to-one mapping with items of content. To mount a Zenith attack, we assume an adversary adds nodes to the network at some fixed cost. As functioning peers, these nodes will serve as intermediary routers for some proportion of the messages that are routed through the system. The adversary maintains a frequency count for each request for an item of content (i.e. key) that its sees. Once the adversary judges that enough data has been collected, the most popular items are identified by comparison of frequency counts. We refer to this process as a naive Zenith attack.

Exploiting Structure More advanced attacks may take advantage of network structure in order to exploit topology-related effects and improve the accuracy of results. Since structured overlay networks route over a keyspace using a distance metric, the proportion of lookups for a specific key which pass through a particular node is quantifiable. Below, we provide an example using Chord which allows an adversary to create a scaling vector with which to augment its frequency count. More accurate techniques utilizing, e.g., noise reduction and signal analysis are certainly possible, but experimental results (see Section II-B) indicate little need for anything beyond the naive attack.

Attack Amplification As a final note, Zenith attacks can be further augmented by attacks which concentrate the amount of traffic passing through adversarial nodes. Sybil attacks [7], in which a single adversary cheaply instantiates an extremely large number of nodes, are particularly effective. Since node IDs are assigned via collision-resistant hash algorithms, there is a very high probability that Sybil nodes will be distributed over the entire keyspace. As a result, many paths will intersect with at least one of the adversary's nodes. The Eclipse attack [3], is also useful to amplify Zenith attacks. This technique uses a set of colluding adversarial nodes to manipulate the routing tables of honest nodes to ensure that adversarial nodes are intermediate routers for a large portion of network traffic. The exact mechanism is somewhat protocol dependent, but generally relies on corrupting the neighbour-selection algorithm used by honest peers.

A. Chord Zenith Attack

As a proof of concept, we present a Zenith attack against a Chord network [25] which leverages its distance metric to increase the accuracy of scaling results. Chord places nodes in arithmetic order on a ring with 2^m distinct positions, each of which is identified by an m-bit hash of their IP address, public key, etc. Each node is responsible for content which hashes to the range between its identifier and that of its immediate predecessor. In order to facilitate fast lookups, each node n maintains a routing table of entries that point to the successor node of a key equal to $n + 2^{i-1}$, where i is the i^{th} entry in the table. Messages originating at n or passing through n will be sent to the successor node of the routing table entry closest to, but not exceeding, the destination key.

Exploiting Distance Note that messages are always routed via the longest possible hop, and that distances are asymmetric. From this we can conclude that a node i (where i is the network identifier) will see a portion of queries for a key k proportional to D_{ik}, the distance from i to k, where:

$$D_{ik} = \frac{1}{2^{\lceil log(k-i)+1 \rceil}} \qquad (1)$$

An adversary can then add nodes to a Chord network and record frequency counts as usual. Before aggregating the measurements, however, each key in each count can be scaled by the distance between the measuring node and the key. This will allow a closer approximation of the actual key popularity.

B. Vulnerability Assessment

Of the many structured overlay networks in use today, Chord [25] and Kademlia [16] (or variants thereof) are two of the more widely deployed. Chord is used in the CFS distributed storage system [5], and Kademlia variants are used by both the Gnutella network and Bittorrent.

Setup We simulated Chord and Kademlia over a 1000 node network in order to assess the practicality and efficacy of

Zenith attacks. All of our experiments use a trace-driven dataset with a power-law distribution derived from the 2009 Blogspot memetracker dataset [15], of which there are 56888 unique items. As indiciated by the work cited in section **??**, content popularity in many real-world settings follows a power law. We chose to use a dataset which tracks quote activation on blogs, as this easily generalizes to the case in which users (blogs) express interest in a particular piece of information (quote).

An adversary controlling a fraction of randomly selected nodes attempts to identify the top 10 items. We chose to use the most naive Zenith attack: the adversary merely maintains a count of lookups that is seen for each key, and outputs the 10 highest-count keys at the end of the simulation. Results for are shown in Figure 1. Each datapoint represents fifty trials.

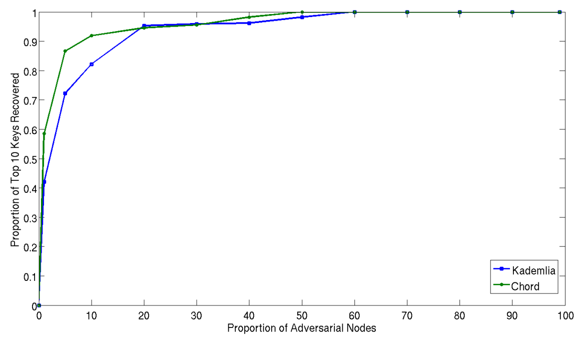

Fig. 1. Key Popularity in Chord and Kademlia

Results Note in Figure 1 that for Chord, approximately 90%, and for Kademlia approximately 80%, of the most 10 popular keys are recovered with 10 nodes under adversarial control. This suggests that popularity-revealing attacks are both practical and cheap to mount. Furthermore, the adversary for these experiments was confined to a naive attack. By exploiting the deterministic properties of DHTs (e.g. key distance metrics), a significant amount of noise could be removed from lookup frequency counts and even fewer adversarial nodes would be necessary. Since both Chord and Kademlia are deployed extensively in the wild, these experiments motivate the development of a popularity-concealing structured overlay network.

III. Threat Model

Given a structured overlay network with $|N|$ nodes and $|K|$ m-bit keys, we define the following terms:

- N - The set of all nodes in the network.
- N' - The set of nodes controlled by the adversary. Note that $N' \subset N$.
- K - The set of all possible keys. Note that $|K| = 2^m$.
- P - The secret vector of key lookup probabilities for all keys $k \in K$. These probabilities can be thought of as being derived from key lookup frequency counts over a unit of time.
- $L = \{k|P_k > \gamma\}$ - The set of most popular keys for some threshold of lookup probability, γ.

Adversarial Powers We assume that the adversary, A, is a passive global observer able to see all messages sent over the network. We further add the power to control a subset of nodes N'. These nodes are assumed to be colluding, and may deviate from the protocol if they choose. The adversary may add new nodes to N' at a fixed, per-node cost. They may not, however, choose where new nodes are placed (i.e. choose the network identifier for new nodes), since structured overlays use a cryptographic hash of IP address or public key to assign identifiers.

Adversarial Objectives The adversary is further assumed to possess a prior probability distribution P' over all keys $k \in K$ and a distance function $f(X, Y)$ such that $f(X, Y) = 0 \leftrightarrow X = Y$, where X and Y are random variables. We define f loosely in order to allow the adversary to choose the most beneficial distance metric for its purposes. The adversary will attempt to form a posterior probability distribution $\hat{P}$ such that:

1) Given the prior P'_k, $f(\hat{P}_k, P_k)$ is minimized for an adversarially chosen target key k.
2) $|L - L'|$ is minimized, where $L' = \left\{k|\hat{P}_k > \gamma\right\}$.

In doing so, the cost function $|N'|$ should be minimized.

Estimating Key Popularity For the first objective the adversary, depending on its identity and aims, may have one or more keys which it is interested in and has some expectation about their popularity (defined as lookup probability). A government trying to assess the interest in the web address of a leaked document that made headlines, for example, can reasonably expect the key which identifies that address to be popular. This expectation is made prior to joining the network. A's objective is to arrive (as cheaply as possible) at a modified expected value for the key of interest, which is as close as possible to the actual popularity of the key.

Finding Popular Keys For the second objective, A is interested not in determining the popularity of a specific key, but instead in attempting to identify which keys are the most popular. An example would be a purely malicious attacker interested simply in maximizing damage against the network as a whole. Again, some prior expectation of a popular key set is assumed to be present prior to joining the network. If the network is often used to exchange media files, for example, then hit songs, television shows, movies, etc. could be expected to be present.

Adversarial Success Note that none of the above objectives has a clear notion of success or failure. Rather, each objective can be considered a best-effort attempt, with a success rate proportional to the distance between A's posterior distribution and the actual distribution of key lookup probabilities, with respect to some adversarially chosen distance function. Crypsis does not claim to eliminate all information leakage which could lead to adversarial success. It does, however, place strong bounds on what information can be leaked. This topic is addressed in more detail below.

IV. Crypsis

Crypsis is designed to address the above threat model via a process of one-way aggregation (i.e. aggregation such that disaggregation is intractable). Specifically, the protocol leaks the sum of all lookup requests for a segment of the keyspace, while concealing the lookup frequency of any individual key in that segment from all observers except the single node hosting that key. Intuitively, the popularity of an individual key will have an upper bound equal to the aggregate popularity of its keyspace segment. Similarly, trying to determine the most popular keys will ultimately only allow the determination of the most popular segments of the keyspace. Average-case routing and state bounds are both $O(log\ N)$; the constant varies with respect to a system parameter, μ, described in Section IV-A2.

A. Architecture

Crypsis is a two-tier system, similar in this respect to Herbivore [10] and Agyaat [24]. The first tier is a global structured overlay network, represented in Figure 3 as a circular keyspace onto which nodes and items of content are hashed. This layer provides connectivity and resolution of coarse-grained lookups. The second tier consists of unstructured complete mesh networks localized over contiguous nonoverlapping regions of the keyspace, represented in Figure 3 as 3-cliques around the edge of the ring. This layer provides popularity concealment and resolution of fine-grained lookups. All members of the network are present in, and maintain separate routing tables for, each tier.

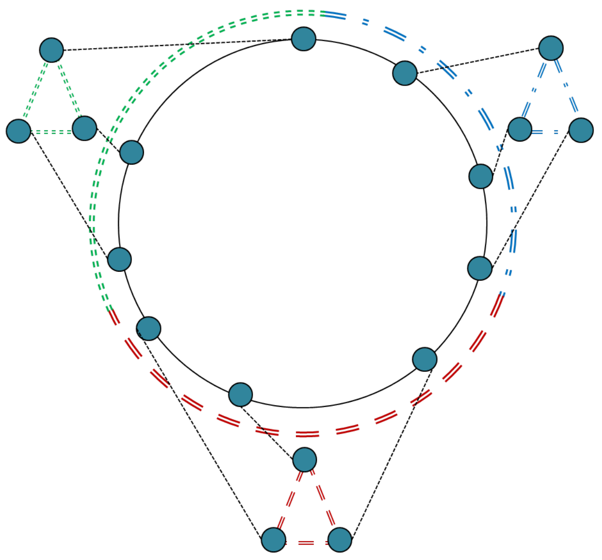

Fig. 2. Crypsis Network

1) Global Overlay: This layer of the Crypsis network is a standard structured overlay. While this paper assumes that the overlay network is Chord-like [25], there is no reason that a different topology (e.g. a Kademlia-like tree [16]) could not be used instead. We merely require the presence of a consistently hashed, closed keyspace. Nodes use this network for two functions: joining the local mesh which their position in the keyspace makes them a member of, and contacting the remote mesh network containing a target key.

Node Joins When a new node joins a Crypsis network, it bootstraps into the global overlay using that overlay's joining mechanism. Once an identifier is assigned to the node, it joins the local mesh network responsible for the keyspace range containing its identifier. Since keyspace ranges are nonoverlapping and contiguous, bootstrap information can easily be obtained from the node's immediate successor or predecessor (or both) in the global overlay. The only time this would not be true is if the new node is the sole member of a local mesh, at which point there is no need to obtain bootstrapping information.

Backbone Routing The global overlay is also used to contact the remote mesh network responsible for a target key in a simple two step process:

1) A querying node first computes the keyspace range containing the target key. This is a deterministic computation that can be performed locally, and is described in Section IV-B.
2) The node then generates a random number in that range and performs an iterative lookup for that number over the global overlay. When the hosting node is found, it provides the querying node with a copy of its local mesh network routing table.

In this way, the global overlay essentially acts a backbone connecting members of different mesh networks.

2) Mesh Networks: Mesh networks in Crypsis are cliques of all nodes whose identifiers fall within a particular range. Ranges do not overlap, and every node is a member of either one or two meshes. [1] These networks are used to conceal the actual target of a query. In essence, a query for any key in a particular mesh network generates masking traffic to all members of the network, such that the real query is indistinguishable from a masking query. An observer can determine only that the query was for some key in the range assigned to that mesh. Section IV-B describes the protocol in detail.

The Number of Meshes In order to preserve logarithmic routing and state bounds, the number of mesh networks in a Crypsis network, μ, must be a function of the network population. Specifically, we set:

$$\mu = \frac{\eta}{log\ \eta} \tag{2}$$

[1]A node is a member of a single mesh so long as the range of keys it is responsible for falls entirely within that segment of the keyspace. If its range spans two segments, it will maintain two disjoint routing tables.

where η is an estimate of the network population. Obviously, efficiently setting a system parameter based on an estimate of a dynamically changing quantity is nontrivial. We provide a more in-depth discussion of how to estimate the network population and avoid frequent reconfigurations in Section V-A.

Tuning Concealment While Crypsis seeks to preserve the beneficial properties of a structured overlay, μ can alternatively be seen as a tuning parameter that trades routing and state efficiency for security. Decoupling it from the network population would create a fixed number of unstructured sub-networks connected by a structured overlay backbone. As the number of sub-networks decreases, the amount of keys aggregated together increases, and less can be determined about the overall shape of the key popularity distribution. Simulations demonstrating this effect are presented in Section IV-D.

3) Trusted Infrastructure: Like the vast majority of structured overlay security protocols, Crypsis requires the presence of an offline Certificate Authority (CA). As we discuss in Section IV-C, the CA is needed to prevent man-in-the-middle attacks during key exchange.

Root Verification In addition to an offline CA, Crypsis relies on the assumption that overlay nodes cannot lie about which region of the keyspace they are responsible for. This assumption does not hold in basic structured overlays like Chord and Kademlia. Nodes in these networks can claim responsibility for arbitrary regions of the key space, opening the possibility for redirection, identity spoofing, and denial or service attacks.

The Myrmic [26] root verification protocol is explicitly designed to address this. Myrmic assumes the existence of an offline Certificate Authority, as well as an online Neighborhood Authority (NA). NAs assign keyspace regions to newly joined overlay nodes and issue a signed certificate (witnessed by the new node's neighbors) attesting to the binding. Routing is iterative, and a queried node must present its certificate to the querying node at each hop.

While Crypsis only requires a generic root verification protocol, the authors are unaware of any other work comparable to Myrmic. We therefore assume that Myrmic is deployed on the global overlay network.

B. Query Protocol

The objective of Crypsis is to conceal key popularity by aggregating all queries made to one segment of the keyspace. As discussed above, the global structured overlay acts as backbone connecting local unstructured cliques, each of which is responsible for all keys in its segment. Our query protocol first navigates the global overlay using a dummy key which is in the same segment as the target key. Next, every node in the mesh network responsible for the target key is queried over a secure channel. The key's host is asked for the key, while other hosts are asked for a dummy key. Since Crypsis is designed to conceal key popularity, we handle only two types of query:

- $get(k)$ - Returns the item bound to key k.
- $put(k,\ v)$ - Inserts the item identified by v into the network, bound to key k.

The following protocol is invoked whenever either type of query is launched for a target key, k, in the keyspace, by any node, A, in the network:

1) A: Determine the keyspace segment, S_k, responsible for the target key:
$$S_k = \frac{k\mu}{2^b}$$
where b is the number of bits in the keyspace.
2) A: Compute $range(S_k)$ = $[\frac{S_k * 2^b}{\mu}, \frac{(S_k+1)*2^b}{\mu}]$
3) A: Select $k' \in_R range(S_k)$
4) A: Find the node, B, responsible for k' using an iterative DHT lookup. For security purposes, assume k' is broadcast to all members of the network.
5) $A \rightarrow B$: (RRT), where RRT is a message indicating a request for the receiver's local mesh routing table.
6) $B \rightarrow A$: (RT), where RT is a routing table of all IP addresses in S_k.
7) $\forall n \in RT$:
   a) $A \leftrightarrow n$: Negotiate symmetric key S using an authenticated Diffie-Hellman key exchange.
   b) $A \rightarrow n$: $(Range)_S$
   c) $n \rightarrow A$: $(rangeStart, rangeEnd)_S$
   d) If $rangeStart \leq k \leq rangeEnd$:
      i) $A \rightarrow n$: $(k, v, get/put)_S$ where v is the fixed-length value associated with the specified key.
   e) Else:
      i) Select $rangeStart \leq k'' \leq rangeEnd$ uniformly at random.
      ii) $A \rightarrow n$: $(k'', v, get)_S$
   f) $n \rightarrow A$: $(v)_S$

Consequences Note in the above protocol that all communication taking place between a querying node and a node in a mesh network uses a secure channel over which a fixed number of fixed-sized messages are exchanged regardless of message content. While necessary to ensure that the target key is masked, it has three important side-effects:

- In order to establish the secure channel (Step 7a), a public key operation is required in order to prevent man-in-the-middle attacks during key negotiation. This is the only step which uses public-key encryption, it but does require a PKI in order to ensure secure binding of public keys and identities.
- Items in a Crypsis network are limited to a fixed maximum size. Anything less than this size must be padded, and anything larger either truncated or split over multiple keys.
- Message length remains constant regardless of query type or success. This incurs a certain amount of wasted bandwidth, but is necessary to effectively mask the genuine query.

State Bounds Crypsis requires nodes to keep routing state for both the global overlay and their local mesh network(s).

The amount of state required for the former is $O(log\ n)$ by definition, with specific constants varying by overlay choice. The latter is at worst $O(2m)$, where m is the size of the local complete mesh network. Note, however, that since we have set $\mu = \frac{\eta}{log\ \eta}$, it follows that $m = \frac{n}{\mu}$. Therefore $m \in O(log\ n)$ as long as η is within an order of magnitude of the network population (achievable using algorithms described in [13] and [14]) and nodes are distributed uniformly. Then the total state required remains $O(log\ n)$.

Routing Bounds Routing bounds are calculated similarly. The number of hops from a querying node to the remote mesh network responsible for a target key is $O(log\ n)$ as it iteratively traverses the global structured overlay. This process covers steps 1-6 in the above protocol. Once a routing table for the mesh is received (Step 6), the querying node contacts each member of the target mesh network. Three messages are sent per node: a key exchange message (step 7a), a range query (step 7b), and a get/put query (step 7d or 7e). The total number of hops is therefore $O(log\ n) + O(3m)$. As shown above, $m \in O(log\ n)$, so routing remains in $O(log\ n)$.

C. Security Analysis

In this section we discuss the security properties of the Crypsis protocol. Since Crypsis is built over a structured overlay network, it inherits a number of properties (e.g. key-based routing, Sybil attack vulnerabilities, etc.) which are relevant to network security. With one exception, these concerns lie outside the scope of our threat model and are not addressed below. Note, however, that a significant amount of work has been done in recent years on addressing these issues. Crypsis is complementary to such work, and there is no reason why it could not be implemented on a secured overlay. All theorems presented in this section are proved in the appendix. Sketches are given to provide insight into the intuition behind the proof.

Range Reporting The exception mentioned above forms a key assumption that is necessary for Myrmic's security guarantees to hold:

Assumption 4.1: Nodes do not lie about the key range they are responsible for.

Myrmic, described in Section IV-A3, implements a certificate-based system that ensures the above assumption will hold with high probability, given a limited number of colluding adversaries and the prescence of a CA. We use this assumption to prove a key claim:

Claim 4.1: Receiving a false mesh network routing table will not leak any more information than is leaked in normal protocol operation.

The proof, in Appendix A, is a case analysis of a querier's possible reactions given an arbitrary set of nodes in its routing table. Nodes which do not host the querier's target learn only that they don't host the key being looked up, and the node (which would be present in the correct routing table) that does host the querier's target learns the key that the querier is looking up.

Global Observer We first discuss Crypsis' security versus a passive global observer able to see all messages sent over the network but not control any nodes. We use a standard cryptographic game-based approach to analyse security. To do this, we model the adversarial global observer's interaction with the Crypsis system (which for the purpose of this particular security analysis, consists of only honest nodes) as a game between an adversary and a challenger, as follows:

- A single challenger controls all nodes on the network.
- There is a nonempty set K which contains all keys in the network
- There is a nonempty set S which contains all mesh network identifiers
- There is a public function $F : k \in K \rightarrow s \in S$ which maps all keys (for both nodes and items) to mesh networks, such that each key is mapped to a single mesh network.
- One conversation in the protocol (i.e. the interaction between a querying node and all nodes in the target key's mesh network, as described in Section 4.2) is modelled as a series of tuples of the form (seq, d, m) where:
    1) seq is a sequence number which denotes all tuples which are part of that conversation. The number of tuples in one conversation is a function of F, and is equal to the number of nodes mapped to the same mesh network identifier as the target key.
    2) $d = F(k)$ represents the mesh network containing the key being looked up. This corresponds to the dummy key, k_d used in protocol step 4. Since k_d is chosen independently of k subject to the constraint that $F(k_d) = F(k)$, publishing the mesh network id itself is an equivalent representation.
    3) $m = E(k, 0, get/put)$ represents the encrypted request for a key k made in step 7d or 7e. Since k is the only field which isn't static, we use the shorthand notation $m = E(k)$.

    All other protocol interactions are either local and do not involve any interaction, do not contain information relevant to recovery of the key being queried, or are addressed separately below.

Given these assumptions, define the following game:

1) The adversary, A, chooses the following values and gives them to the challenger:
    a) A nonempty set K of keys.
    b) A positive integer a, the transcript length.
2) The challenger generates a transcript T of length a and gives it to the adversary. A transcript represents a series of key lookups made over a Crypsis network. T is a set of tuples of the form (seq, d, m) described above. For each protocol iteration, the challenger draws a key $k \in K$ with probability $\frac{1}{|K|}$ and sets $d = F(k)$. The number of tuples generated is equal to the number of nodes which share the same mesh identifer as k. For each tuple, $m = E(k)$ iff the range of the node being queried contains k, otherwise $m = E(k')$, where $k' \in K$

is drawn uniformly at random from the range of the node being queried.

3) The adversary chooses and outputs a key $t \in K$ and tuple $w \in T$ s.t. A believes that $w_m = E(t)$ (i.e. the tuple w contains an encrypted request m for the target key t).

Definition 4.1: For the above game, we say a Crypsis interaction is secure if a computationally bounded adversary can correctly distinguish that for a chosen $t \in K$ and $m \in T$, $w_m = E(t)$, with the following probability:

$$|P(A_c((w_m = t) = t)) - P(A_c((w_m = \neg t) = t))| \leq \epsilon_c \quad (3)$$

Theorem 4.1: If all components of step 7 in the Crypsis query protocol use a semantically secure encryption scheme, then the Crypsis protocol meets the security definition with respect to a global observer.

We prove this theorem in the contrapositive in Appendix A, using a reduction which shows that if there exists an adversary A_c which can win the Crypsis game, then we can construct an adversary A_e which can break semantically secure encryption.

Theorem 4.2: For n iterations of the above game, quantify the number of correct adversarial outputs of the form $(w_m = t)$ for a chosen target key t as X. Then X is subject to the following upper Chernoff bound:

$$P(X \geq (1+\delta)\frac{n}{|M|} \leq e^{-\delta^2 \frac{n}{|M|*3}} \text{ and } 0 \leq \delta \leq 1 \quad (4)$$

We prove this in Appendix A by defining a binary indicator variable which tracks adversarial successes, and showing that it follows a binomial distribution.

Active Adversary An active adversary is one which both has global observer capabilities, and controls some fraction of nodes participating in the network. These nodes may collude and deviate from protocol. We do not offer a formal cryptographic security argument against this adversary. Instead, we now discuss why the active adversary gains only a small advantage over the passive observer. Namely, the active adversary learns:

- The popularity (lookup frequency) of keys hosted by adversarial nodes.
- The fact that a key being queried which is not hosted by an adversarial node cannot be in the set of keys hosted by adversarial nodes.

We use an exhaustive case analysis in Appendix A to argue this claim. Ultimately, an adversary which does not deviate from protocol gains complete information about the keys it hosts. For those it does not host, it learns only that it does not host them. An adversary which does deviate from protocol can cause denial of service, but not gain any new information. See the appendix for the complete argument.

D. Simulation

In order to confirm our analytical results, we simulated Crypsis over a 1000-node Chord network with trace-driven queries derived from [15]. This dataset contains traces of quote activations over blogs hosted by blogspot during 2008. There are 56888 unique quotes, each of which was mapped into a 160-bit keyspace.

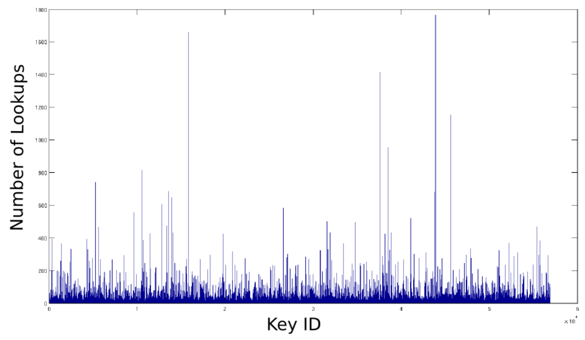

Fig. 3. Actual key lookup frequencies

Using this data as a source of keys for $get()$ queries, we set the number of mesh networks, $\mu = 10/100/1000$ and plotted the observable key lookup frequencies from the perspective of a passive global observer. The results are presented in Figures 4-6.

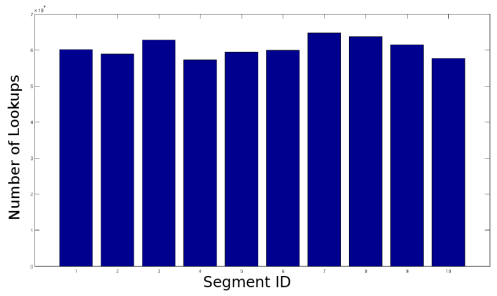

Fig. 4. Keyspace segmentation at $\mu = 10$.

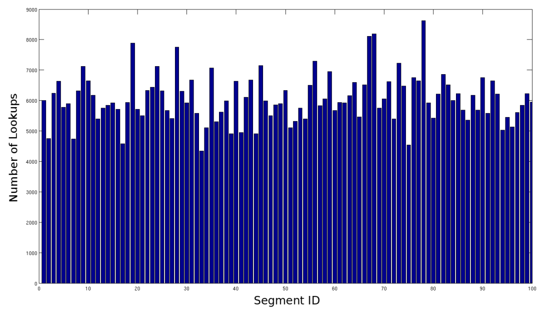

Fig. 5. Keyspace segmentation at $\mu = 100$.

Observe that as the number of mesh networks increases, the shape of the underlying distribution is revealed with higher

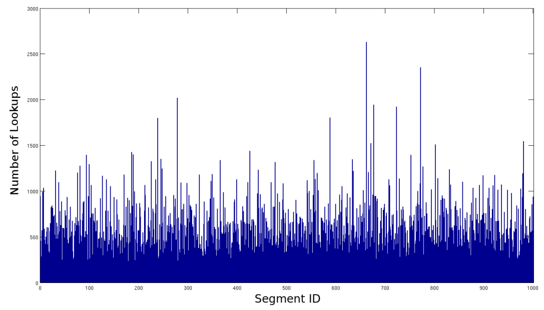

Fig. 6. Keyspace segmentation at $\mu = 1000$.

and higher resolution. This demonstrates a tradeoff between performance and security. Setting μ to a small value will map a larger number of keys into each mesh network and more thoroughly conceal which mesh networks have the most (un)popular keys. However, this will also result in more nodes being mapped to each mesh, and therefore increase the routing and state overhead per node. Setting μ to a high value will have the opposite effect. As discussed in Section IV-A2, we set μ to be a function of population. This allows for the preservation of logarithmic routing and state bounds.

V. Discussion

A. Estimating Network Population

Accuracy Recall that the number of mesh networks, $\mu = \frac{\eta}{log\eta}$, where η is an estimate of the network population. η can be efficiently computed using either active or passive measurement techniques, as described in [13] and [14]. The accuracy of these estimates is sufficiently fine grained, as our only concern is keeping routing bounds and state within a logarithmic amount of the actual population. Note that the value of μ (and therefore the ranges assigned to each mesh network) must be equal across all nodes in order to maintain routing correctness, as otherwise mesh ranges may overlap. However, due to dynamic network populations, it is impossible to guarantee all nodes will independently arrive at the same value of η, even in networks with no adversary. Clearly, some protocol for agreeing on the value of η is needed.

Who Computes η? Using a single node to compute η raises issues of trust and the possibility of a single point of failure. If, for example, an adversary compromises the estimator, it may arbitrarily set η and therefore μ. The adversary could set μ to be so high that every node is isolated in its own mesh, or report different values to different nodes and destroy routing correctness. One alternative is a distributed agreement protocol, ideally one resilient to Byzantine failures. Such a protocol would allow nodes in the network to reach a consensus on η without a central authority. However, these protocols add a nontrivial amount of complexity and overhead. We propose a different solution. Crypsis already uses Myrmic to provide root verification. Its Neighborhood Authorities form a stateless, trusted network which does not participate in normal routing. Neighborhood Authorities are ideally suited to compute and distribute a common value for μ. Distribution may be either via piggybacking on normal messages, or by inserting signed updates into the existing structured overlay at a well-known location. If another root verification system is being used which does not employ online, trusted nodes, the CA can perform a similar role or instantiate a trusted set of stateless nodes to do so.

Updates Despite the fact that network populations are dynamic, updates to η must be infrequent, as changing the value of μ prompts a reconfiguration of all mesh networks. At first this seems like a significant design flaw, but there are mitigating factors:

- The coarse granularity of η allows updates to μ to be made on the order of hours or days in the steady-state case.
- The value of μ has no effect routing correctness; only on performance. This allows updates to be delayed in times of high network traffic or periods of heavy node churn (e.g. flash crowds).

B. Practical Considerations

We have, so far, considered the problem of popularity-revealing attacks and proposed a system using one-way aggregation to bound the information on key lookup frequency which can be leaked by the network. Some discussion of the practical implications of this bound may be necessary. Specifically, we would like to address the question of what can be done with observations of aggregate popularity. This is a necessarily informal treatment, as an adversary's ultimate objectives (e.g. denial of service, censorship, prosecution of participants) and strategy may vary.

Relative Popularity Consider the example of an adversary which has a prior distribution such that a particular key is expected to be extremely popular (e.g. a key that maps to a popular leaked document or pirated blockbuster).

Upon joining and monitoring a Crypsis network, that adversary can determine how popular the keyspace segment containing that key is relative to all other segments. If the relative popularity is low, the adversary may conclude that the item is not present in the network at all, or that its existence in the network is not a well-known fact, etc., and adjust its posterior distribution accordingly. A high relative popularity, conversely, may do little to alter the adversary's expectations. The segment is popular, which may indicate the key in question is popular, or that a number of keys are moderately popular, etc. The adversary may draw any number of conclusions.

In other words, high relative popularity of a keyspace segment does not constitute sufficient evidence that the item itself is popular; only that the keyspace segment containing that item is popular.

Deniable Popularity Essentially, Crypsis leaks information about the distribution of lookup requests (e.g. shape) while concealing enough information on individual key lookup frequencies to provide a kind of plausible deniability with respect to popularity. This feature is useful in contexts where a penalty for providing access to an item of content is tied to the number of accesses made to that item, e.g. damages for copyright infringement.

VI. Related Work

To the best of our knowledge, there has been no published research on concealing key lookup frequencies in structured overlay networks. Related work can be divided into two categories: the evidence of content following power law distributions which motivated our work, and research on anonymous communication over structured overlay networks that is similar to our work.

A. Power-Law Popularity

A large body of work has been published on the power-law distribution of Internet services. Breslau et al. found that the distribution of web page requests to web caches follows a Zipf-like rule [2]. Jung et al. find a similar heavy-tailed distribution to be present for DNS resolution queries [12]. Ripeanu et al. analyze the Gnutella network and conclude that its topology also follows a power law-like distribution [23]. Beehive [21], a fast lookup service for structured overlays, is designed around power-law query distributions. Due to the sheer number of services which follow power laws, in addition to the specific applicability of Beehive, DNS queries, and file sharing to structured overlays, we feel that exposure of the few dominating keys constitutes a significant adversarial advantage. By concentrating attacks on these keys only, an adversary can dramatically amplify the ratio of damage caused per resources spent.

B. Anonymous Structured Overlays

There has been a significant amount of work on anonymity in sturctured overlay networks. Borisov [1] and Mislove et al. [18] demonstrated that complete origin anonymity is possible in Chord and other DHTs via random walks and probabilistic forwarding, respectively. [1] also concludes that complete destination anonymity is impossible without sacrificing the routing guarantees provided by structured overlay networks.

Various attempts have been made to address the problem of destination anonymity, which is related to the problem of concealing key lookup frequencies. (Since content and nodes share the same address space, knowing the destination of a message exposes information about the key being looked up, and vice versa.) We stress that the problems are not equivalent, however, as destination anonymity does not necessarily require the key being looked up to be hidden; only the node responsible for that key. Similarly, strong concealment of key lookup frequencies may still leak information on the destination of a message.

Salsa [19] divides the keyspace into a contiguous series of smaller circular keyspaces using namespace partitioning. Nodes employ redundant lookups and a variant of random circuits to provide mutual anonymity for networks with fewer than 20% of their population under adversarial control.

SurePath [27] performs a variant of onion routing over a DHT to provide both origin and destination anonymity. It uses the concept of relay set anchors, which are DHT nodes chosen by hashing a node identifier, secret key, and current time. These nodes are used to onion route a message from origin to destination, relying on both the origin and destination plausibly being onion routers and not endpoints.

Torsk [17], designed as a distributed replacement to the Tor directory service, also uses a variant of onion routing combined with cover traffic and root verification (see Section 4) to implement a form of origin anonymity. The system is primarily designed to locate a series of onion routers for tunnel construction in an anonymous fashion, and is not concerned with hiding the content of lookups; only their origin.

Nisan [20] is a modified Chord-like system designed to be resilient to many common attacks on structured overlay networks. It uses an iterative, aggregated search function to provide redundant lookup capabilities. Nisan hides the value of an aggregate search query by asking intermediate nodes for their entire routing table, rather than the entry closest to the queried ID. Bounds checking is used for each returned routing table, in order to prevent malicious nodes from responding with false route information. Note that, while Nisan hides query values from intermediate nodes, this does not conceal key lookup frequency from a weak global adversary. Such an adversary would be aware that the sequence of nodes being queried converges on a single node.

Cashmere [28] applies the principle of Chaum-Mixes to DHTs. It divides nodes in a structured overlay into sets, and treats each set as one relay point in a routing path. The destination may be in any of the sets present on the path, and onion encryption is used to ensure the relay groups cannot infer previous or future hops.

Agyaat [24] is architecturally similar to Crypsis, but differs in both intent and a few key design choices. It uses random meshes ('clouds') of nodes to conceal origin and destination. Lookups are mapped to cloud identifiers, and the query is flooded to all members of the cloud. The query itself is not concealed, and the threat model being considered uses an adversary attempting to discover the originating or termination node for a query; no consideration is given to the query itself.

Herbivore [10] uses a similar approach, leveraging a global DHT backbone for communication between clusters of nodes. In this work, however, each cluster is a dining cryptographer network organized into a star topology. Herbivore provides k-anonymity, with k being the size of an individual's cluster. Key lookup frequency is not concealed in Herbivore, and indeed cannot be given the local network protocol that the system relies on.

VII. Conclusions

We have defined the Zenith attack, a class of attack which reveals the popularity of content in a structured overlay network. We launched a Zenith attack against the Chord and Kademlia protocols, and found that the cost of doing so is extremely low. To mitigate the impact of this attack we proposed, analyzed, and evaluated Crypsis, a popularity concealment protocol that implements one-way aggregation of queries over segments of the keyspace, while maintaining logarithmic routing and state bounds. Crypsis uses a structured overlay as a backbone connecting complete mesh networks, each of which is responsible for a segment of the total keyspace. A special query protocol which conceals query type and destination is used to limit popularity-related information to data on the keyspace segment containing a target key.

Given recent high-profile leaks of sensitive information, as well as ongoing debates related to file sharing and network neutrality, the need for a sense of plausible deniability with respect to content popularity seems clear. Since no prior work has been done on popularity concealment, we hope that Crypsis will serve as a motivation to explore this aspect of network security in both structured and unstructured overlays.

VIII. Acknowledgements

This research was supported by NSF Grants 0735974, 0820138, 0952145, and 1012798.

References

[1] N. Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays*. PhD thesis, University of California, Berkeley, 2005.

[2] L. Breslau, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume XX, pages 126–134 vol.1. Ieee, 1999.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review*, 36(SI):299–314, 2002.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[5] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, volume 35, pages 202–215. ACM, Dec. 2001.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, page 21. USENIX Association, 2004.

[7] J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[8] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user dht. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement - IMC '07*, page 129, New York, New York, USA, 2007. ACM Press.

[9] M. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with Coral. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*, page 18. USENIX Association, 2004.

[10] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication, 2003.

[11] D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazons highly available key-value store. In *In Proc. SOSP*, pages 205–220. Citeseer, 2007.

[12] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop - IMW '01*, page 153, New York, New York, USA, 2001. ACM Press.

[13] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers. Decentralized Schemes for Size Estimation in Large and Dynamic Groups. *Fourth IEEE International Symposium on Network Computing and Applications*, pages 41–48, 2005.

[14] E. Le Merrer, A.-M. Kermarrec, and L. Massoulie. Peer to peer size estimation in large and dynamic networks: A comparative study. *2006 15th IEEE International Conference on High Performance Distributed Computing*, pages 7–17, 2006.

[15] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, New York, New York, USA, 2009. Citeseer.

[16] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[17] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. In *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, page 590, New York, New York, USA, 2009. ACM Press.

[18] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 30. ACM, 2004.

[19] A. Nambiar and M. Wright. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 17–26. ACM, 2006.

[20] A. Panchenko, S. Richter, and A. Rache. Nisan: Network information service for anonymization networks. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 141–150. ACM, 2009.

[21] V. Ramasubramanian and E. Sirer. Beehive: O (1) lookup performance for power-law query distributions in peer-to-peer overlays. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation*, number 1, page 8. USENIX Association, 2004.

[22] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A public DHT service and its uses. *ACM SIGCOMM Computer Communication Review*, 35(4):73–84, 2005.

[23] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *Computing Research Repository*, 2002.

[24] A. Singh and L. Liu. Agyaat: Providing Mutually Anonymous Services over Structured P2P Networks. In *Proc. of the 14th World Wide Web Conference*, pages 422–431. Citeseer, 2005.

[25] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, Feb. 2003.

[26] P. Wang, I. Osipkov, N. Hopper, and Y. Kim. Myrmic: Secure and robust DHT routing, 2006.

[27] Y. Zhu and Y. Hu. SurePath: An Approach to Resilient Anonymous Routing. *International Journal of Network Security*, 6(2):201–210, 2008.

[28] L. Zhuang, F. Zhou, B. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 301–314. USENIX Association, 2005.

APPENDIX

Claim A.1: Recieving a false mesh network routing table will not leak any more information than is leaked in normal protocol operation.

Proof: By protocol step 7b, the querier must recieve the range of the node being queried before sending any key-related information. By Assumption 1, this range is truthfully reported. If the querier recieves a list of adversarially chosen nodes, each node will be either adversarial or benign:

- If adversarial, and it does not contain the target key, then by protocol step 7e, the querier will request a dummy key uniformly at random from within that node's range. The adversary can revise the set of keys which contains k to not include those keys which fall within the range hosted by that node. However, it could do this via normal operation as well.
- If an adversarial, and it does contain the target key, then the adversary learns k. However, this would occur during normal protocol operation.
- If benign, then by Theorem A.1, the adversary cannot distinguish a masking query from a genuine query.

In any of the above cases, the protocol will terminate once the entire list of nodes has been queried. ■

Theorem A.1: If all components of step 7 in the Crypsis query protocol use a semantically secure encryption scheme, then the Crypsis protocol meets the security definition with respect to a global observer.

Proof: We prove the above claim via a reduction to encryption. Assume that the thesis is false, and a Crypsis adversary can succeed in distinguishing messages with a non-negligible probability. Formally, assume there exists an algorithm A_c such that:

$$|P(A_c((m.w = t) = t)) - P(A_c((m.w = \neg t) = t))| > \epsilon_c \tag{5}$$

Then we can define an algorithm A_e which uses A_c to succeed in distinguishing semantically secure messages. Consider the following game:

1) A_c chooses the set K and transcript length a, and outputs them to A_e.
2) A_e generates a transcript using the following procedure:
   a) Draw a key from k with uniform probability.
   b) Ask the encryption oracle to encrypt $(k, 0, get)$.
   c) Generate the tuple (seq, d, m). Set $m = E(k, 0, get)$.
   d) Add the tuple to T.
   e) Repeat steps 1-4 until $|T| = a - 1$.
   f) Draw two keys m_0, m_1 and submit both to the encryption oracle with the above concatenation.
   g) Insert the returned c^* into T at a uniformly random position.
3) Give T to A_c.
4) A_c outputs the tuple (t, w, b). If $w \neq c*$ or $t \notin \{m_0, m_1\}$, abort A_e.
5) Otherwise:
   - If $(t = m_0)$ guess $c^* = m_0$
   - If $(t = m_1)$ guess $c^* = m_1$

Note that A_c is useful only if the advantage it contributes is non-negligible. Since A_e effectively simulates A_c:

$$\begin{aligned}
&w = P(A_c(m.w = t) = t) \cap \overline{ABORT})\\
&x = P(A_e(Enc(t)) = t)\\
&w = x\\
&y = P(A_c(m.w = \bar{t}) = t) \cap \overline{ABORT})\\
&z = P(A_e(Enc(\bar{t})) = t)\\
&y = z
\end{aligned}$$

Then it follows that:

$$|w - y| = |x - z| \tag{6}$$

As described above, an abort occurs when A_c does not output both c^* and either m_0 or m_1. Since c^* is inserted into the transcript uniformly at random and $m_0, m1$ are drawn from K with probability $\frac{1}{|K|}$, respectively, the probability of aborting is independent of c^*, m_0, and m_1. Therefore $P(m = c^*) = 1/|T|$ and $P(t \in \{m_0, m_1\}) = 2/|K|$. So not aborting happens with the probability $P(\overline{ABORT}) = \frac{2}{|T||K|}$. Then we can rewrite $|w - y|$ as:

$$P(\overline{ABORT})|P(A_c(m.w = t) = t) - P(A_c(m.w = \bar{t}) = t)| \tag{7}$$

By definition of A_c's win condition, $|P(A_c(m.w = t) = t) - P(A_c(m.w = \bar{t}) = t)| > \epsilon_c$. Then substituting into the above relations, it follows that:

$$|x - z| > P(\overline{ABORT}) * \epsilon_c \tag{8}$$

Therefore the advantage gained by simulating A_c is non-negligible. ■

Theorem A.2: For n iterations of the security game, quantify the number of correct adversarial outputs of the form $(m.w = t)$ for a chosen target key t as X. Then X is subject to the following upper Chernoff bound:

$$P(X \geq (1 + \delta)\frac{n}{|M|} \leq e^{-\delta^2 \frac{n}{|M|*3}} \text{ and } 0 \leq \delta \leq 1 \tag{9}$$

Proof: Assume that n iterations of the game take place. For each iteration, define an indicator variable x_i, where $0 \leq i \leq n$. Set x_i as follows:

$$x_i = \begin{cases} 1 \text{ if A is correct} \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

$$\text{Let } X = \sum_0^a x_i \tag{11}$$

Then X is equal to the number of correct guesses made by the adversary. By Theorem 1.1, X is the sum of independent Bernoulli trials conducted with a coin biased by at most

$\frac{1}{|M|} + \epsilon$. This satisfies the requirement for establishment of a Chernoff bound. The quantity $\frac{n}{|M|}$ is, by Eqn 1, the expectation on X. ■

Claim A.2: An active adversary gains only limited information

First assume the adversarial nodes do not deviate from protocol. Then with respect to a particular key k, any adversarial node A will fall into one of two categories:

- $F(A) \neq F(k)$: The adversarial node is in a different mesh network, and does not participate in any secure protocol iterations for k (i.e protocol step 7). It learns only the dummy key being used to route to the mesh network hosting k.
- $F(A) = F(k)$: If A is the host of k, the adversary will learn k. If the hash function used to assign IDs to nodes and items is secure (i.e. acts like a random oracle) and uniformly maps pre-images to its domain, then P(A hosts k) = $\frac{1}{|N|}$, where N is the set of all nodes in the network. If A is not the host of k, then the adversary can revise the set M to exclude those keys which the node does host.

If adversarial nodes do deviate from protocol, then there are four places in to protocol where they may do so:

- Step 4: An adversarial node may re-route or drop messages on the overlay. These messages contain only F(k), however, which is not secret information.
- Step 6: This step is addressed in the section on range reporting.
- Step 7c: By Assumption 1, the adversary cannot report false information on this step.
- Step7f: An adversarial node may falsifiy the value of a key that it hosts. This may amount to denial of service, but does not compromise our security properties. Consider: the value is either meaningless (if responding to a dummy key request) or is the value associated with k, which is the case only if the adversary hosts k and trivially knows the popularity of k.