Securing Structured Overlays Against Identity Attacks

Krishna P. N. Puttaswamy, Ben Y. Zhao and Haitao Zheng
Computer Science Department, University of California at Santa Barbara
{*krishnap, ravenben, htzheng*}*@cs.ucsb.edu*

***Abstract*—Structured overlay networks can greatly simplify data storage and management for a variety of distributed applications. Despite their attractive features, these overlays remain vulnerable to the Identity attack, where malicious nodes assume control of application components by intercepting and hijacking *key-based routing* (KBR) requests. Attackers can assume arbitrary application roles such as storage node for a given file, or return falsified contents of an online shopper's shopping cart. In this paper, we define a generalized form of the Identity attack, and propose a light-weight detection and tracking system that protects applications by redirecting traffic away from attackers. We describe how this attack can be amplified by a Sybil or Eclipse attack, and analyze the costs of performing such an attack. Finally, we present measurements of a deployed overlay that show our techniques to be significantly more light-weight than prior techniques, and highly effective at detecting and avoiding both single node and colluding attacks under a variety of conditions.**

I. Introduction

As the demand for Internet and web-based services continues to grow, so does the scale of the computing infrastructure they are deployed on. Recent literature shows that structured peer-to-peer (P2P) overlays such as Chord [26], Pastry [21], and Tapestry [31] can greatly simplify data storage and management for a variety of large-scale distributed applications [17], [22], [33]. Finally, the usefulness of these infrastructures has been validated by recent studies of real world deployments of structured overlay applications, including Amazon's distributed key-value storage system Dynamo [9], and the popular BitTorrent client Azureus [12].

Despite the success of new application deployments, these application infrastructures remain vulnerable to several critical malicious attacks. One such attack, the Identity attack [13], would allow a malicious peer in the network to hijack application-level requests and assume the responsibility of any application component. For example, Amazon uses Dynamo servers to store information about user shopping carts, which is read and converted to html for user consumption. While Dynamo servers are accessed only by internal servers, a compromised host can potentially hijack read operations on a user's shopping cart, and return to requesting web servers a modified shopping cart.

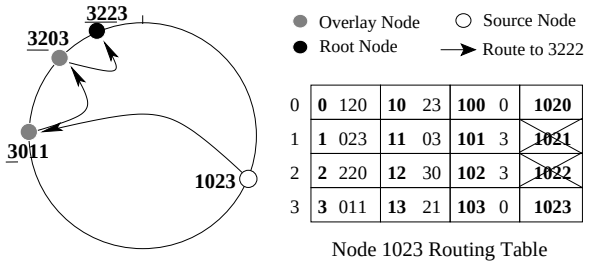
At their core, structured overlays scale to large networks because each node stores routing state that scales *sub-linearly* with the network size. Using this limited state, peers cooperatively map *keys* to physical network nodes using a multi-hop lookup mechanism called *key-based routing* (KBR) [7]. KBR maps a given key to a specific live node called its *root*. Overlay applications then use KBR to "choose" nodes for specific application components, *e.g.* the root node of a file's content hash is the file's storage server [6], and the root of a multicast session key becomes the root of the multicast tree [22]. By using KBR to choose application components, these applications expose themselves to hijacking attempts by malicious peers, similar in principle to BGP hijacking attacks [2].

In our preliminary work [13], we first described a limited version of the Identity attack and a basic framework for its detection in the Tapestry [31] and Pastry [21] protocols. In this work, we greatly expand our study in several key dimensions. First, we present a generalized form of the Identity attack defense, and demonstrate its applicability to 16 of the most popular structured overlay protocols. Second, we propose a novel tracking mechanism that allows the network to accumulate statistics on prior attacks through self-verifying *evidence*. By routing evidence to current and potential victims, we effectively protect application traffic by redirecting KBR requests away from attackers. Not only does this render attackers harmless, but it also does not unfairly penalize the few nodes who are falsely accused due to network instability. Third, we describe how Identity attacks can be amplified using the Eclipse attack [24], and perform analysis to quantify its required cost in terms of node identifiers requested. Finally, we measure the effectiveness of our solution on a deployed structured overlay network. In addition to experiments on the overlay, we implement a version of the Cooperative File System [6], and show how our techniques effectively reduce forged data blocks while requiring order of magnitude lower overheads than an alternative approach using redundant routing entries.

The paper is organized as follows: Section II summarizes background and related work. Section III summarizes the Identity attack, and describes our comprehensive framework for light-weight detection and countermeasures. Next, in Section IV, we examine Identity attacks enhanced by collusion via the Eclipse attack, and analyze the costs of collusion. We present detailed evaluation of our defense framework in Section V, and measurements of application-level impact via a study of CFS in Section VI.

II. Background and Related Work

We begin with a background discussion of structured overlays and their use of key-based routing. We then summarize



0	**0** 120	**10** 23	**100** 0	**1020**
1	**1** 023	**11** 03	**101** 3	~~1021~~
2	**2** 220	**12** 30	**102** 3	~~1022~~
3	**3** 011	**13** 21	**103** 0	**1023**

Node 1023 Routing Table

Fig. 1. *Prefix routing.* Top: node `1023` sends a message to key `3222` in a structured overlay using prefix routing. Bottom right: routing table for node `1023`. The node itself fills one entry in each column. Entries that no nodes match are crossed out. Names are represented in base 4.

work on security for structured overlays and related topics.

Structured Overlays and KBR. A structured overlay is an application-level network connecting any number of nodes, each representing an instance of an overlay participant. Nodes are assigned nodeIds uniformly at random from a large identifier space. To enhance overlay security, we assume that nodes register with a centralized certificate authority (CA) for a public/private key pair, and the CA binds the node's random nodeId with its public key using a public key certificate. Application-specific objects are assigned unique identifiers called keys from the same space.

The overlay dynamically maps each key to a unique live node, called its *root node*. While a key's root can change with network membership, at any given time in a consistent network, a single node is responsible for each key. The root is usually defined as the peer with nodeId closest to the key. Our detection mechanisms verify this overlay *invariant* to detect identity attacks. To deliver a message based on a key to its root node (*key-based routing* [7]), each node forwards the message using a locally maintained routing table of overlay links. P2P applications use this to deterministically choose peers to perform specific functions.

The large set of existing structured overlay protocols differ in the specifics of their routing algorithms. In order to support a network of size N, most protocols require per-node routing state that scales as $O(\log N)$ and provide worst case $O(\log N)$ overlay hops between any two nodes. Routing proceeds by forwarding the message incrementally closer in the namespace to the desired key. Figure 1 shows an example of overlay routing in Chimera [1], a structured overlay using prefix routing similar to Tapestry [31] and Pastry [21]. Finally, while each system defines a function that maps keys to nodes, the exact function may vary slightly. For example, keys can be mapped to the live node with the closest nodeId as in Pastry, or the closest nodeId clockwise from the key as in Chord [26].

Attacks on P2P Systems. Previous work describes two attacks on structured overlays, the Sybil attack [11] and the Eclipse attack [3], [24]. Attackers with significant resources perform a Sybil attack by generating arbitrarily large number of identities in the overlay network. In the Eclipse attack, attackers collude to increase their influence on a target by introducing each other as performance optimizing alternative entries into the victim's routing table. Both attacks focus on using multiple identities to gain influence and control in the overlay routing layer.

Some prior work limits the Eclipse attack by putting additional constraints such as low in-degree count or geographic proximity on how nodes choose their neighbors in the overlay [3], [18], [24]. While these limit attackers from attracting more than their share of normal traffic, they cannot prevent them from harming their portion of the overlay traffic. In contrast, our work seeks to identify and actively eliminate the influence of attackers on all traffic. Condie et al. proposed using periodic routing table resets (induced churn) to limit the impact of Eclipse attacks. Sit and Morris [25] also alluded to a variant of the identity attack in their initial study.

The structured overlay security work by Castro et al. [3] is highly relevant to this work. The authors propose a number of techniques to secure routing through the use of redundancy. A number of significant differences distinguish our work. First, mechanisms proposed in prior work require proper tuning of system parameters, and comes at a significant cost in network bandwidth overhead. More specifically, the constrained routing table approach doubles the amount of neighbor maintenance traffic, and significantly reduces the overlay's ability to perform locality-aware optimizations such as proximity neighbor selection [15]. In contrast, we focus primarily on *identifying* the active attackers through a lightweight detection and evasion system (less than 1500 lines of C, including our implementation of a simple Certificate Authority). Our mechanisms do not limit routing optimizations, and add minimal per-node state. We provide a detailed comparison of the two approaches in Section VI.

Reputation systems have been explored as application-level security mechanisms for peer-to-peer systems. They can be used to quantify reliability of resources [8], [29] or individual peers [19], and have been applied to peer-to-peer network communities [27]. Finally, the PeerReview project [16] proposed secure message logs to improve accountability in distributed systems.

III. Defending the Identity Attack

Structured peer-to-peer applications use Key-based routing (KBR) [7] as a way to assign application components such as traffic indirection points, storage servers or measurement sensors to live nodes in the network. For example, distributed storage systems using the Distributed Hash Table (DHT) interface [6], [17] choose the storage server for each file or block as the root node of the file's content hash key. Thus `Put` and `Get` operations are KBR requests that terminate at the storage node. Data streaming applications such as Scribe [22] hash session names to generate keys, and use KBR to choose the root of the multicast tree. While application peers can communicate directly via IP, they must first use KBR requests to locate each other.

An attacker can hijack and claim KBR messages as their own, by exploiting the fact that each nodes only sees a small subset of the overlay members. We call this an *identity attack*.

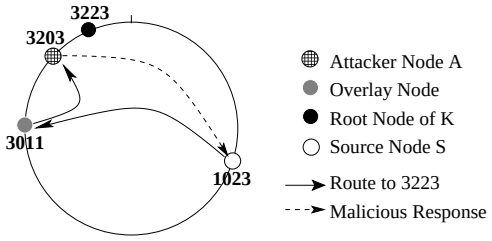


Fig. 2. *The Identity attack.* `1023` sends a message towards key `3222`. Before the message reaches the root (`3223`), an attacker intercepts it and responds as the root.

Any malicious peer on the path of a KBR message can respond to the source node and claim to be the request's destination. Undetected, the attacker claims control over a particular key and its associated application data. Intuitively, this abuse of limited per-node state is similar to prefix-hijacking attacks in BGP routing [2]. Multiple attackers can collude and perform stronger attacks such as isolating a node from the network, effectively performing a manual partitioning of the overlay.

In this section, we quickly summarize the basic identity attack described in our preliminary work [13], then define a generalized lightweight detection framework for structured overlays, and introduce techniques to mark attackers and redirect traffic away from their influence. We will delay discussion of advanced attacks involving multiple identities to Section IV.

A. The Single-node Identity Attack

In the single-peer attack [13], attacker A intercepts a key-based routing request from source peer S for key K, and responds to S that it is K's root node by virtue of being closer to K than any other peer in the network. Without local knowledge of peers closer to K than A, S then interacts with A as required by the application. Thus, A has effectively hijacked the overlay connection between S and K's root node at setup time. Figure 2 illustrates the attack.

By claiming to be K's root node, the attacker can intercept application requests and return data of its own choosing. The extent of control granted to the attacker varies across different applications. In distributed storage systems that use the Distributed Hash Table (DHT) interface [6], [17], the attacker can prevent blocks from being written and provide forged data to peers by hijacking *put* and *get* operations respectively. For directory services relying on decentralized object location and routing (DOLR) [20], an attacker can prevent the publication of resources, and redirect requests to malicious nodes hosting forged data. In multicast and anycast systems that relying on KBR for peer rendezvous [4], [22], clients can be redirected to join sessions run by malicious hosts.

Note that while applications can verify content integrity using mechanisms such as block checksums, the distribution of these mechanisms often rely on KBR and are themselves vulnerable to these attacks. Detecting the identity attack is difficult without an out-of-band peer-rendezvous and communication mechanism.

B. Detecting Identity Attacks

We now describe a generalized, light-weight detection mechanism suitable for most if not all of the structured overlays in current literature. We begin by stating several reasonable assumptions about the overlay network.

- During registration, a Certificate Authority (CA) assigns each node a unique nodeId, along with a public/private key pair [11]. The two are embedded in a public key certificate. This limits the impact of Sybil attacks, but is not a requirement for our mechanisms to work well.
- Local clocks at overlay nodes are loosely synchronized using the Network Time Protocol [10].
- Nodes digitally sign all of their responses to KBR messages with their public key.

Nodes detect identity attacks through the generation and timely dissemination of self-verifying "existence proofs." Overlay nodes periodically sign and distribute these proofs on behalf of well-defined regions of the namespace they reside in. For each region, a small number of randomly selected "proof managers" store these proofs and provide them on request.

Existence proofs are digitally signed certificates that prove at some time t, at least one live node existed whose nodeId resides inside a particular region of the overlay namespace. A node periodically constructs existence proofs for each namespace region it belongs to, and sends them to the set of *proof managers* for that region. An existence proof includes the signer's nodeId and a timestamp signed by the sender, and is valid for some time period after its issue. The use of the local timestamps prevents malicious nodes from replaying existence proofs and falsely accusing well-behaved nodes.

After receiving a signed reply to a KBR request for key K, a node S examines its local routing table to find the longest prefix column for which it has all entries filled. This threshold T is an approximate measure of the number of nodes in the network. A legitimate response should match the desired key K with at least T prefix digits. If not, S becomes suspicious, and attempts to verify the existence of nodes matching longer prefixes of key K. S computes addresses of proof managers by applying a secure one-way hash (SHA-1) to the region identifier and several small natural numbers (1, 2, 3). Proof managers are the "root nodes" of the resulting keys. S queries the relevant proof managers matching K's first T prefix digits for any existence proofs. If successful, the proof provides indisputable evidence of an attempted identity attack. Figure 3 shows node `1023` seeking proofs of nodes with prefix `322` by contacting three proof managers.

Optimizing Detection for Scalability and Robustness. We summarize several techniques that improve the scalability and robustness of the detection system. We refer the reader to [13] for additional details.

1) Limiting Prefix Groups– Certifying every possible prefix group in the network would result in a large number of existence proofs. However, given a rough estimate of the network size, each node needs to only certify a small constant number of namespace ranges. Each node can estimate the

Overlay Protocol(s)	How Regions are Named	Region Naming Example
Tapestry [31], Kademlia [IPTPS'02], Pastry [21], Bamboo [Usenix'04], LAND [SODA'04], Z-Ring [ICNP'05]	Numeric range (Prefix-based Routing)	[123*] (prefix length $L = 3$)
Chord [26], Symphony [Usenix'03], Viceroy [PODC'02], Accordion [NSDI'05]	Numeric range	[1235, 9] (center, size)
Koorde [IPTPS'03]	Numeric (de Bruijn Routing)	[1235, 9] (center, size)
SkipNet [USITS'03]	2 Ranges: numeric & alphabet	[abc*] and [123*]
CAN [SIGCOMM'01]	D-dimensional: Numeric	[1235, 9], [5675, 9] ...
Ulysses [ICNP'03]	Numeric range and Level	[1235, 9], L
Kelips [IPTPS'03], Tulip [IPTPS'05]	Gossip-based: no ranges	Name of affinity group

TABLE I

THE NAMESPACE REGION CERTIFICATION TECHNIQUE IS APPLICABLE TO MOST KNOWN STRUCTURED OVERLAYS. THIS SHOWS HOW DIFFERENT SIZED REGIONS ARE SPECIFIED IN 16 POPULAR PROTOCOLS. GOSSIP PROTOCOLS CAN ONLY CERTIFY REGIONS OF A FIXED SIZE.

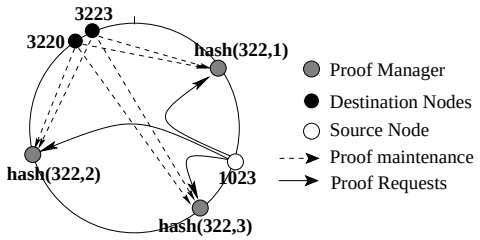


Fig. 3. *Identity Attack Detection.* Nodes `3220` and `3223` provide signed existence proofs for prefix `322` to three proof managers. Node `1023` seeks proof of a `322*` node.

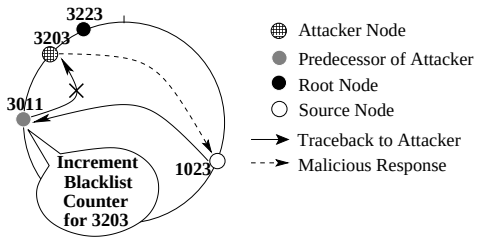


Fig. 4. *Tracking the attacker.* Node `1023` detects an attack by node `3203`, and sends an alert towards the attacker. `3203`'s predecessor `3011` receives the alert, increments `3203`'s local blacklist counter and switches its traffic to an alternate route.

number of active nodes n by examining the density of its local routing table, and need to certify at most $Log_2 n$ prefix groups or namespace ranges. Application of prior measurement and analysis results [32] show that certificate proofs of *three* prefix groups are sufficient to provide coverage of attacked nodes for different network sizes.

2) Replicating Proof Managers– Several factors can limit the success of proof managers. Node churn limits their availability; attackers between a client and proof manager can drop verification requests; and proof managers themselves can be malicious and deny any knowledge of requested existence proofs. We make detection more robust by using multiple proof managers for each namespace region. This "replication" increases the probability that one or more non-malicious managers will be online despite network churn. Replicating the proof managers also addresses the problem that our verification mechanism is dependent on overlay routing. Multiple independent proof managers increase the number of verification requests that avoid interception by malicious peers, providing more reliable routing without any complex techniques. We quantify the benefits of proof manager replication through detailed experiments in Section V.

3) Caching Existence Proofs– An overlay node can cache existence proofs it observes in the network. Since proofs are self-certifying, any node can observe and cache them as they are sent on their way to proof managers. Locally cached unexpired proofs can be referenced as an alternative to sending verification requests.

Defining Namespace Regions. Our defense against identity attacks relies on certifying that nodes exist in different regions of varying sizes. This mechanism generalizes to any structured overlay with a notion of continuous namespaces. Only the mechanism for specifying a region needs to be customized for each protocol. For example, prefix-routing protocols like Tapestry [31] and Pastry [21] define regions as all nodes whose nodeId share a matching prefix, *e.g. prefix*$\{123\}$. The length of the prefix defines the region size. In contrast, protocols like Chord [26] that use pure numerical closeness for routing can define the same region using a center identifier and a region size, *e.g.* [1230–1239] is *range*$\{1235,9\}$. To illustrate the generality of our mechanism, we show in Table I how it applies to 16 protocols described in literature. We describe how each protocol defines ranges in its namespace, and give an example to show how to define regions of varying sizes. Note that for protocols like Tulip and Kelips, namespace regions can only be divided into fixed affinity groups.

C. Tracking and Avoiding Attackers

Merely detecting an attack is insufficient. Nodes must reliably perform key-based routing despite the presence of malicious peers. In this section, we describe novel mechanisms that use self-certifying evidence of the attack to track down and mark attackers, allowing overlay peers to locate and avoid attacker nodes in favor of more reliable alternative routes.

Self-verifying Evidence of an Attack. For overlay nodes to take action after an attack by A on key K, nodes must observe unforgeable evidence of A's malicious behavior. This evidence

comes in the form of two components, the original reply to the KBR message signed by A, and an existence proof signed by a node who is a better root node for K. Since both the reply and proof contain signed timestamps, a third party can observe whether the KBR reply was sent while the existence proof was still valid. Assuming peers are loosely synchronized via the NTP protocol, peer timestamps should be synchronized within 200ms [10]. Since we expect existence proofs to be valid on the order of minutes (30 sec to 2 min), errors in time synchronization should not prevent validation of the evidence from an attack.

Tracking Attackers via Blacklists. After detecting an attack, the message source S can forward evidence of the attack to interested third parties in the network. If node A attacks a KBR message for key K, S prepares an *alert* message including K, A's signed reply, and the existence proof from a proof manager, and sends the alert as a special message to attacker A. Each node forwards it towards A, and checks to see if its next-hop is node A. If so, the node is a predecessor of A, verifies the evidence is valid, and adds A to a local *blacklist*. Each node on the blacklist has an associated counter that is incremented each time a new alert is presented showing that node performed an attack. Figure 4 shows this tracking mechanism in detail.

Blacklists are not foolproof. First, nodes in a dynamic network can observe a small number of false positives in attack detection due to network churn. Over time, nodes blacklist values slowly decay using an exponentially weighted moving average, thus preventing long-term accumulation of false positives. Second, multiple attackers can collude where one attacker forwards traffic to other attackers, and acts as a shield by dropping alert messages. We are actively investigating an effective approach to identify and avoid these collusion groups.

Evading Attackers via Malice-aware Routing. Once attackers have been identified with blacklists, nodes can actively avoid them when routing KBR requests. The blacklist counter for a routing entry acts as a simplistic reputation value that indicates a likelihood of malicious behavior. Routing policies can consider blacklist values in conjunction with a node's link quality and network latency when choosing between multiple routes. This proactive avoidance approach to routing increases application reliability without unfairly punishing nodes involved in false positives. Finally, malice-aware routing policies allow nodes to dissociate from attackers over time, thus reducing attackers' network in-degree and future attacks.

IV. Collusion-enhanced Identity Attacks

Peer-to-peer systems that provide zero-cost identities suffer from the Sybil attack [11], where a single user can obtain large numbers of virtual identities. These virtual nodes can collude to infiltrate the routing table of a single target and perform a collusion-enhanced Identity attack (or the Eclipse attack). In this section, we describe this attack in more detail, analyze its cost in virtual identities, and outline defense mechanisms.

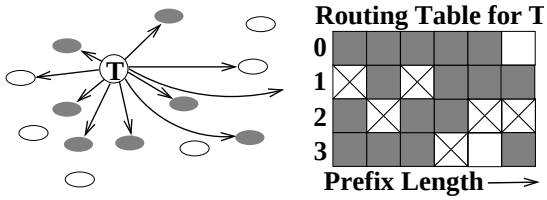


Fig. 5. *The Eclipse attack.* Multiple peers collude against target node T to infiltrate its routing table. Dark nodes denote colluders. Entries where T fills its own routing table are marked with a cross.

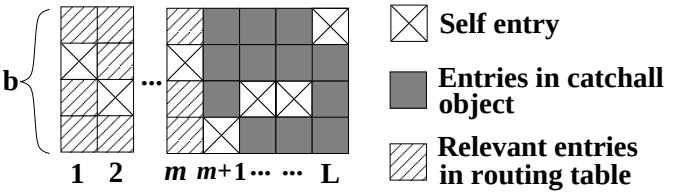


Fig. 6. *Mapping per-column-filling to the General Coupon Collector's problem.*

Eclipse Attacks and User Collusion. In the Eclipse attack [3], [5], [24], malicious peers exploit overlay optimization algorithms such as Proximity Neighbor Selection [15] to introduce colluding peers into a target's routing table. An example is shown in Figure 5. If successful, a group of peers control the majority of outgoing traffic from the target, and can then be used as a platform to launch powerful attacks. First, colluding peers can perform a colluding Identity attack by hijacking all KBR requests from a peer, effectively *isolating* it from the network, surrounding it with a self-consistent virtual network where all exchanges are controlled by a colluder. Second, colluding peers can drop all outgoing verification requests, making it extremely difficult to detect ongoing attacks.

Estimating the Cost of Eclipse Attacks. To perform a successful Eclipse attack, an attacker must have control over specific nodeIds that satisfy routing constraints in the target node's routing table. Because nodeIds are assigned at random, an attacker must obtain a large number of nodeIds in order to obtain the requisite nodeIds that infiltrate a particular target. Thus, we quantify the "cost" of launching an Eclipse attack as *the average number of nodeIds must an attacker request to attack a target* T. Since the network is likely to be sparse, a node's routing table will contain valid entries (other than the local node) only for the first few levels. Therefore, the cost depends on what levels of the routing table would the attacker have full control over.

For our analysis, the relevant parameters are b, the base of the nodeId, N, the size of the namespace, $L = Log_b N$, the total number of levels in the routing table, and $m(1 \leq m \leq L)$, the number of levels in the routing table that the attacker would have full control over.

Mathematically, we can reduce this problem into a variant of *The General Coupon Collector's Problem* [30]. In each level of the routing table, the local node fills one entry, leaving $b-1$ entries to be filled (recall from Figure 1). Each randomly obtained nodeId will have probability of b^{-l} to fill one of the $b-1$ entries at level l $(1 \leq l \leq L)$. Hence, the attacker

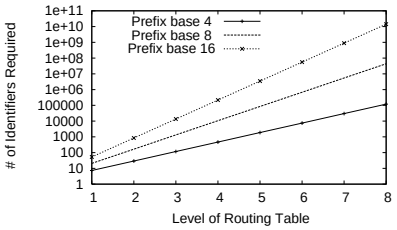


Fig. 7. Expected # of randomly obtained nodeIds required to fill L levels of the target's routing table.

needs to obtain nodeIds to fill $m(b-1)$ routing entries (m levels and $b-1$ entries per level). Finally, we introduce an artificial "catch-all" entry that represents all the remaining routing entries that the attacker doesn't need to have control over (shown in Figure 6). The probability of any nodeId falls into this entry is $1-\sum_{l=1}^{m}(b-1)b^{-l}$.

By mapping each of the $m(b-1)$ routing entries and the catch-all entry into an object, we form the corresponding General Coupon Collector's Problem as: *Let $K = m(b-1)+1$ objects to be picked repeatedly and randomly; with probability p_i that the i^{th} object is picked on a given try, and $\sum_{i=1}^{K} p_i = 1$; the expected number of tries T_E required after which all K objects have been picked at least one is* [28]:

$$T_E = \sum_{1\le i_1\le K}\frac{1}{p_{i_1}} - \sum_{1\le i_1<i_2\le K}\frac{1}{p_{i_1}+p_{i_2}} + \ldots + (-1)^{K-1}\frac{1}{p_1+p_2+\ldots+p_K}. \quad (1)$$

Since the probability of filling the catch-all entry is significantly higher than that of the $m(b-1)$ routing entries, T_E can accurately represent the cost of Eclipse attack, *i.e.* the number of nodeId randomly acquired in order to fill the first m levels of the routing table.

We apply our analytical result to prefix-routing networks using digit bases of b=4, 8, and 16. Figure 7 shows the cost of filling the first m levels of the routing tables. Note that the number of nodeIds required increases exponentially with each additional level. Using base 16, the cost of filling the first 2 levels of the routing table is only 100–1000 nodeIds, but jumps sharply to 100,000 at the 4^{th} level and 1000,000 at the 5^{th} level. The results clearly show that the cost of the attack increases exponentially with each level, and quickly becomes prohibitively expensive past a few levels. We now examine how to leverage this result to evade the Eclipse attack.

Evading the Eclipse Attack. Our goal is to limit the impact, and to allow successful detection of Eclipse attacks. We adopt two complementary approaches in Chimera. First, nodes periodically induce artificial "churn," resulting in periodic refresh of routing table state [5]. This limits the overall proportion of malicious peers in a node's routing table. Second, nodes use one-hop indirection for verification requests, forwarding them through one or more random members of its leafset peers. These leafset entries represent entries at the highest levels of the routing table. Therefore, they share long prefixes with the target and are difficult to compromise using randomly assigned nodeIds. In addition, leafset nodes are likely to be geographically distributed across the physical network, making them more resistant to Eclipse attacks exploiting physical proximity optimizations. They serve a similar function as *constrained routing entries* in secure Pastry [3]. We evaluate the effectiveness of these mechanisms in Section VI.

V. System Evaluation

We use detailed measurements of a deployed prototype to evaluate the effectiveness of our mechanisms for detecting and avoiding single-node and collusion-enhanced Identity attacks. We examine the effectiveness of attack detection under multiple attack environments and churn models, and also quantify the overhead of our mechanisms.

A. Measurement Methodology

We implemented our protection framework on Chimera, a lightweight structured peer-to-peer overlay implemented as a C library [1]. Since Chimera uses prefix routing similar to Tapestry [31] and Pastry [21], we implemented a version of our detection framework customized for prefix routing. We deployed a network of 1500 Chimera peers on a 32 machine server cluster connected via switched Gigabit Ethernet. NodeIds are 160 bits long, generated from SHA-1 hashes of public keys, and represented as 40 hexadecimal digits. Unless otherwise specified, each prefix group stores existence proofs at three proof managers, nodes sign and distribute proofs every 15 seconds, and each proof expires after 30 seconds.

We implemented a simple centralized Certification Authority (CA) that runs at a well-known address. Each new node entering the network obtains a public-private key pair and certificate from the CA. We assume that peers communicate with the CA through a secure channel, and the CA can verify each peer's identity.

Attack Models. We first evaluate our defenses against two different attack models:

- *Verification Denials: Type 1*. Malicious peers hijack all KBR messages including application messages (*e.g. put/get for a DHT*), but route verification protocol messages (*e.g. existence proofs, verification request/responses*) correctly. Malicious peers also deny access to any existence proofs it stores as a proof manager. We refer to this model as Attack Type 1.
- *Denials and Existence Proof Drops: Type 2.* Malicious peers hijack all KBR messages, deny existence proofs as proof managers, and collude by dropping any existence proofs en route to proof managers.

Each experiment includes results from at least 3 runs. For each experimental run, we choose a random subset of the network nodes to be attackers. All attackers exhibit the same behavior and remain malicious throughout the experiment. Malicious source peers still forward their KBR requests correctly. Malicious destination peers also behave correctly, since they do not need to hijack traffic destined for them. We use Chimera's built-in reliable transmission mechanisms to ensure

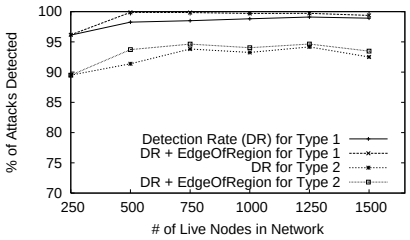


Fig. 8. Attack Detection with 20% malicious nodes, three proof managers.

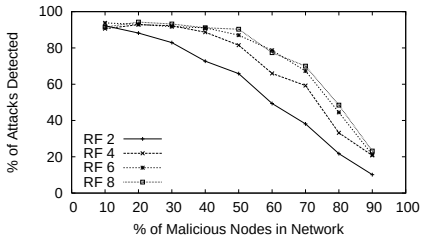


Fig. 9. Effect of Proof Manager Replication on Detection Robustness in a 1000 node network.

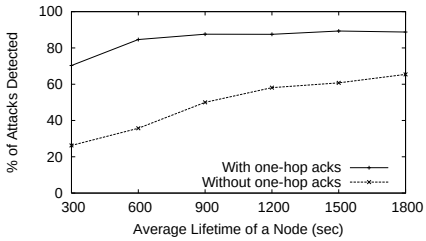


Fig. 10. Detection with and without one-hop acknowledgments, 1000 nodes, 20% malice.

no messages are lost and all KBR messages are responded to by its destination or a hijacker.

Note that the percentage of messages hijacked can be significantly higher than the percentage of malicious nodes in the network. A message is attacked if any of the nodes in its path are malicious. If p is the fraction of attackers in the network, the probability a message is attacked on a path with h hops is $1-(1-p)^h$. Our experiments confirm this expectation. Finally, in the absence of prior studies on level of malice in P2P networks, we assume a default 20% rate of malicious peers unless otherwise specified.

B. Measurement Results

In our tests, each peer in the network sends a predefined number of KBR messages to random destinations. Malicious peers attack messages they receive according to the experiment's attack model. After all messages are sent, replies received and processed by our framework, we use detailed per-message logs to compute the total number of attacks and detections.

1) Basic Detection and Analysis: Our first experiment measures the effectiveness of our detection mechanisms against Identity attack with Verification Denials (Type 1), assuming 20% of all network peers are malicious attackers. The results plotted in Figure 8 show that basic detection rate is very high (average of 95% or higher). Through careful analysis of the logs, we can attribute all detection failures to one of three scenarios. While we discuss our observations on Chimera, similar scenarios exist for other protocols such as Chord.

The few detection failures we observe can be attributed to three scenarios. First, because our certification mechanism has fixed namespace regions centered around a fixed point, keys near the edge of each region are more vulnerable. For example, key `5999`'s root node might be `6001`, but an attacker at `5900` matches the namespace region for $prefix\{5\}$ despite being further away in the namespace. We refer to this as the *Edge of Region Effect.* Second, since existence proofs each cover a namespace region, their precision is limited to the granularity of the smallest region. If the real root and the attacker reside in the same region, they cannot be distinguished from each other. We refer to this as the *Limits of Precision.* Finally, since existence proofs are self-certifying, a valid response from a single proof manager is sufficient to detect an attack. However, attackers can compromise all proof managers and complete deny access to the proofs (*Unavailable Proofs*). This accounts for $\approx 1\%$ of all attacks going undetected. This is exactly as we expected, given that we use 3 proof managers per group in a network with 20% attackers: $0.2^3 \approx 1\%$. Our experiments show that these scenarios are responsible for all undetected attacks under attack model 1. Figure 8 shows the undetected attacked accounted for by the Edge of Region effect.

2) Attacks and Proof Manager Replication: Figure 8 shows the detection rate for attack Type 2, which allows all malicious nodes to collude by dropping all existence proofs en route to their proof managers. There is a visible drop in detection rate compared to Type 1, since the chance of an existence proof routing through and being dropped by a malicious node is significant. Detection rates remain high (90%) because of the redundancy provided by multiple nodes providing existence proofs for each shared namespace region. Since this is the most powerful single node attack model, we use this as the default attack model for the remainder of our single-node attack measurements.

Proof Manager Replication. To understand the impact of replicating proof managers, we vary the proof manager "replication factor" (RF) and plot the detection rate with increasing malice in Figure 9. Results show that increasing the replication factor from 2 to 4 provides the greatest gain, with diminishing returns at RF 6 and 8. They also show that that higher replication (RF 6 or 8) can provide robust detection rates (70%) even when a large majority (70%) of the network is malicious.

3) Detection Under Network Churn: Realistic evaluation of overlay networks must include tests under network dynamics (network churn). We evaluate our implementation using a range of artificial churn models that follow exponential peer lifetime distributions. This exponential distribution produces even more extreme network churn compared to recent churn measurements on Skype [14] and Gnutella [23]. Our experiments maintain a near-constant network size by introducing new peers into the network as others leave. Our experiments using those churn models produced very high detection rates, which we omit here for brevity.

We observe that the main impact of churn is messages lost en route or being processed by a node leaving the network. By default, messages in Chimera are sent via UDP and acknowledged per overlay hop. If no ack is received, a message is retransmitted up to three times. Chimera also measures overlay links for loss, and performs route switching based

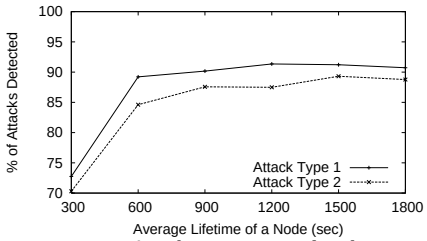


Fig. 11. Attack Detection under churn.

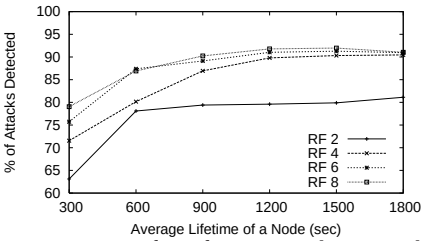


Fig. 12. Impact of proof manager replication under churn, Attack Type2.

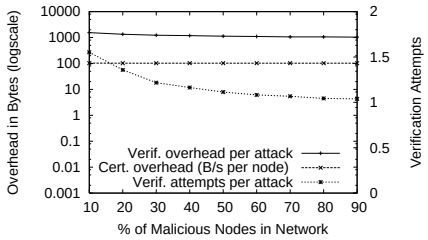


Fig. 13. *Bandwidth measurement of certification and verification messages.* Attack Type 1 on a 1000 node network.

Proof Expiration Time (s)	30	60	90	120
% of False Positives	6.0	11.9	12.8	14.6

TABLE II
% OF FALSE POSITIVES AND PROOF EXPIRATION TIMES

Cert. Overhead (bytes per second per node)	102.86
CRT Overhead (bytes per second per node)	196.11

TABLE III
COMPARISON OF CERTIFICATION OVERHEAD WITH CRT OVERHEAD, NETWORK HAS 1000 NODES.

on loss rates. Figure 10 shows the same experiment (1000 nodes, 20% malicious nodes, attack Type 2) with and without one-hop retransmissions. Clearly, local retransmissions made transmission of proofs and requests reliable, thus dramatically improving detection under churn, *e.g.* from 26% to 80% for networks with node life time distributions around 300 seconds.

Figure 11 shows that detection rates drop significantly for high churn rates (lifetime = 300 sec). With higher churn, new proof managers might not have received existence proofs and those with proofs might have left the network, making proofs unavailable and lowering detection rates. We also plot the impact of increasing number of proof managers under churn in Figure 12. The results are consistent with our non-churn experiment, and show less incremental improvement for more than 4 managers.

4) False Positives: Churn can produce temporary inconsistencies in peer's routing state. In some cases, peers might be unaware of better root nodes for a given KBR request, resulting in an observed Identity attack. Note that our tracking and avoidance mechanisms do not adversely impact falsely accused nodes. While very few instances are actually observed in our experiments, false positives for our system can occur under two rare scenarios:

Unexpired Existence Proofs: a node sends an existence proof to its proof managers just before leaving. Its proofs are valid for some time after it has left (expiration time of 30 seconds in our tests). Requests for keys controlled by the departed node can cause false positives. Reducing the proof expiration period reduces false positives but increases system overhead. Table II shows a clear correlation between our false positive rate (out of all detected attacks) and shorter existence proof expiration periods. *New Roots:* a node between the KBR key and its root enters the network as a KBR message reaches its root. Existence proofs from the new node arrive before any verification requests, leading to a false positive.

5) System Bandwidth Overhead: Figure 13 summarizes our detailed bandwidth measurement results from a 1000 Node network with attack Type 1[1]. Certification overhead per second per node remains constant as rate of malicious attackers increases. The low number of verification requests per actual attack means our system is issuing few spurious verifications. The decrease in verification attempts per attack can be attributed to the additional attacks actually decreasing the number of spurious verifications attempted.

We also compare the overhead of our approach to the previously proposed constrained routing tables solution [3]. We implemented constrained routing tables on Chimera and maintained entries in the routing table with the same heartbeat period as Chimera routing entries: 20 seconds. Measurements from this implementation for a 1000-node Chimera network are shown in in Table III. Note that while certification overhead of our proposal stays constant across different network sizes, we expect the constrained routing table maintenance overhead to increase for larger networks as the number of routing entries increases, showing that the routing table maintenance overhead of our proposal is lower than that of earlier proposal. We quantify the overhead of routing failure test and the redundant routing overhead in the later section.

6) Blacklists and Avoiding Malicious Nodes: We measure the effectiveness of alerts to track down attackers by their predecessors. We run a Chimera network of 1000 nodes, and randomly mark 20% of nodes as malicious attackers. Each node sends 50 KBR messages to randomly chosen key destinations; malicious nodes attack all messages; and peers send alerts for each detected attack.

In Figure 14, the X-axis sorts each predecessor-attacker combination by the number of attacks performed across the link, and the Y-axis plots the corresponding blacklist value. We introduce a random variance of 0.15 on Y-values to distinguish multiple points. As the results show, blacklist values are closely correlated to the actual number of attacks, meaning alert messages are locating the attacker's predecessor. Predecessors whose traffic the attacker hijacks will accrue high

[1]We performed bandwidth measurements for all attack types, and found that Type 1 incurs the highest overhead since fewer messages are dropped and more replies are made to verification requests.

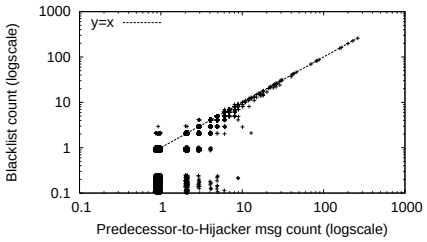


Fig. 14. *Effectiveness of blacklist counting.* Alert messages track down the attacker's predecessor and increment a blacklist count for the attacker. Attack Type 2, 1000 node network.

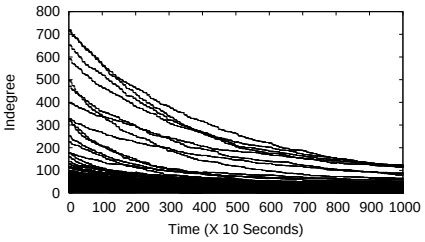


Fig. 15. *In-degree changes of the malicious nodes.* Attack Type 2, 1000 node network, KBR messages sent at a rate of 1 msg/sec.

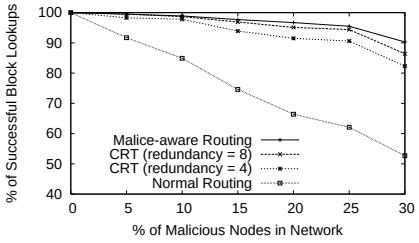


Fig. 16. *CFS block lookups comparing our approach against constrained routing tables.* Attack Type 2, 1000 node network.

blacklist values for the attacker.

To examine the impact of our countermeasures, we implemented a routing policy in Chimera to prioritize attacker-avoidance over other metrics such as reliability or link latency when performing route selection. By default, Chimera maintains three routes per entry in the routing table.

We take snapshots of the "in-degree" of each attacker at ten second intervals in an experiment that totals 10,000 seconds, where in-degree is the number of predecessors for which the node is the preferred route for some path. We plot each attacker's in-degree at each snapshot as a dot in Figure 15. As malicious nodes attack, they are detected and blacklisted, causing their in-degree counts to drop quickly. Malicious attackers with a high in-degree at the beginning of the experiment quickly drop as their many predecessors receive alerts of their attacks. The results confirm that the malice-aware routing policy is highly effective in marking and redirecting traffic away from attackers.

VI. Cooperative File System Measurements

Ultimately, our detection and avoidance framework should effectively protect applications from the Identity attack. To quantify the impact of our system on a real application, we implemented a version of the Cooperative File System (CFS) [6] on Chimera, which supports the `PUT`, `GET` block operations and block replication. To measure just the effect on lookup, we ensured that only `GET` messages are hijacked by malicious nodes. Upon detecting a `GET` hijack, nodes retry the operation. To contrast our approach against existing work, we implemented the secure constrained routing approach proposed by Castro et al. [3], and compare the two systems side-by-side on effectiveness and overheads.

For these experiments, we use an even stronger attack model than before (Type 3), where each malicious peer denies access to existence proofs and hijacks *all* messages routing through it, including verification request messages. This simulates the effect of a complete Eclipse attack, when attackers are trying to effectively partition the target from the network.

A. Effectiveness of Malice-aware Routing

To improve the lookup success of CFS under attack, we implemented the malice-aware routing policy described earlier in CFS. Nodes favor routes with lower blacklist counters, and use backup routes or a leafset entry to evade attackers unless the blacklisted entry is itself the root of the blockId.

We run a 1000 node Chimera network with Type 2 malice. Each node performs a `PUT` and `GET` on 25 unique blocks, resulting in read and write operations on a total of 25,000 blocks. Figure 16 shows that malice-aware routing mechanism significantly improves the lookup success of CFS compared to CFS with normal routing. Results also show that malice-aware routing performs better than constrained routing with redundancy factors of four and eight. These values represent the number of parallel outgoing requests following an attack, and are set high to maximize the effectiveness of constrained routing.

The next step is to understand the overheads incurred by the defense mechanisms. We measured the overhead of our proposal as the number of all request, proof and verification messages exchanged. To measure the overhead of constrained routing, we measured the number of routing failure tests triggered, the number of test successes, and the number of replica messages sent in the block lookup tests. Combining these values and certificate and hash sizes and equations from [3], we compute the total overhead. We optimized our implementation of the constrained routing approach to use the best parameters that minimize errors in the routing failure test. Where appropriate, we used identical parameter values from our implementation (*e.g.* leafset size = 8, IdSize=40 bytes). All other parameters used in the implementation and calculations are taken directly from [3]. Results in Figure 17 show that our proposal incurs more than an order of magnitude less overhead compared to constrained routing.

B. Effectiveness of Defense against Collusion-based Attacks.

We now evaluate our solution for evading collusion-enhanced eclipse attacks from the perspective of a single target node. As discussed in Section IV, we use a combination of periodic induced churn [5] and one-hop redirection through other nodes.

We model the Eclipse attack by initializing the network with 20% randomly chosen malicious nodes. We assume a node's routing table starts with the same rate, and introduce more malicious peers into the routing table at a constant rate, updated every 20 seconds. The target resets its routing table every 100 seconds, reducing the percentage of attackers back

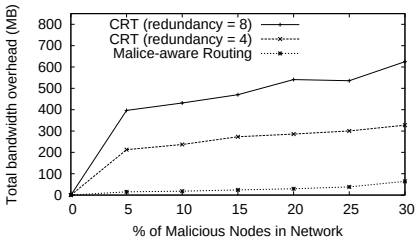


Fig. 17. *Comparing overhead of CFS block lookups in our system and constrained routing.* Attack Type 2, 1000 node network.

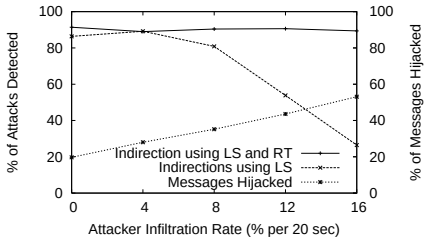


Fig. 18. *Effectiveness of Eclipse Attack defenses with induced churn and one-hop indirection.* Attack Type 3, 1000 node network.

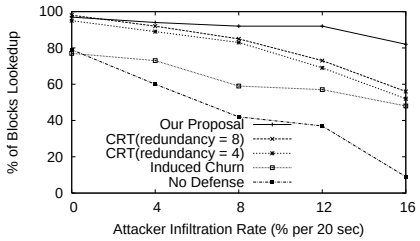


Fig. 19. *Effectiveness of various Eclipse attack defenses in CFS.* Attack Type 3, 1000 node network.

down to the same level as the rest of the network (20%). All verification requests are routed using one-hop indirection through four nodes chosen from our leafset or routing entries. We run the experiment for 1200 seconds, and measure the portion of messages attacked, and the portion of all attacks detected. These results are plotted against infiltration rate (per 20s) in Figure 18.

We plot two separate results in Figure 18. First, we look at the scenario where the attackers have sufficient resources to request enough nodeIds to attack all routing levels of the target node. Therefore, we introduce malicious peers uniformly at random across all routing levels (and leafset) of the target node. We use indirection across the leafset (LS) nodes, and plot the results as "indirection using LS." While indirection results in high detection at low infiltration rates, as infiltration rates increase, leafset nodes are also affected, and indirection is less successful. In the second scenario, we assume the attacker has limited resources, and only has enough nodeIds to attack the lower half of the routing table. In this case, we can use indirection across both leafset nodes and higher level routing entries. The result is detection rates similar to those of single node attacks. Note that we are effectively using leafset entries and higher level routing entries as "hard-to-compromise" nodes, similar to constrained routing tables.

The second scenario described above is more practical and follows directly from our analytical results. Our results show that our proposal works well in this practical case, and is generally able to detect around 90% of all attacks in the stronger attacker model (Type 3).

We implemented various proposals from Castro et al. and Condie et al. to evade collusion-based Eclipse attacks in CFS. We ran the CFS block lookup experiments under the strongest attack model (type 3), and plotted the results in Figure 19. Our proposal performs the best, and works well even under extremely high rates of node compromise. More than 80% of lookups using our proposal succeed even when all lower level neighbor entries are malicious, while alternative solutions provide lookup success rates around 50%.

VII. Conclusion

In summary, we have proposed a general defense for the Identity attack on structured overlays. Using existence proofs, blacklists, and malice-aware routing, our system effectively detects, marks and redirects traffic away from attackers. We showed that our techniques are easily applied to and highly effective on real applications such as CFS. Measurements on CFS show that they perform at least as well as other proposed approaches in detection and recovery, but require an order or magnitude lower costs in bandwidth overhead. Finally, our analysis shows that performing a basic Eclipse attack and corrupting the lower-levels of a victim's routing table is practical, but can be successfully defended using our mechanisms. The vulnerabilities of KBR we described here are common to all structured overlay protocols. As more critical applications are deployed on structured overlays, application designers must be aware of these attacks, and integrate defense mechanisms into their protocols and applications.

References

[1] Alebouyeh, R., Allen, M. S., Puttaswamy, K. P. N., and Zhao, B. Y. Chimera software distribution. http://current.cs.ucsb.edu/projects/chimera.

[2] Ballani, H., Francis, P., and Zhang, X. A study of prex hijacking and interception in the internet. In *Proc. of SIGCOMM* (Kyoto, Japan, Sept. 2007).

[3] Castro, M., et al. Security for structured peer-to-peer overlay networks. In *Proc. of OSDI* (December 2002).

[4] Castro, M., et al. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proc. of SOSP* (October 2003).

[5] Condie, T., et al. Induced churn as shelter from routing-table poisoning. In *Proc of NDSS* (February 2006).

[6] Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. Wide-area cooperative storage with CFS. In *Proc. of SOSP* (October 2001).

[7] Dabek, F., Zhao, B., Druschel, P., Kubiatowicz, J., and Stoica, I. Towards a common API for structured P2P overlays. In *Proc. of IPTPS* (February 2003).

[8] Damiani, E., et al. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proc. of CCS* (November 2002).

[9] DeCandia, G., et al. Dynamo: Amazon's highly available key-value store. In *Proc. of SOSP* (Stevenson, WA, Oct. 2007).

[10] Deeths, D., and Brunette, G. Using NTP to control and synchronize system clocks. Tech. Rep. 816-1475-10, Sun Microsystems Inc., Palo Alto, CA, July 2001.

[11] Douceur, J. R. The Sybil attack. In *Proc. of IPTPS* (Cambridge, MA, March 2002).

[12] Falkner, J., Piatek, M., John, J. P., Krishnamurthy, A., and Anderson, T. Proling a million user dht. In *Proc. of Internet Measurement Conference* (San Diego, CA, Oct. 2007).

[13] Ganesh, L., and Zhao, B. Y. Identity theft protection in structured overlays. In *Proc. of (NPSec)* (Boston, MA, June 2005).

[14] Guha, S., Daswani, N., and Jain, R. An experimental study of the skype peer-to-peer voip system. In *Proc. of IPTPS* (2006).

[15] Gummadi, K., et al. The impact of DHT routing geometry on resilience and proximity. In *Proc. of SIGCOMM* (Sep 2003).

[16] Haeberlen, A., Kouznetsov, P., and Druschel, P. Peerreview: Practical accountability for distributed systems. In *Proc. of SOSP* (Stevenson, WA, Oct 2007).

[17] HAEBERLEN, A., MISLOVE, A., AND DRUSCHEL, P. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *Proc. of NSDI* (Boston, MA, May 2005).

[18] HILDRUM, K., AND KUBIATOWICZ, J. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. of DISC* (October 2003).

[19] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of WWW* (Budapest, Hungary, May 2003).

[20] RHEA, S., ET AL. Pond: The OceanStore prototype. In *Proc. of FAST* (April 2003).

[21] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware* (November 2001).

[22] ROWSTRON, A., ET AL. SCRIBE: The design of a large-scale event notification infrastructure. In *Proc. of NGC* (Nov. 2001).

[23] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (January 2002).

[24] SINGH, A., NGAN, T.-W., DRUSCHEL, P., AND WALLACH, D. Eclipse attacks on overlay networks: Threats and defenses. In *Proc. of INFOCOM* (Barcelona, Spain, April 2006).

[25] SIT, E., AND MORRIS, R. Security considerations for peer-to-peer distributed hash tables. In *Proc. of IPTPS* (March 2002).

[26] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM* (2001).

[27] SWAMYNATHAN, G., ZHAO, B. Y., AND ALMEROTH, K. C. Exploring the feasibility of proactive reputations. In *Proc. of IPTPS* (Santa Barbara, CA, February 2006).

[28] VON SCHELLING, H. Coupon collecting for unequal probabilities. *American Mathematics Monthly 61* (1954), 306–311.

[29] WALSH, K., AND SIRER, E. G. Evaluation of a deployed, distributed object reputation system for peer-to-peer filesharing. In *Proc. of NSDI* (San Jose, CA, May 2006).

[30] WEISSTEIN, E. W. Coupon collector's problem. http://mathworld.wolfram.com.

[31] ZHAO, B. Y., HUANG, L., STRIBLING, J., RHEA, S. C., JOSEPH, A. D., AND KUBIATOWICZ, J. Tapestry: A global-scale overlay for rapid service deployment. *IEEE JSAC 22*, 1 (January 2004).

[32] ZHAO, B. Y., KUBIATOWICZ, J. D., AND JOSEPH, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. CSD-01-1141, U. C. Berkeley, 2001.

[33] ZHUANG, L., ZHOU, F., ZHAO, B. Y., AND ROWSTRON, A. Cashmere: Resilient anonymous routing. In *Proc. of NSDI* (Boston, MA, May 2005), ACM/USENIX.