Unblocking the Internet: Social networks foil censors
NYU Technical Report TR2008-918

Yair Sovran, Jinyang Li, Lakshminarayanan Subramanian
Computer Science Department, New York University
http://www.kspro.org

Abstract

Many countries and administrative domains exploit control over their communication infrastructure to censor online content. This paper presents the design, implementation and evaluation of *Kaleidoscope*, a peer-to-peer system of relays that enables users within a censored domain to access blocked content. The main challenge facing Kaleidoscope is to resist the censor's efforts to block the circumvention system itself. Kaleidoscope achieves blocking-resilience using *restricted service discovery* that allows each user to discover a small set of unblocked relays while only exposing a small fraction of relays to the censor. To restrict service discovery, Kaleidoscope leverages a trust network where links reflects real-world social relationships among users and uses a limited advertisement protocol based on random routes to disseminate relay addresses along the trust network; the number of nodes reached by a relay advertisement should ideally be inversely proportional to the maximum fraction of infiltration and is independent of the network size. To increase service availability in large networks with few exit relay nodes, Kaleidoscope forwards the actual data traffic across multiple relay hops without risking exposure of exit relays.

Using detailed analysis and simulations, we show that Kaleidoscope provides $> 90\%$ service availability even under substantial infiltration (close to 0.5% of edges) and when only 30% of the relay nodes are online. We have implemented and deployed our system on a small scale serving over 100,000 requests to 40 censored users (relatively small user base to realize Kaleidoscope's anti-blocking guarantees) spread across different countries and administrative domains over a 6-month period.

1 Introduction

Internet censorship is widely prevalent in many parts of the world where governments and administrative domains censor the network access of users inside the domain [4]. While censorship of clearly objectionable content such as child pornography is acceptable and typically accompanied by real-world punishment for viewing such content, often a large volume of harmless content is censored due to poor traffic blocking policies (Figure 1). In many cases, even popular sites such as YouTube [35], Wikipedia [5] and Google [2] have been blocked in some regions.

The fundamental problem with a traffic-blocking based censorship system is that it is almost impossible to configure precise router policies to filter only objectionable content. For instance, Thailand and Pakistan had blocked the entire YouTube site just to prevent access to one offensive video clip [35]. The high collateral damage caused by imprecise and coarse-grained blocking policies motivates the need for a censorship circumvention system for accessing non-objectionable content (lower-right quadrant in Figure 1).

This paper presents the design, implementation, evaluation and deployment experiences of Kaleidoscope, a *practical and decentralized circumvention system* that enhances availability to blocked content while making it difficult for the censor to defeat the system. Existing censors actively try to block any popular system with a centralized component that circumvents their "firewalls", as witnessed by the blocking of SafeWeb [7] and Anonymizer [1]. Our design is completely decentralized to maximize availability but not privacy. Users concerned with privacy need to combine our solution with an existing privacy preserving anonymity system [19]. Kaleidoscope is also not designed for the case where

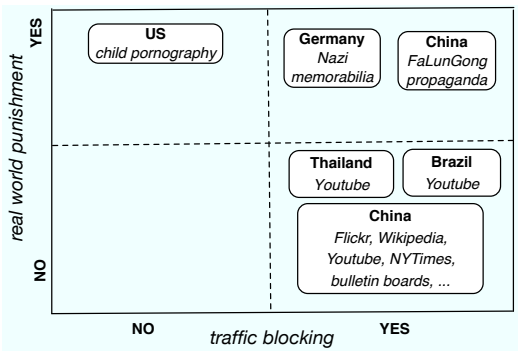

Figure 1: A classification of censored content. Accessing illegal materials results in real world punishment. For a large class of blocked content in the lower right quadrant, there is little risk of real world punishment for viewing.

the censor may persecute or harass users for viewing blocked content or using (or participating in) a circumvention system.

Kaleidoscope is a peer-to-peer circumvention system that uses a network of relay nodes (both inside and outside the censored domain) to help users inside the censored domain forward traffic outside the censored domain to access otherwise blocked websites. A subset of relays outside the censored domain serves as *exit points* (i.e. web proxies) to directly forward unencrypted traffic to websites. All traffic between nodes in Kaleidoscope is encrypted to thwart deep packet inspection-based traffic blocking. In addition, Kaleidoscope assumes that the number of relays outside the censored domain is large enough to avoid detection of relays using volume-based traffic analysis. However, we do not consider timing-based traffic analysis in the current design.

The design of Kaleidoscope reconciles the tension between (1) helping users learn about relays and forward their traffic to exit points and (2) preventing censors from blocking the relay system (by infiltrating the system to learn relay nodes). To achieve this objective, Kaleidoscope performs *restricted service discovery* [18, 22] that enables each user to discover a small number of relays; this restricts the censor's ability to discover relays.

Kaleidoscope achieves restricted service discovery using three main ideas. First, it disseminates relay information using a trust network whose links reflect real world social relationships among users. The use of a trust network prevents a censor from joining the network at arbitrarily many vantage points since doing so requires many trust links with honest users which are costly to obtain and maintain. Second, Kaleidoscope uses a limited advertisement protocol based on short random routes over the trust network [40]. This protocol disseminates relays' addresses in a way that restricts the number of users to which each relay node is exposed. We show that to handle a maximum fraction of infiltration, f, the number of nodes reached by a relay advertisement should be at most $2/f$, which is independent of the network size. Third, to achieve good service availability in large networks ($\gg 2/f$ nodes) with few exit points, Kaleidoscope uses multi-hop relay paths to forward traffic to exit points. Multi-hop forwarding in Kaleidoscope dramatically increases the service availability without risking exposure of the exit points to the censor. Kaleidoscope's multihop traffic forwarding differs from the forwarding mechanism in existing privacy-preserving anonymity systems such as Tor [19] because Kaleidoscope limits the exposure of relays whilst Tor obfuscates the origin of a traffic flow.

We demonstrate the effectiveness of Kaleidoscope using a real-world deployment as well as detailed analysis and evaluations using online social network traces. Our evaluations using online social network traces show that Kaleidoscope provides $> 90\%$ service availability even under substantial infiltration (close to 0.5% of edges) and when only 30% of the relay nodes are online. Our implementation bootstraps itself in a decentralized fashion and leaves no obvious network fingerprints. Our deployment has served over 100,000 web requests to a small community of 40 users from different countries with known censorship practices over a six-month period. Although our deployed network is too small to provide real anti-blocking guarantees, it has been used quite extensively by our users to access blocked content.

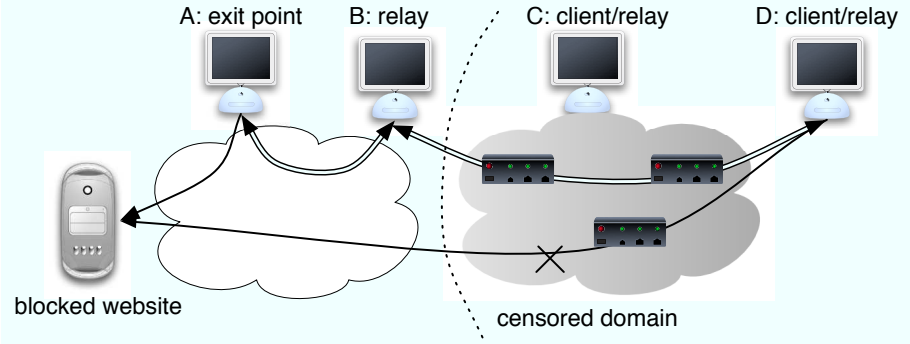

Figure 2: Routers within the censored domain drop user D's packets to the blocked website. All users can operate their nodes as relays and users outside the censored domain can operate exit points to forward unencrypted web traffic to blocked websites. The double lines correspond to encrypted communication channels while the single lines refer to unencrypted web traffic.

2 Problem Statement

This section discusses the general problem setting of circumventing traffic blocking censors (Section 2.1), our threat model and assumptions (Section 2.2) and the design guidelines for Kaleidoscope (Section 2.3).

2.1 Problem setting

We consider a censored domain where the censor controls all routers within that domain, as illustrated in Figure 2. This censor deploys filtering policies on those routers to prevent users within the censored domain from accessing selected content outside the censored domain. In the example of Figure 2, routers in the censored domain are configured to drop all packets whose destination IP address belongs to the blocked website. In addition to dropping packets based on IP addresses, routers can sabotage communication based on deep packet inspection. In particular, routers inside the censored domain reset TCP connections if unwanted keywords appear in the packet payload [16, 17] and also pollute DNS replies corresponding to unwanted domains.

Our proposed circumvention system, Kaleidoscope, aims to relay otherwise blocked web traffic outside the censored domain. As we are defending against a realistic censor that does not persecute individual users, a user inside the censored domain can install our software to forward her traffic via encrypted channels to other relay nodes. All users (both within and outside the censored domain) operate their nodes as *relay nodes* for others. A subset of users outside the censored domain also operates their nodes as *exit points* (i.e. web proxies) to forward the actual unencrypted web traffic to blocked websites; in general, not every relay node outside the censored domain may be willing to act as an exit point [19]. Relaying encrypted traffic across multiple Kaleidoscope nodes is beneficial as it allows an exit point to serve a larger population of users inside the censored domain without directly revealing its network address to them. For example in Figure 2, exit point A can indirectly serve user D via B without revealing its network address to D. Since forwarded traffic among Kaleidoscope users is encrypted, routers cannot perform keyword-based dropping. Furthermore, exit points perform DNS queries so routers inside the censored domain cannot disrupt DNS.

2.2 Threat model and assumptions

Our threat model is motivated by practical assumptions and threats. We assume that the censor will attempt to block the circumvention system but not target individual users. We must realize that since the censor controls the underlying communication infrastructure, any attempt to elude the most determined censor's attempts to block a circumvention system is futile: a censor could always deploy the "nuclear option" of pulling the plug on all external traffic (e.g. the governing junta in Burma) or drop all encrypted traffic to prevent evasions of content filters or white-list the set of allowed outside nodes.

We consider three types of attacks by the censor:

First, the censor can easily disable any centralized component in a circumvention system by blocking either the IP address or DNS name of the centralized component. As many existing circumvention

systems use a centralized web site to inform users of web proxies located outside the censored domain, they are vulnerable to this attack. For example, Anonymizer [1] and SafeWeb [7] were disabled this way in China.

Second, the censor can pose as a legitimate user in order to discover as many relays located outside the censored domain as possible and to block them. It is easy to launch such attacks against systems that publish the identities of all nodes. For example, the Tor anonymity network allows any user to discover all relay servers [19]. As a result, the Chinese censor has recently managed to block 80% of all Tor servers [11].

Third, the censor could also pose as a legitimate exit point or web site to attract traffic from many unsuspecting relays or users in order to track them. Node tracking in peer-to-peer systems already exists in the wild. A recent study of the Gnutella network has shown that a large number of peers are tracked by a few nodes that possibly collude with the RIAA [10]. Unlike other attacks, it is not clear what damage a censor can inflict upon the circumvention system as a result of user tracking. The censor could theoretically cut off the entire Internet connection of exposed users.

Our system is not designed to defend against sophisticated traffic analysis or timing attacks. We explicitly assume that simple traffic analysis based on traffic volume cannot learn of specific trust links or relay nodes within Kaleidoscope. This assumption typically holds if the cut-size of trust links that cross the censored domain is high and the volume of encrypted traffic across any single trust link is relatively small and comparable to the volume of encrypted traffic across many other communicating host-pairs. Although we cannot design a perfect system that remains undetected by sophisticated traffic and timing analysis, forcing a censor to implement more expensive countermeasures is still useful. If the censor must deploy strategies that affect legitimate traffic (blocking encrypted traffic also affects online commerce, for instance) or to undertake increasingly complex analysis schemes, the censor is likely to judge that the gain to be had in blocking Kaleidoscope does not justify the required cost (a cost that our system increases).

2.3 Design goals

The central challenge of a relay-based circumvention system is to help users discover relays and exit points while avoiding exposing them to the censor, even when the censor is able to pose as a legitimate user or exit point. Based on practical attacks on existing systems, we derive the following essential design requirements for blocking-resilient circumvention system:

1. *Decentralization*. A fully decentralized system foils any attempt to bring down the system by blocking its centralized component.
2. *Limited relay exposure*. The system must partition knowledge of all relays (including exit points) among users: if any user can learn the identity of every relay, then so too can the censor, who will block all of the relays. Since a decentralized system lacks strong user identity, the system must also defend against Sybil attacks [20], where a censor creates arbitrarily many user identities to try to discover as many different relays as possible.
3. *Limited user exposure*. The system must restrict each relay or exit point to serving (hence, learning) a small number of relays or users, otherwise, the censor can pose as a legitimate exit point to serve and track many users. Unlike the first two requirements, limiting user exposure is not strictly necessary for the health of the system, but rather a precautionary defense to prevent the censor from being able to track too many users.

A circumvention system meeting the above three requirements offers *restricted service discovery* for users. When under attack, such a system offers degraded service, i.e. not all users have access to some working relays outside the censored domain. We do not expect the system to guarantee service to all users: the censor will inevitably join the system and block some relays, disrupting service for some legitimate users. These disrupted users, unfortunately, must not be allowed to learn the identity of new relays, since if they could always learn the addresses of new relays, then so could the censor. Repeating this process would allow the censor to block all relays in the system.

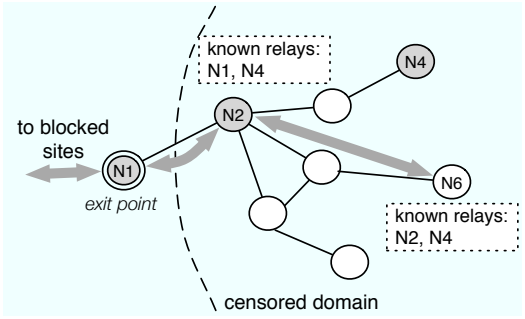

Figure 3: Kaleidoscope provides exit service using relay advertisement and multi-hop traffic forwarding. Gray circles denote relays and N1 is an exit point. N6 learns of relays N2, N4 and N2 learns of N1,N4 as a result of relay advertisement. The thick grey lines show the multi-hop path taken by N6's data traffic.

3 Basic Approach

Figure 2 describes the basic setting of Kaleidoscope that comprises of three entities: end-users, relay nodes and exit points. At the heart of its design, Kaleidoscope relies on a trust network where links reflect real world social relationships between participating users. It is also important to have a sizeable user population spread both within and across the censored domain; if this number is small, the censor can easily block all communication to these nodes upon detecting them. All nodes participating in the Kaleidoscope trust network by default act as *relay nodes* and a subset of relay nodes outside the censored domain volunteers to act as *exit points* (like proxies). Relay nodes use a limited advertisement protocol to advertise their presence and discover other relay nodes. Once relay nodes and exit points discover each other, they directly communicate with each other to set up encrypted channels for forwarding the actual data traffic. Relay nodes forward traffic between end-users and exit points and these routing paths in the peer-to-peer relay network need not involve the original trust links. Since each relay node may discover many other relay nodes apart from its neighbors, Kaleidoscope can provide high service availability even when many relay nodes are offline.

Kaleidoscope uses three key ideas to address the problem of *restricted service discovery*. First, Kaleidoscope disseminates the identities of relays and exit points along links in the trust network. If there are few attack trust links between honest users and the censor, the censor has limited ability to launch Sybil attacks and learn of many relays. This requires users to establish trust links with care. There are strong disincentives to choosing links carelessly: subverted trust links cause a node to lose its known relays and exit points. The censor could infiltrate the trust network by corrupting real users; however, we assume that the number of users colluding with the censor is small. Social links also provide the important benefit of allowing users to utilize offline communications to bootstrap the system in a completely decentralized fashion. In contrast, existing peer-to-peer networks require a fixed set of bootstrap servers which can be easily blocked.

Second, Kaleidoscope uses a limited advertisement protocol to restrict the scope of information dissemination about relays and exit points within the trust network. Contrast this approach with existing darknets [8], which reveal all nodes' identities in a trust network to each other. Consequently, any successful infiltration by the censor will expose all relays. On the other hand, each relay or exit point in Kaleidoscope advertises its address to a small number of other nodes along a series of trust links by performing a few short random walks beyond its immediate neighborhood (as illustrated in Figure 3). Hence, any node participating in the trust network learns only a small number of the relay nodes in the network.

Third, relay nodes play an important role in enhancing service availability in Kaleidoscope. In practice, due to limited the advertisement scope in Kaleidoscope, not every node will be aware of an exit point. To overcome this barrier and route traffic towards some exit point, relay nodes propagate information about the availability of exit service without directly revealing the identity of exit points. Multi-hop traffic forwarding allows each exit point to serve more users via a few intermediary relays without directly revealing its address to them(as shown in thick lines in Figure 3).

4 Kaleidoscope's Design

In this section, we elaborate upon three important design aspects of Kaleidoscope: how it leverages the trust network to disseminate relays' addresses (Section 4.1), how a node finds a multi-hop path to an exit point (Section 4.2), and how to set various parameters in Kaleidoscope (Section 4.3).

4.1 Limited relay advertisement

In Kaleidoscope, all users operate as relays by default and a fraction of nodes outside the censored domain act as exit points. To enable clients inside the censored domain to find short forwarding paths to an exit point, each relay and exit point should advertise its address to r nodes over the trust network, where r is larger than a node's immediate trust neighborhood.

Flooding advertisements is undesirable as the number of recipients can far exceed the target reach (r) with increasing number of hops. An alternate approach is to advertise a relay node's address along a few short random walks. A relay with degree d can perform d random walks of length $w = r/d$ via each neighbor. Any node receiving the advertisement learns of the relay's address before forwarding it on. To ensure that periodic advertisements reach the same set of r nodes, random walks should be made repeatable: each time a node forwards an advertisement it remembers the next-hop node and uses the same next-hop node for all future messages from the same advertiser.

Advertising along random walks has one short-coming: the censor could create many Sybil identities "behind" a single attack edge and have each identity advertise itself to r different nodes as a decoy exit point. As such, the censor could potentially learn of a huge number of clients exponential with the random walk length, violating design requirement 3. To defend against this attack, we use *random routing* [39, 40], where the basic idea is for every participating node to forward a relay advertisement message along a pre-determined outgoing link based on the incoming link of the message instead of the message originator's identity (as in repeatable random walks). Random routing is used in SybilLimit [39] to restrict the number of Sybil identities that an adversary can advertise along an attack edge. Therefore, by fixing the maximum allowed random route length (w_{max}), advertisements from adversarial nodes and their Sybils can reach at most w_{max} honest nodes with each attack edge.

Each relay's advertisement contains the necessary information required to access that node: its IP address, port number and an access cookie. Relays and exit points do not act as open proxies: a client must present the advertised cookie in order to forward traffic via the corresponding relay. A relay generates a new access cookie every few days and expires the old ones. To forward advertisements, each node constructs a routing table that matches each of its neighbors with another randomly chosen one, e.g. $N_x \rightarrow N_y$. Upon receiving an advertisement from neighbor N_x, a node forwards it to N_y. Each node generates the routing table *a priori* and saves it on the local disk so the same table is used across machine reboots.

The maximum number of distinct random routes a node can perform is bounded by its degree, which can vary a lot for different nodes. For example, 10% of nodes in the YouTube social network have degrees higher than 10 and 37% of nodes have only one neighbor. To accommodate such degree differences, we give each relay some flexibility in choosing its route length (w) with the constraint that $w_{min} \leq w \leq w_{max}$. Nodes with a degree smaller than r/w_{max} advertise with route length w_{max} via all their neighbors but might not reach the target of r nodes. We have to keep w_{max} small in order to bound the number of adversarial advertisements that can propagate via each attack edge to no more than w_{max}. A node with degree greater than r/w_{min} chooses a random subset of r/w_{min} neighbors to send advertisements through. The identities of the chosen neighbors are saved to disk so the same routes are performed across reboots. The lower limit w_{min} is necessary to ensure that a relay's random route can reach a spread-out neighborhood in the trust network.

In order to achieve the target reach r, a random route of length w should ideally visit w distinct nodes. To increase the chances of a route visiting unique nodes, routing tables must not have entries so that a node routes to itself. There is a simple heuristic for achieving this. A node computes a random permutation $\wp$ among its d neighbors as $N_{\wp(1)}, N_{\wp(2)}, ...N_{\wp(d)}$ and creates routing table entries of the form $N_{\wp(i)} \rightarrow N_{\wp(i+1 \mod d)}$ for all $i \in [1..d]$. By not routing packets received from the same neighbor to itself, a route never visits the same node in two hops. However, a route might still visit the same node in three or more hops. Specifically, the probability of a route visiting the same node in three hops is

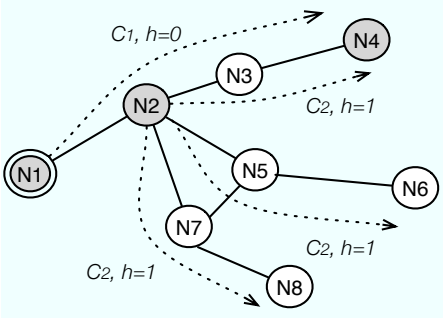

Figure 4: N1 (an exit point) advertises its access cookie C_1 and exit hop-count ($h = 0$) along a random route (shown in dotted lines). Similarly, the relay N2 advertises its cookie C_2 and exit hop-count ($h = 1$) along multiple random routes. To find a multi-hop path to some relay, a node (e.g. N6) sets up a tunnel to the known relay with the minimal hop-count (N2) which repeats the process until an exit point (N1) is reached.

c/d, where c is the clustering coefficient. The clustering coefficient measures the probability that two neighbors of the same node are neighbors themselves and is typically around 0.3 in measured social networks [28]. Therefore, the probability of a route visiting the same node in three or more hops is small.

We expect the trust network to change infrequently as the underlying social relationships tend to remain stable over long periods of time. Nevertheless, topology changes will occur occasionally, e.g. when new users join the system. Incorporating changes with minimal modifications to existing routing tables is straightforward [40]: when a node establishes a trust link with a new neighbor N', it replaces a randomly chosen routing table entry $N_x \rightarrow N_y$ with two new entries $N_x \rightarrow N', N' \rightarrow N_y$. Similarly, when a node abolishes an existing neighbor N', it merges the two affected routing entries $N_x \rightarrow N'$ and $N' \rightarrow N_y$ into one entry $N_x \rightarrow N_y$. It is easy to see that these routing table changes preserve the property that no neighbor is matched to itself.

4.2 Traffic forwarding

It is not enough to rely on relay advertisements alone to provide good service coverage. In a network with n nodes, a fraction p of which are exit points, the probability that a node receives an advertisement from at least one exit point is $1 - (1 - \frac{r}{n})^{p \cdot n}$, assuming simplistically that each exit point's advertisements reach r random nodes. The resulting service coverage is only 63% for large n, when $r = 100$ and as many as 1% of all nodes ($p = 0.01$) operate as exit points. Using a larger r does not solve the low coverage problem: a larger r causes more exit points to be discovered by the censor and allows the censor to potentially track more clients. A better alternative is to forward traffic via multiple hops to an exit point. Since each hop along the path only needs to know the address of its next hop, many more clients will be able to use an exit point without directly knowing its address.

To enable multi-hop forwarding, every node maintains a local forwarding table that includes each known relay's address with that relay's exit hop-count. The exit hop-count of a relay measures the number of forwarding hops required to reach some exit point. A relay includes its exit hop-count in periodic advertisements. An exit point advertises a hop-count of zero; a relay's hop-count is one plus the minimum hop-count of all the relays found in its forwarding table. This advertisement protocol resembles that of a simple distance vector routing protocol [30], except that Kaleidoscope does not include the final destination (i.e., the exit point) in its advertisements. As latencies affect users' browsing experience, long forwarding paths are not useful. Thus, we limit the maximum forwarding hops to be $h_{max} = 3$. Relays with hop-count $\geq h_{max}$ do not further advertise their service.

When forwarding the actual data traffic, a client first attempts to set up a multi-hop tunnel by contacting the known relay R with the smallest hop-count in its forwarding table. Upon checking that the client has the correct access cookie, R further contacts another relay in its forwarding table with a smaller hop-count. The process is repeated until some exit point is reached. The resulting forwarding tunnel consists of at most three hops.

Parameter	Meaning	Default Value
r	Targeted number of nodes a relay's advertisements should reach.	100
h_{max}	Maximum traffic forwarding hops.	3-4
w_{max}	Maximum allowed random route length.	20
w_{min}	Minimum route length used by a relay.	7

Table 1: A glossary of Kaleidoscope's system parameters and their default values.

Figure 4 gives an example of the relay advertisement and traffic forwarding process. The trust links between nodes are shown in solid lines while the dotted lines denote the random routes traversed by advertisement messages. In the example, N1 is the only exit point and N2,N4 are relays. Since N1 has only one neighbor, it performs a single short random route to advertise its access token C_1 and IP/port information. N2's advertisement traverses three random routes and contains N2's access cookie, C_2, and exit hop-count ($h = 1$). To forward the actual data traffic, N6 uses C_2 to set up an encrypted tunnel with N2 which extends the path by tunneling to the exit point N1 using access cookie C_1.

The power of multi-hop traffic forwarding lies in its ability to increase service coverage without requiring each exit point to advertise its identity to more nodes. For example in Figure 4, all nodes are able to use N1's exit service even though N1 is only directly known to three nodes (N2,N3,N4). Multi-hop forwarding is vulnerable to timing analysis: if the censor can observe and correlate traffic going in and out of all nodes simultaneously, it could deduce the address of the exit point used by an adversarial node even if the forwarding path has multiple hops. One can mitigate timing analysis using known techniques [34]; our current implementation does not yet address timing attacks.

4.3 Parameter analysis

Two critical parameters that affect the design of Kaleidoscope are: (a) r, the target reach threshold and (b) h_{max}, the maximum number of forwarding hops. Both these parameters are dependent upon the maximum fraction of infiltration that Kaleidoscope hopes to handle. We now provide a simplified analysis that provides intuition on how to set these parameters. The analysis assumes that adversarial nodes collect relay addresses by receiving their advertisements. Exposed relays and exit points are blocked by the censor. The analysis is done for a *regular* graph where each node has exactly d neighbors. Furthermore, we make the simplifying assumption that a relay's advertisements reach r *random* nodes in the system. These simplifications are not true in practice. However, as we will show in Section 5, the resulting analysis still approximates the actual simulation results well.

The target reach of a relay (r) is dependent on f, the level of infiltration to the trust network denoting the fraction of all attack edges between the censor and honest users. We model a censor's infiltration process by randomly choosing nodes from the graph to collude with the censor and mark all colluders' edges as attack edges. Thus, in a network where a fraction f of links are attack edges, $f/2$ fraction of all nodes are colluding with the censor.

Observation 4.1 *Let X be the number of nodes receiving advertisements from a single unexposed relay or exit point, the expected value of X is* $E(X) = r(1 - \frac{f}{2})^r$*, where* f *is the fraction of attack edges and* r *is the target reach of a relay.* $E(X)$ *is maximized when* $r = \frac{-1}{ln(1-\frac{f}{2})} \approx 2/f$.

Choosing r, w_{max}, w_{min}: For any relay, the probability that none of r recipients is adversarial is $(1 - \frac{f}{2})^r$ ($f/2$ is the fraction of colluding nodes). Therefore, the expected number of advertisements from an unexposed relay is $E(X) = r(1 - \frac{f}{2})^r$. The optimal r for which $E(X)$ is minimized is $r_{opt} = -1/ln(1 - \frac{f}{2}) \approx 2/f$. In our current system, we pick $f = 2\%$ (which represents an extremely high degree of infiltration) as the attacker infiltration threshold. Correspondingly, we set $r = 2/f = 100$ as a conservative estimate for r. The value of w_{max} is set based on r/d. We observe a median degree of nearly 5 in many social networks; hence we set $w_{max} = 20$. The choice of w_{min} is dependent on

Network	Nodes $\times 1000$	Edges $\times 1000$	Avg:Med. Degree	Cluster Coeff.
YouTube [29]	439	1022	4.7 : 2	0.06
Flickr [28]	1121	2953	5.3 : 1	0.11
LiveJournal [28]	4028	21936	10.9 : 5	0.22
Synthetic [36]	1121	3195	5.7 : 5	0.19

Table 2: Characteristics of sanitized social networks and the synthetic model network used in the evaluation.

the diameter of the social network. Since social networks typically satisfy the fast mixing property [39], setting a minimum path length of $w_{min} = 7$ allows each random route to explore and advertise a relay node to random nodes within the social network.

Choosing h_{max}: Note that the optimal value of r is independent of the network size. Multi-hop forwarding across relays becomes essential only when the network contains very few exit-points. The value of h_{max} dictates the maximum outreach of any exit point; for a given h_{max}, in the best case topological scenario, an exit point may be visible to $r^{h_{max}}$ users. We suggest $h_{max} = 3$ or $h_{max} = 4$ for common use cases. For $h_{max} = 3$ and $r = 100$, Kaleidoscope can scale up to a 1 million user network with few exit points. Choosing a larger value of h_{max} incurs performance overhead as well as runs the risk of exposing too much encrypted traffic to the censor if the censor analyzes traffic flow at several points within the censored domain.

The *service coverage* of Kaleidoscope is the probability that a node is able to find a multi-hop path along unexposed relays to some exit point. Since it is hard to obtain a closed form expression for the service coverage, we derive a recursive and conservative estimate based on the the fraction of exit points (p), relay availability (q, the probability that a known relay is online) and the fraction f: (Derivation details are in the appendix):

Observation 4.2 *The probability that a node finds a working exit path with $\leq j$ hops is $\Pr_j\{service\} = 1 - (1 - \frac{r}{n})^{\sum_{i=0}^{j} R_i}$, where r is the reach of a single relay, n is the number of nodes, and R_i represents the number of unexposed, working i-hop relays. R_i can be expressed as a recursive formula: $R_i = y \cdot q \cdot (n - \sum_{k=0}^{i-1} R_k) \cdot (1 - (1 - \frac{r}{n})^{R_{i-1}})$, where y is the probability that a single relay is exposed and R_i is the number of unexposed exit points. In particular, $y = (1 - \frac{f}{2})^r$ and $R_0 = p \cdot n \cdot y$, where q is the relay availability.*

Observation 4.2 helps us understand the importance of parameter choices such as the number of relay hops. To give a concrete example, in a network with 100000 nodes of which 0.5% are exit points, and 2% of all edges are attack edges, for an observed reach $r = 40$ (the median reach for the YouTube topology discussed in Section 5.2), the service probability is approximately 80% when using 1-hop relays with only 30% availability of relay nodes. When allowing 2 relay hops, the service probability increases to above 90%.

5 Evaluation

We evaluate the performance of Kaleidoscope using simulations on a number of real world social networks. The goal of our evaluation is to examine how well Kaleidoscope provides exit service to clients inside the censored domain and how restricted service discovery defends against a censor that achieves different levels of infiltration to the trust network. Our experiments support the following conclusions:

1. Kaleidoscope helps most nodes find a multi-hop forwarding path to some exit point. The service coverage is high even though only a fraction of nodes act as relays. (Section 5.2)
2. Kaleidoscope degrades gracefully as the censor manages to control more attack edges in the trust network. Even when the fraction of attack edges reaches 0.5%, Kaleidoscope manages to provide service to >90% nodes in most networks. (Section 5.3)
3. Kaleidoscope sustains infrequent changes in the trust network without exposing many extra relays to the censor over time. (Section 5.4)

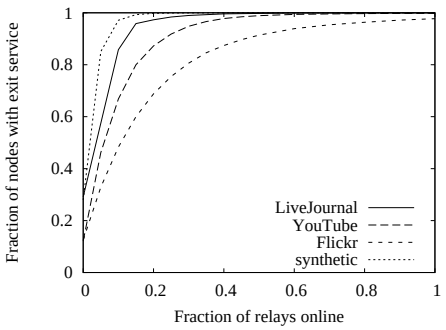

Figure 5: (Left) The fraction of nodes that have access to some exit point grows quickly when more relays are online. Experiments include 0.5% nodes as exit points and no adversarial nodes.

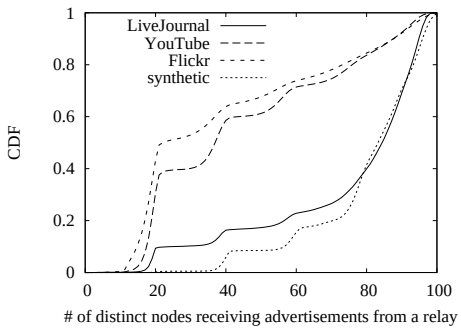

Figure 6: (Right) The distribution of the actual number of distinct nodes that receive advertisements from a relay. The presence of many low degree nodes causes many relays to reach much fewer nodes than the target r (r=100, $w_{max} = 20$).

5.1 Experimental setup

We examine the performance of Kaleidoscope on a number of network traces collected from online social networking sites [28] as well as a synthetic social network model [36]. These network traces are not ideal for evaluating Kaleidoscope: social networking users create links to share pictures/videos/blogs with friends or acquaintances while we expect a Kaleidoscope user to create a link to a trusted friend if she believes the friend will not collude with the censor. We sanitize the traces collected from social networking sites to better match the deployment scenario of Kaleidoscope. First, since Kaleidoscope uses bi-directional trust relationships, we prune all the uni-directional links between a pair of nodes. Second, the Kaleidoscope software restricts the maximum node degree to be 50 to encourage users to choose neighbors carefully. Thus, we prune redundant links so that each node has no more than 50 neighbors. Since only a very small fraction of nodes have degrees larger than 50, only a small fraction of links are eliminated. Table 2 shows the basic statistics of the sanitized social networks as well as the synthetic network.

We configure Kaleidoscope to use the default parameter values in Table 1. In all experiments, we select 0.5% nodes at random to act as exit points. We use a small percentage of nodes as exit points since users tend to be wary about providing web access for others despite a flexible exit policy. On the other hand, since relays require little configuration and can reside both within and outside the censored domain, we expect that more users are willing to operate their nodes as relays.

5.2 Service coverage

The main power of Kaleidoscope lies in its use of relays to forward traffic over multiple hops to exit points. For an end user to gain access to an exit point, it is necessary that all relays along at least one relay path be online at the same time. Figure 5 shows the fraction of client nodes that find an exit path of no more than three hops as a function of the relay availability. In the simulator, a potential relay node is brought online with probability q. There are no adversarial nodes in these experiments. The service coverage of Kaleidoscope increases quickly as more relays are present in the system. When more than 40% relays are online, the service coverage is very high ($\sim$98%) for all networks except Flickr.

The reason for why Kaleidoscope performs much worse on Flickr than other graphs is because Flickr has a particularly sparse network with median node degree of only one. As a result, half of the relays only manage to advertise themselves to w_{max}=20 or fewer other nodes, causing Flickr to have much worse service coverage than the other networks in the absence of adversarial nodes. Figure 6 presents the cumulative distribution of the number of distinct nodes that receive an advertisements from a relay or exit point. The smaller the actual reach is in Figure 6, the worse the corresponding service coverage is in

Figure 5. The "bumps" of various lines are due to a significant fraction of nodes with low degrees (1, 2 or 3). Although the presence of many low degree nodes impair the overall performance of Kaleidoscope, a low degree node is also less likely to receive exit service itself. We believe such a correlation between service and node degree can be a good incentive for users to establish more trust links.

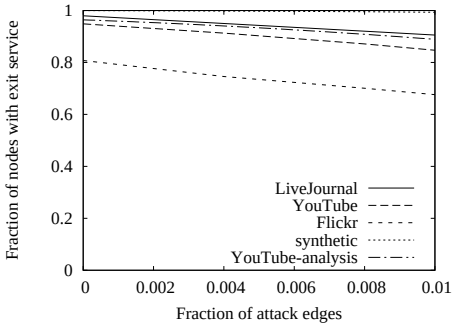

Figure 7: (Left) The fraction of nodes that find an exit path of unexposed relays and exit points as a function of increasing censor infiltration.

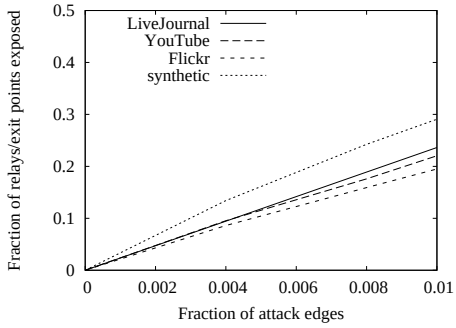

Figure 8: (Right) More relays and exit points are exposed as the fraction of attack edges increases. Despite many exposed relays, high service availability is achieved using multi-hop traffic forwarding.

5.3 Attack resilience

Restricted service discovery limits the censor's ability to discover relays and exit points according to the number of attack edges. We simulate varying levels of censor infiltration by choosing a random fraction of nodes that collude as adversarial nodes. All the edges of an adversarial node link to honest nodes, therefore, the fraction of attack edges among all links is twice the fraction of adversarial nodes. All experiments assume 30% of relay nodes are online.

We first evaluate the case of passive adversarial nodes. Passive adversarial nodes do not actively advertise themselves as relays but only receive others' advertisements and block the exposed relays and exit points. Figure 7 shows the service coverage as a function of increasing censor infiltration. The probability of a node receiving exit service decreases slowly as the fraction of attack edges grows to 1%, which is twice the fraction of exit points (0.5%). Even when the censor manages to attach as many as 0.5% attack edges, the service availability remains high for all networks except Flickr: more than 90% of users can still find paths of unexposed relays and exit points. Again, service availability is much lower for Flickr because its network is extremely sparse.

Figure 7 also includes the analytical results for the YouTube network. The analysis is based on a slightly more complicated version of Observation 4.2 augmented to account for topology-specific phenomena. Specifically, we used the observed reach (r) in the simulations to calculate the service coverage. In addition, we account for message overlapping in the following way: Observation 4.2 assumes that each advertisement reaches r random nodes. However, we know that by our advertisement protocol, a node that receives a j-hop advertisement message from its immediate neighbor N_1 and forwards that message to another neighbor N_2, might also send out a (j+1)-hop advertisement message. This message will surely be sent to N_1 and N_2 among its other neighbors. However, we know that these specific messages sent through N_1 and N_2 are not going to cause any new nodes to become relays, and thus, need to be discounted from the reach r. Mathematically we approximate this type of overlapping by scaling down the target reach r of all relay nodes excluding 0-hop relays (exit points) by a factor of $(d-2)/d$, where d is the average degree of that topology. The discounted 2 edges represent the "lost" paths through nodes N_1 and N_2 in the above explanation. Note that this approximation over-discounts paths from relays at the end of a message path (nodes that do not forward received messages any further), however there are relatively few such nodes. As can been seen from the graph, this analysis gives a reasonable approximation

of the simulation results (the results hold for all topologies in the graph).

Figure 8 shows the actual fraction of relays and exit points exposed as a function of increasing attack edges. As we can see, $20 - 30\%$ relays and exit points are exposed when the fraction of attack edges is 1% for all networks. Despite the large number of exposed relays, the overall service availability remains high ($> 90\%$) because of the effects of multi-hop traffic forwarding. In particular, the majority of nodes have a working exit path via at least one hop of intermediate relays.

When adversarial nodes actively participate as decoy relays or exit points, they could learn of unsuspecting clients by actually forwarding traffic for them. The ability of the censor to pollute the network with decoy relays' information is limited by the number of attack edges times the system parameter w_{max}, regardless of the number of Sybil identities it can create. Figure 9 shows the fraction of client nodes that could potentially contact an adversarial node for exit service. We regard a client node as trackable if it has received at least one advertisement from some adversarial nodes. As seen in Figure 9, the number of trackable nodes can be significant when the censor controls more than 0.3% attack edges. For accessing objectionable content, a client should consider using an anonymity system in combination with Kaleidoscope to avoid exposing its identity to the censor.

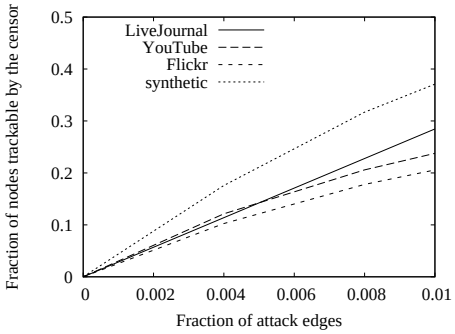

Figure 9: (Left) The number of clients that could potentially request service from a decoy exit point is bounded by the number of attack edges and w_{max}.

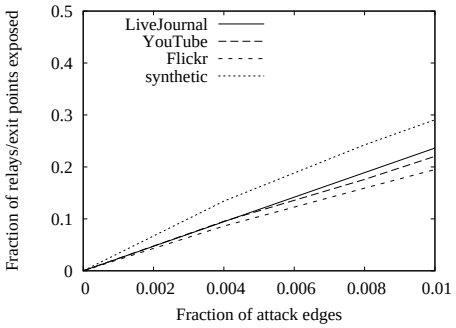

Figure 10: (Right) When new nodes join the network and activate trust links, the censor can learn of more relays via each of its attack edge as a result of the routing table changes at honest nodes. However, the rate of increase in the censor's knowledge is slow. (0.5% attack edges).

5.4 Effects of a changing topology

Kaleidoscope exploits the relative static nature of the social network to achieve restricted service discovery. Changes in the trust network could expose more relays. When a node establishes or deletes a trusted neighbor, it modifies its routing table entries accordingly. These modifications might cause an existing random route to change its course so new advertisements from an existing relay will reach a different set of nodes, causing more nodes to learn of the relay's address than it intends to.

We simulate the scenario of new users joining Kaleidoscope, which is probably the biggest source of changes in the network. The experiments contain 0.5% attack edges. We start each experiment with half of the node population already present. The initial population is selected by performing a breadth-first-search from a random node until the resulting network includes half of the nodes. During an experiment, we add one remaining node at a time by activating its trust links to those nodes that are already present in the network. Each newly activated trust link causes an existing node to change its routing table and one of the random routes going through that node will be routed differently as a result. Each change in a random route might give away additional relay information to adversarial nodes that are already present in the network. Figure 10 shows the number of relays exposed per attack edge as a function of the number of trust links activated over time. As expected, the censor can collect information on more

Network	Direct connection to an exit point	One hop to an exit point	Two hops to an exit point
YouTube	0.14	0.42	0.44
Flickr	0.15	0.41	0.44
LiveJournal	0.24	0.39	0.37
Synthetic	0.15	0.56	0.29

Table 3: Fraction of serviced users at different distances from an exit point.

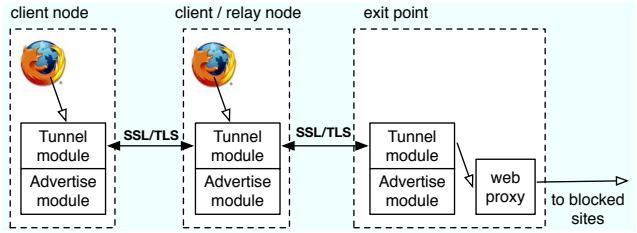

Figure 11: The software architecture of Kaleidoscope. The client browser communicates with the local tunnel module which further forwards the traffic using SSL/TLS over multiple hops to an exit point. The exit point runs a separate web proxy to access web sites allowed by the node's exit policy.

relays due to network changes. Nevertheless, the rate of the increase in the censor's knowledge is slow, indicating that topology changes are unlikely to impair Kaleidoscope's performance. In practice, the IP address of a relay could also change slowly over time which serves to delete old information collected by the censor.

5.5 Distance to an exit point

Table 3 shows the distribution of distances to an exit point in hop-count for all nodes with a working exit path. Since an exit point needs to make an additional connection to the origin web server, a user traverses at most $h_{max} = 3$ hops to reach the origin web server. As expected, most users do not know an address of an unexposed exit point. However, for all networks more than 10% of the serviced nodes can reach an exit point directly and more than half of the serviced nodes can reach an exit point via at most one relay. This experiment assumes 30% of relay nodes are online, 2% attack edges and 0.5% exit points.

6 Implementation and Deployment Experiences

Kaleidoscope has been fully implemented and available for public download since 9/2008. This section describes the implementation details and our experiences in running a small scale deployment over a six-month period.

6.1 Implementation

Kaleidoscope runs as a daemon process that operates in one of three modes: exit point, relay/client or client-only. Currently, only nodes with non-NATed addresses can operate as relays or exit points. We have built a Firefox plug-in to help users turn traffic forwarding on and off easily from within the browser. The plug-in also serves as the GUI for configuring Kaleidoscope.

Figure 11 illustrates the overall architecture of the implementation. Kaleidoscope consists of two modules: the advertisement module and the tunnel module. The advertisement module is in charge of communicating with a node's trusted neighbors via UDP to learn the address information of other relays and their exit points. During the actual data forwarding, the browser sends plain web traffic to the local tunnel module which relays the traffic over a chain of SSL/TLS tunnels to the exit point. The exit point runs a separate web proxy which is configured to only accept local connections and is not accessible to non-Kaleidoscope users and port scanners.

The implementation must address several practical challenges to be resilient against blocking. Below,

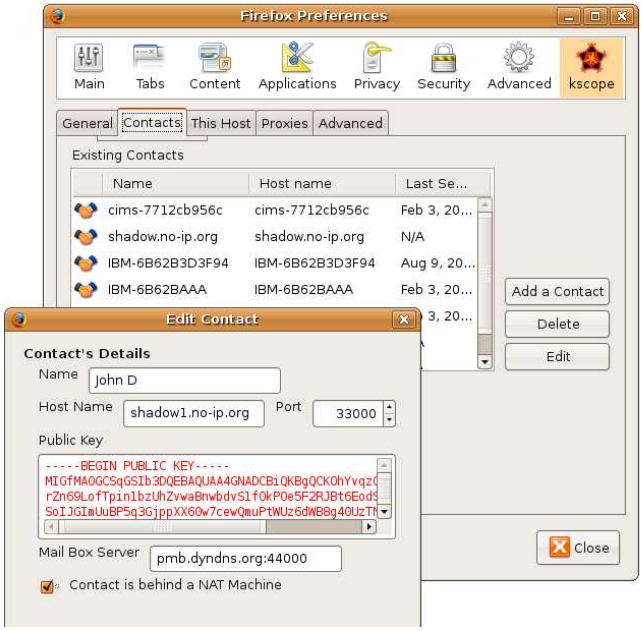

Figure 12: Kaleidoscope's Contacts Management window and Add-Contact dialog. Kaleidoscope's configuration options and controls are integrated into the Firefox web browser. (*kscope* is Kaleidoscope's code name.)

we discuss these challenges and our solutions.

Bootstrapping the trust network. Upon first installation, Kaleidoscope generates a new public/private key-pair. Users establish trust relationships by exchanging public keys with each other. The advertisement module uses the public key of a neighbor to authenticate its communication with that neighbor. The current implementation relies on manual key exchange. Figure 12 shows a screen shot of the configuration window for managing trusted neighbors. To add a neighbor, the user pastes the neighbor's details including the public key received over emails into a configuration box.

Handling dynamic IP addresses. To communicate with trusted neighbors, a node must learn of their IP addresses. Since a large number of home users have dynamic IP addresses, Kaleidoscope needs to automatically track IP address changes over time. There are two requirements for mapping a trusted neighbor to its current address: first, the mapping mechanism itself must not be blocked easily. Second, the censor should not be able to discover the IP addresses of Kaleidoscope users easily.

Our current implementation recommends using dynamic DNS for managing IP address changes. There are many existing dynamic DNS services offering hundreds of domain names and many services are free of charge. Users register dynamic DNS domain names of their choice and exchange them along with public keys among trusted friends as part of the out-of-band bootstrapping process (see Figure 12). We assume the censor is unlikely to block all dynamic DNS services nor pollute all dynamic DNS domain names since most clients that use dynamic DNS services do not run Kaleidoscope. A promising alternative is to encrypt a node's current IP address using its neighbors' keys and publish the encrypted addresses on websites that host user-generated content. One can even employ steganography techniques to hide encrypted content in uploaded images [9].

Handling nodes behind NATs. When a pair of users both lie behind NAT, they cannot forward advertisements via normal methods. Standard techniques for initiating communication to nodes behind NATs rely on centrally managed rendezvous servers (e.g. STUN [33] and TURN [32]) and thus are not suitable. Our implementation employs nodes with non-NATed addresses as mailboxes to pass encrypted UDP advertisement packets for users behind NATs.

Any non-NATed node can operate as a mailbox to temporarily store encrypted messages for others. Like proxies and relays, a mailbox announces its service via periodic advertisements. To forward an advertisement to a NATed neighbor, a node deposits encrypted UDP messages at both of the neighbor's

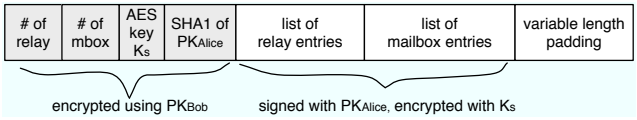

Figure 13: The format of a UDP advertisement message from Alice to Bob. The message header is shaded in gray and an AES key is generated for each message. The body contains a list of relay entries each of the form (IP address, port, access cookie, random route TTL, exit hop-count) and a list of mailbox addresses.

mailboxes. A NATed neighbor periodically requests deposited messages from its mailboxes. A NATed node tries to maintain two working mailboxes at all times: if a chosen mailbox becomes unavailable, the node replaces it with another discovered mailbox and notifies its neighbors of the change. The Kaleidoscope software distribution includes the identities of two initial mailboxes. As Figure 12 shows, users with NATed nodes exchange the identities of their initial mailboxes as part of the out-of-band bootstrapping process. The mailboxes included in the software distribution could be blocked by the censor, but such blockage only impacts those NATed nodes that have not yet learnt of any other mailboxes from Kaleidoscope. As nodes do not remain online all the time, mailboxes can also be optionally used by non-NATed nodes to speed up advertisement forwarding.

Hiding network fingerprints. Since the censor controls the underlying communication infrastructure, Kaleidoscope must make sure that its traffic leaves no distinguishable fingerprints. To prevent port-based filtering, the software installer randomly selects unused ports for the advertisement and tunnel module upon first installation and uses port 443 for the tunnel module if it is available. Figure 13 shows the format of an advertisement message in the UDP payload. The advertisement message is encrypted and has a variable size so it cannot be filtered based on simple pattern matching. The communication between tunnel modules is done using standard SSL/TLS. Nodes use self-signed certificates with randomly chosen Common Names to avoid leaving identifiable patterns during the SSL/TLS handshake. The current implementation only defends against known filtering techniques such as pattern matching or port-based filtering. More sophisticated traffic analysis is possible to identify and block encrypted Kaleidoscope advertisements and tunnels. Even though we have not implemented any known defenses to traffic analysis [34], the current implementation still raises the bar for Internet censorship.

6.2 Deployment Experiences

The Kaleidoscope software has been available for public download since September, 2008. We did not publicize Kaleidoscope widely because we are still addressing usability issues in software installation and upgrade on various types of client machines. We personally recruited friends from different countries with known censorship practices to run Kaleidoscope. At least 40 users have installed and used Kaleidoscope since our latest software release on March, 2009. Kaleidoscope gives users the option to automatically submit usage logs to our server. Unfortunately, because of the sensitive nature of collecting usage statistics, most users did not turn on this option. Based on the two Kaleidoscope logs at the authors' nodes, we noticed at least 20 active users at any given time in the current deployment. We correlate an anonymized proxy log from an exit point with the Kaleidoscope log from that exit point. Among all IP addresses observed in the proxy logs, approximately half of them have been used by that exit point's direct neighbors according to the Kaleidoscope log. The rest of the IP addresses belong to nodes that are at least 2 hops away from the exit point in the trust graph.

Table 4 shows website access statistics gathered from the proxy log of an exit point. This exit point has served over 100,000 web requests which represents fairly heavy usage because users typically turn off Web access via Kaleidoscope using our simple "turn-off option" within the browser when surfing unblocked websites. On the other hand, not all web pages accessed via Kaleidoscope are necessarily blocked at the requesting nodes because users might forget to re-enable direct access after using Kaleidoscope to access some blocked sites. The most common types of content accessed via Kaleidoscope were news and video media from popular web sites. Many sites in these categories, e.g. `mitbbs.com`, are known to be blocked in some countries. Anecdotally, we are also surprised to find out that many users are also using Kaleidoscope to circumvent the access control mechanism implemented by some websites to make content available only to users in particular geographical regions.

Types of Websites	# of requests	# of distinct domains	Example site
News (English)	11,814	2	*nytimes.com*
News (other languages)	33,118	12	*mitbbs.com*
Video/Photos	19,630	12	*youtube.com*
Information and searching	15,851	12	*wikipedia.org*
Universities	13,151	13	*kaist.ac.kr*
Other	13,695	368	
Total	107,259	419	

Table 4: Types of Web content accessed via Kaleidoscope during a small scale deployment from 03/09-09/09. The statistics presented were gathered from a single exit point.

Lessons learned. Our deployment experience has taught us two lessons. First, manual bootstrapping of trust relationships is non-trivial for most users. Kaleidoscope only requires a user to copy and paste a text message obtained from an friend's email to establish a trust link. Despite its simple interface, this process seems cumbersome enough to deter many users from successfully establishing trust links. To achieve a larger scale deployment, we need to explore some alternative means to automatically establish trust links. For example, Kaleidoscope nodes could exchange public keys by publishing information on popular social networks. Second, even though the implementation is based on Java and Firefox plugin, Kaleidoscope is not as portable as we had hoped. The vast diversity of client machine installations has caused numerous issues such as JVM version mismatches, missing Firefox installations, wrong Firefox versions etc. We are currently investigating a self-contained distribution of Kaleidoscope on USB keys.

7 Discussion

Implications of discovering exit points: A censor can trigger web requests from a client node under its control to a website it owns in the hope of determining the identity of the exit point in use. Blocking exit points discovered in this fashion is not a serious problem. This is because Kaleidoscope relies on a large number of relays outside the censored domain to forward traffic to some exit points. As the censor can only block the direct access to a discovered exit point from within the censored domain, relays outside the censored domain can continue to use such an exit point.

User incentives: In addition to mitigating Sybil attacks, social networks also provide good incentives for users to help out their friends or even friends of friends to access blocked websites. Since sharing information with an ill-chosen friend might cause one's own relays to be blocked, there is also a strong incentive to choosing one's trust links carefully. When there are not enough users, Kaleidoscope is fragmented into many "islands" of trust each consisting of a small group of users. Unlike most peer-to-peer applications that are useful only when they reach a certain scale, a small Kaleidoscope network can be as helpful as a large one so long as there is some exit point in the "island".

Relationship with anonymity systems: Kaleidoscope is designed to access gray material for which client anonymity is not required. In fact, the challenges of providing access are so different from those of anonymity that the resulting solutions are polar opposites. To be resilient against blocking, Kaleidoscope reveals each relay to a small subset of users and no user is allowed to discover many relays. In contrast, anonymity systems would like to keep the identities of all relays and traffic mixes public because the strength of anonymity depends on *all* users being able to use *all* servers to "blend in" with each other. The relay mechanism used by Kaleidoscope is also at odds with that required of an anonymity system. Relays in Kaleidoscope help an exit point serve more clients without revealing its address to them; in contrast, in anonymity systems [19, 23], it is more secure to let a client pick the forwarding path among all servers. Even in anonymity systems that let a forwarder pick the next hop (e.g. Crowds [31]), the next hops are chosen from the entire pool of nodes so that paths initiated by different clients can blend in with each other maximally. In Kaleidoscope, a relay does not choose the next hop randomly from all nodes, rather, it uses one that minimizes the exit hop-count.

Despite the fundamental design differences, anonymity systems and Kaleidoscope can complement each other. In particular, Kaleidoscope can be viewed as a decentralized gateway for providing access to a centrally managed anonymity system like Tor. Since Tor also uses multi-hop forwarding, anonymity comes at the cost of extra latency and should be selectively enabled by a user. Adding the Tor gateway feature to the current software distribution is part of the future work on Kaleidoscope.

8 Related Work

Most existing research on censorship circumvention has focused on censorship-resilient content publishing [15, 37, 38] and anonymous communication systems [12, 13, 19, 23, 25, 31]. Censorship-resilient publishing systems replicate censored content widely on many servers and thus are not suitable for web content as popular websites today host enormous dynamic content. Most existing anonymity systems are centrally managed and do not defend against an adversary that tries to block access to the system. As discussed earlier, anonymity systems can be integrated with Kaleidoscope to provide both access and anonymity.

In the censorship circumvention literature, relatively few proposals [18, 22, 26] address the access problem to gray material. All proposed solutions recognize that the key challenge in a proxy-based circumvention system is to restrict the number of proxies discoverable by each user so that a censor cannot learn of and block all proxies by posing as a few legitimate users. In [18], the Tor developers have proposed a variety of mechanisms to disseminate the identities of a small subset of bridge relays to clients using a centralized discovery service. Key-space hopping [22] restricts the proxies known to each client based on its IP address. Block-resistant JAP [26] distributes relay addresses via a mass email list consisting of potentially interested users. These proposals either use a centralized discovery service or require users to know a bootstrap proxy a priori. In contrast, Kaleidoscope operates in a completely distributed fashion and does not require bootstrap proxies. In fact, users could use Kaleidoscope as a bootstrap system to access more relays from existing centralized services. Furthermore, since there is no strong identity associated with each user, existing proposals rely on CAPTCHA puzzles or IP addresses to combat Sybil attacks. These defenses might not be effective against a governmental censor that owns abundant IP addresses and human resources.

Many proxies have been built to relay traffic to blocked websites, e.g. Psiphon [6], Circumventor [3]. Most of these proxies set up SSL/TLS connections for communicating with clients inside the censored domain. Infranet [21] obfuscates an encrypted communication channel using steganography in scenarios where the mere presence of an encrypted channel could cause the suspicion of a censor. Most proxy-based circumvention systems rely on the owners of volunteer proxies to manually disseminate a proxy's identity to her friends. Kaleidoscope automates such manual proxy dissemination processes and allows a user to discover proxies not operated by her immediate friends.

A number of existing systems have used the trust network to defend against Sybil attacks in the context of different applications such as admitting honest servers [39, 40], computing node reputations [14, 27], and fighting unwanted communication [24, 29]. The random route used by Kaleidoscope is inspired by SybilGuard [39, 40]. In SybilGuard/SybilLimit each node performs a few very long routes or many short routes to admit other honest nodes. Kaleidoscope uses random routes for a different purpose, namely, to disseminate each relay's address to a small subset of other nodes. As a result, a few short routes are needed for each relay.

WASTE [8] and other darknets allow users to connect to a group of trusted computers in order to exchange information. WASTE relies on strong encryption to secure the communication between members of small networks. Since WASTE is designed to share information and supports search features, it does not provide *restricted service discovery*. Unlike Kaleidoscope, WASTE creates a fully connected mesh of users by propagating their public keys. If a censor is able to join a WASTE network, it should be able to discover all users of that network. As does WASTE, Kaleidoscope supports using dynamic DNS for identifying hosts.

9 Conclusion

Kaleidoscope is a peer-to-peer system of relays that enables users inside a censored domain to access blocked content. Kaleidoscope provides restricted service discovery over a trust overlay where links

correspond the real world trust relationships between users. By disseminating relay addresses over the trust network and forwarding traffic over multiple hops to some exit point, Kaleidoscope provides good service when a censor achieves only a low level of infiltration to the trust network. We have implemented Kaleidoscope and deployed it among a small community of users for daily use. We believe Kaleidoscope is a promising solution for accessing a vast amount of blocked content on the Internet. Kaleidoscope is publicly available at `http://www.kspro.org`.

References

[1] Anonymizer. `http://anonymizer.com/`.

[2] Bbc news. china criticised for ban on google. `http://news.bbc.co.uk/1/hi/technology/2238236.stm`.

[3] Citizens Lab. Everyone's guide to bypassing internet censorship. `http://deibert.citizenlab.org/Circ_guide.pdf`.

[4] OpenNet Initiative. `http://www.opennet.net`.

[5] PC world. China blocks youtube, restores flickr and blogspot. `http://www.pcworld.com/article/id,138599-c,sites/article.html`.

[6] Psiphon. `http://psiphone.civisec.org`.

[7] Safeweb privacy proxy censored in china. `http://censorware.net/articles/01/03/14/0755209.shtml`.

[8] Waste. `http://wasteagain.sourceforge.net/`.

[9] A. Baliga, J. Killian, and L. Iftode. A web based covert file system. In *11th HotOS*, 2007.

[10] A. Banerjee, M. Faloutsos, and L. Bhuyan. Is someone tracking p2p users? In *IFIP Networking*, 2007.

[11] T. Blogs. Tor partially blocked in china. `http://blog.torproject.org/`.

[12] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[13] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology 1(1)*, pages 65–75, 1988.

[14] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, New York, NY, USA, 2005. ACM.

[15] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[16] R. Clayton, S. Murdoch, and R. N. M. Watson. Ignoring the great firewall of China. In *6th Workshop on Privacy Enhancing Technologies*, June 2006.

[17] J. Crandall, D. Zinn, M. Byrd, E. Barr, and R. East. Conceptdoppler: A weather tracker for internet censorship. In *14th ACM Conference on Computer and Communications Security*, 2007.

[18] R. Dingledine and N. Mathewson. Design of a blocking-resistant anonymity system. `http://www.torproject.org/svn/trunk/doc/design-paper/blocking.pdf`.

[19] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, Aug. 2004.

[20] J. Douceur. The sybil attack. In *IPTPS 2002*.

[21] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[22] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In *Privacy Enhancing Technologies Workshop*, Mar. 2003.

[23] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *ACM Conference on Computer and Communications Security (CCS 9)*, Nov. 2002.

[24] S. Garriss, M. Kaminsky, M. Freedman, B. Karp, D. Mazires, and H. Yu. Re: reliable email. In *Proceedings of the 3rd NSDI*, 2006.

[25] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[26] S. Kopsell and U. Hillig. How to achieve blocking resistance for existing systems enabling anonymous web surfing. In *Workshop on Privacy in the Electronic Society*, 2004.

[27] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998*, pages 18–18, Berkeley, CA, USA, 1998. USENIX Association.

[28] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and S. Bhattacharjee. Measurement and analysis of online social networks. In *7th Usenix/ACM SIGCOMM Internet Measurement Conference (IMC)*, 2007.

[29] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *NSDI'08: Proceedings of the 5th conference on 5th Symposium on Networked Systems Design & Implementation*, Berkeley, CA, USA, 2008.

[30] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 2003.

[31] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1998.

[32] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relays around NAT (TURN), Oct. 2008.

[33] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard), Oct. 2008.

[34] V. Shmatikov and M. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. *Lecture Notes in Computer Science*, 4189:18–33, 2006.
[35] N. Y. Times. Thailand bans youtube. `http://www.nytimes.com/2007/04/05/business/worldbusiness/05tube.html`.
[36] R. Toivonen, J.-P. Onnela, J. Saramäki, J. Hyvönen, and K. Kaski. A model for social networks. *Physica A Statistical Mechanics and its Applications*, 371:851–860, 2006.
[37] M. Waldman and D. Mazières. Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, 2001.
[38] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, Aug. 2000.
[39] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, 2008.
[40] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Defending against sybil attacks via social networks. In *Proceedings of ACM SIGCOMM Conference*, Sept. 2006.

10 Appendix

We show how we derive Observation 4.2 which estimates the probability of a node getting exit service in a n-node network with j-hop relays ($j \geq 0$), with a fraction of attack edges f, a fraction of exit points p, and a fraction q of relays being online. Let R_j denote the expected number of unexposed, working j-hop relay nodes (for each node, j is the minimal possible value). It follows that there are $\sum_{i=0}^{j} R_i$ unexposed relays advertising their service. Since, by our assumption, each relay reaches r distinct random nodes, the probability of an honest node receiving at least one message from a working relay is $\Pr_j\{service\} = 1-(1-\frac{r}{n})^{\sum_{i=0}^{j} R_i}$. One can think of this as the classic bins and balls problem where in each repetition r bins are chosen at random, and we are interested in the probability of a bin ending up with at least one ball in it. It remains to show how we estimate R_i. Let y be the probability that none of the r transmitted messages a relay sends end up at an adversarial node. By our attack edges selection process, there are, on average, $f/2$ randomly chosen adversarial nodes. Under the assumption that advertisements reach r random nodes, $y = (1-\frac{f}{2})^r$. We can estimate R_0 (unexposed exit points) as $R_0 = y \cdot p \cdot n$. To obtain R_1, we first calculate the probability of a node receiving an advertisement from some unexposed exit point. Again, we can view this problem as the classic occupation problem of throwing m balls into n bins. Therefore, the probability of a node receiving at least one of the $R_0 \cdot r$ advertisements from unexposed exit points is $1-(1-\frac{r}{n})^{R_0}$. However, first, only nodes that are not in R_k for $k < i$ will be included, and second, we need to account for the independent probability of a node acting as a relay (the churn factor q). Combining these details with the probability of 1-hop relays being exposed, we get $R_1 = y \cdot q \cdot (n - R_0) \cdot (1-(1-\frac{r}{n})^{R_0})$. By the exact same arguments, we can genreralize this approximation to $R_i = y \cdot q \cdot (n - \sum_{k=0}^{i-1} R_k) \cdot (1-(1-\frac{r}{n})^{R_{i-1}})$ for $i > 1$.