Sybil-Resilient Online Content Voting

Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian
New York University

Abstract

Obtaining user opinion (using votes) is essential to ranking user-generated online content. However, any content voting system is susceptible to the Sybil attack where adversaries can out-vote real users by creating many Sybil identities. In this paper, we present *SumUp*, a Sybil-resilient vote aggregation system that leverages the trust network among users to defend against Sybil attacks. SumUp uses the technique of *adaptive vote flow* aggregation to limit the number of bogus votes cast by adversaries to no more than the number of attack edges in the trust network (with high probability). Using user feedback on votes, SumUp further restricts the voting power of adversaries who continuously misbehave to below the number of their attack edges. Using detailed evaluation of several existing social networks (YouTube, Flickr), we show SumUp's ability to handle Sybil attacks. By applying SumUp on the voting trace of Digg, a popular news voting site, we have found strong evidence of attack on many articles marked "popular" by Digg.

1 Introduction

The Web 2.0 revolution has fueled a massive proliferation of user-generated content. While allowing users to publish information has led to democratization of Web content and promoted diversity, it has also made the Web increasingly vulnerable to content pollution from spammers, advertisers and adversarial users misusing the system. Therefore, the ability to rank content accurately is key to the survival and the popularity of many user-content hosting sites. Similarly, content rating is also indispensable in peer-to-peer file sharing systems to help users avoid mislabeled or low quality content [7, 16, 25].

People have long realized the importance of incorporating user opinion in rating online content. Traditional ranking algorithms such as PageRank [2] and HITS [12] rely on implicit user opinions reflected in the link structures of hypertext documents. For arbitrary content types, user opinion can be obtained in the form of explicit votes. Many popular websites today rely on user votes to rank news (Digg, Reddit), videos (YouTube), documents (Scribd) and consumer reviews (Yelp, Amazon).

Content rating based on users' votes is prone to vote manipulation by malicious users. Defending against vote manipulation is difficult due to the *Sybil attack* where the attacker can out-vote real users by creating many Sybil identities. The popularity of content-hosting sites has made such attacks very profitable as malicious entities can promote low-quality content to a wide audience. Successful Sybil attacks have been observed in the wild. For example, online polling on the best computer science school motivated students to deploy automatic scripts to vote for their schools repeatedly [9]. There are even commercial services that help paying clients promote their content to the top spot on popular sites such as YouTube by voting from a large number of Sybil accounts [22].

In this paper, we present SumUp, a Sybil-resilient online content voting system that prevents adversaries from arbitrarily distorting voting results. SumUp leverages the trust relationships that already exist among users (e.g. in the form of social relationships). Since it takes human efforts to establish a trust link, the attacker is unlikely to possess many attack edges (links from honest users to an adversarial identity). Nevertheless, he may create many links among Sybil identities themselves.

SumUp addresses the *vote aggregation problem* which can be stated as follows: *Given m votes on a given object, of which an arbitrary fraction may be from Sybil identities created by an attacker, how do we collect votes in a Sybil resilient manner?* A Sybil-resilient vote aggregation solution should satisfy three properties. First, the solution should collect a significant fraction of votes from honest users. Second, if the attacker has e_A attack edges, the maximum number of bogus votes should be bounded by e_A, independent of the attacker's ability to create many Sybil identities behind him. Third, if the attacker repeatedly casts bogus votes, his ability to vote in the future should be diminished. SumUp achieves all three properties with high probability in the face of Sybil attacks. The key idea in SumUp is the *adaptive vote flow* technique that appropriately assigns and adjusts link capacities in the trust graph to collect the net vote for an object.

Previous works have also exploited the use of trust networks to limit Sybil attacks [3, 15, 18, 26, 27, 30], but none directly addresses the vote aggregation problem. SybilLimit [26] performs admission control so that at most $O(\log n)$ Sybil identities are accepted per attack edge among n honest identities. As SybilLimit results in 10$\sim$30 bogus votes per attack edge in a million-user system [26], SumUp provides notable improvement by limiting bogus votes to one per attack edge. Additionally, SumUp leverages user feedback to further diminish the voting power of adversaries that repeatedly vote maliciously.

In SumUp, each vote collector assigns capacities to links in the trust graph and computes a set of approximate max-flow paths from itself to all voters. Because only votes on paths with non-zero flows are counted, the number of bogus votes collected is limited by the total capacity of attack edges instead of links among Sybil identities. Typically, the number of voters on a given object is much smaller than the total user population (n). Based on this insight, SumUp assigns C_{max} units of capacity in total, thereby limiting the number of votes that can be collected to be C_{max}. SumUp adjusts C_{max} automatically according to the number of honest voters for each object so that it can aggregate a large fraction of votes from honest users. As C_{max} is far less than n, the number of bogus votes collected on a single object (i.e. the attack capacity) is no more than the number of attack edges (e_A). SumUp's security guarantee on bogus votes is probabilistic. If a vote collector happens to be close to an attack edge (a low probability event), the attack capacity could be much higher than e_A. By re-assigning link capacities using feedback, SumUp can restrict the attack capacity to be below e_A even if the vote collector happens to be close to some attack edges.

Using a detailed evaluation of several existing social networks (YouTube, Flickr), we show that SumUp successfully limits the number of bogus votes to the number of attack edges and is also able to collect $> 90\%$ of votes from honest voters. By applying SumUp to the voting trace and social network of Digg (an online news voting site), we have found hundreds of suspicious articles that have been marked "popular" by Digg. Based on manual sampling, we believe that at least 50% of suspicious articles exhibit strong evidence of Sybil attacks.

This paper is organized as follows. In Section 2, we discuss related work and in Section 3 we define the system model and the vote aggregation problem. Section 4 outlines the overall approach of SumUp and Sections 5 and 6 present the detailed design. In Section 7, we describe our evaluation results. Finally in Section 8, we discuss how to extend SumUp to decentralize setup and we conclude in Section 9.

2 Related Work

Ranking content is arguably one of the Web's most important problems. As users are the ultimate consumers of content, incorporating their opinions in the form of either explicit or implicit votes becomes an essential ingredient in many ranking systems. This section summarizes related work in vote-based ranking systems. Specifically, we examine how existing systems cope with Sybil attacks [6] and compare their approaches to SumUp.

2.1 Hyperlink-based ranking

PageRank [2] and HITS [12] are two popular ranking algorithms that exploit the implicit human judgment embedded in the hyperlink structure of web pages. A hyperlink from page A to page B can be viewed as an implicit endorsement (or vote) of page B by the creator of page A. In both algorithms, a page has a higher ranking if it is linked to by more pages with high rankings. Both PageRank and HITS are vulnerable to Sybil attacks. The attacker can significantly amplify the ranking of a page A by creating many web pages that link to each other and also to A. To mitigate this attack, the ranking system must probabilistically reset its PageRank computation from a small set of trusted web pages with probability ϵ [20]. Despite probabilistic resets, Sybil attacks can still amplify the PageRank of an attacker's page by a factor of $1/\epsilon$ [29], resulting in a big win for the attacker because ϵ is small.

2.2 User Reputation Systems

A user reputation system computes a reputation value for each identity in order to distinguish well-behaved identities from misbehaving ones. It is possible to use a user reputation system for vote aggregation: the voting system can either count votes only from users whose reputations are above a threshold or weigh each vote using the voter's reputation. Like SumUp, existing reputation systems mitigate attacks by exploiting two resources: the trust network among users and explicit user feedback on others' behaviors. We discuss the strengths and limitations of existing reputation systems in the context of vote aggregation and how SumUp builds upon ideas from prior work.

Feedback based reputations In EigenTrust [11] and Credence [25], each user independently computes *personalized* reputation values for all users based on past transactions or voting histories. In EigenTrust, a user increases (or decreases) another user's rating upon a good (or bad) transaction. In Credence [25], a user gives a high (or low) rating to another user if their voting records on the same set of file objects are similar (or dissimilar). Because not all pairs of users are known to each other based on direct interaction or votes on overlapping sets of objects, both Credence and EigenTrust use a PageRank-style algorithm to propagate the reputations of known users in order to calculate the reputations of unknown users. As such, both systems suffer from the same vulnerability as PageRank where an attacker can amplify the reputation of a Sybil identity by a factor of $1/\epsilon$.

Neither EigenTrust nor Credence provide provable guarantees on the damage of Sybil attacks under arbitrary attack strategies. In contrast, SumUp bounds the voting power of an attacker on a single object to be no more than the number of attack edges he possesses irrespective of the attack strategies in use. SumUp uses only negative feedback as opposed to EigenTrust and Credence that use both positive and negative feedback. Using only negative feedback has the advantage that an attacker cannot boost his attack capacity easily by casting correct votes on objects that he does not care about.

DSybil [28] is a feedback-based recommendation system that provides provable guarantees on the damages of arbitrary attack strategies. DSybil differs from SumUp in its goals. SumUp is a vote aggregation system which allows for arbitrary ranking algorithms to incorporate collected votes to rank objects. For example, the ranking algorithm can rank objects by the number of votes collected. In contrast, DSybil's recommendation algorithm is fixed: it recommends a *random* object among all objects whose sum of the weighted vote count exceeds a certain threshold.

Trust network-based reputations A number of proposals from the semantic web and peer-to-peer literature rely on the trust network between users to compute reputations [3, 8, 15, 21, 30]. Like SumUp, these proposals exploit the fact that it is difficult for an attacker to obtain many trust edges from honest users because trust links reflect offline social relationships. Of the existing work, Advogato [15], Appleseed [30] and Sybilproof [3] are resilient to Sybil attacks in the sense that an attacker cannot boost his reputation by creating a large number of Sybil identities "behind" him. Unfortunately, a Sybil-resilient user reputation scheme does not directly translate into a Sybil-resilient voting system: Advogato only computes a non-zero reputation for a small set of identities, disallowing a majority of users from being able to vote. Although an attacker cannot improve his reputation with Sybil identities in Appleseed and Sybilproof, the reputation of Sybil identities is almost as good as that of the attacker's non-Sybil accounts. Together, these reputable Sybil identities can cast many bogus votes.

2.3 Sybil Defense using trust networks

Many proposals use trust networks to defend against Sybil attacks in the context of different applications: SybilGuard [27] and SybilLimit [26] help a node admit another node in a decentralized system such that the admitted node is likely to be an honest node instead of a Sybil identity. Ostra [18] limits the rate of unwanted communication that adversaries can inflict on honest nodes. Sybil-resilient DHTs [5, 14] ensure that DHT routing is correct in the face of Sybil attacks. Kaleidoscope [23] distributes proxy identities to honest clients while minimizing the chances of exposing them to the censor with many Sybil identities. SumUp builds on their insights and addresses a different problem, namely, aggregating votes for online content rating. Like SybilLimit, SumUp bounds the power of attackers according to the number of attack edges. In SybilLimit, each attack edge results in $O(\log n)$ Sybil identities accepted by honest nodes. In SumUp, each attack edge leads to at most one vote with high probability. Additionally, SumUp uses user feedback on bogus votes to further reduce the attack capacity to below the number of attack edges. The feedback mechanism of SumUp is inspired by Ostra [18].

3 The Vote Aggregation Problem

In this section, we outline the system model and formalize the vote aggregation problem that SumUp addresses.

System model: We describe SumUp in a centralized setup where a trusted central authority maintains all the information in the system and performs vote aggregation using SumUp in order to rate content. This centralized mode of operation is suitable for web sites such as Digg, YouTube and Facebook, where all users' votes and their trust relationships are collected and maintained by a single trusted entity. We describe how SumUp can be applied in a distributed setting in Section 8.

SumUp leverages the trust network among users to defend against Sybil attacks [3,15,26,27,30]. Each trust link is directional. However, the creation of each link requires the consent of both users. Typically, user i creates a trust link to j if i has an offline social relationship to j. Similar to previous work [18, 26], SumUp requires that links are difficult to establish. As a result, an attacker only possesses a small number of attack edges (e_A) from honest users to colluding adversarial identities. Even though e_A is small, the attacker can create many Sybil identities and link them to adversarial entities. We refer to votes from colluding adversaries and their Sybil identities as bogus votes.

SumUp aggregates votes from one or more trusted *vote collectors*. A trusted collector is required in order to break the symmetry between honest nodes and Sybil nodes [3]. SumUp can operate in two modes depending on the choice of trusted vote collectors. In *personalized vote aggregation*, SumUp uses each user as his own vote collector to collect the votes of others. As each user collects a different number of votes on the same object, she also has a different (personalized) ranking of content. In *global vote aggregation*, SumUp uses one or more pre-selected vote collectors to collect votes on behalf of all users. Global vote aggregation has the advantage of allowing for a single global ranking of all objects; however, its performance relies on the proper selection of trusted collectors.

Vote Aggregation Problem: Any identity in the trust network including Sybils can cast a vote on any object to express his opinion on that object. In the simplest case, each vote is either positive or negative (+1 or -1). Alternatively, to make a vote more expressive, its value can vary within a range with higher values indicating more favorable opinions. A vote aggregation system collects votes on a given object. Based on collected votes and various other features, a separate ranking system determines the final ranking of an object. The design of the final ranking system is outside the scope of this paper. However, we note that many ranking algorithms utilize *both* the number of votes and the average value of votes to determine an object's rank [2, 12]. Therefore, to enable arbitrary ranking algorithms, a vote aggregation system should collect

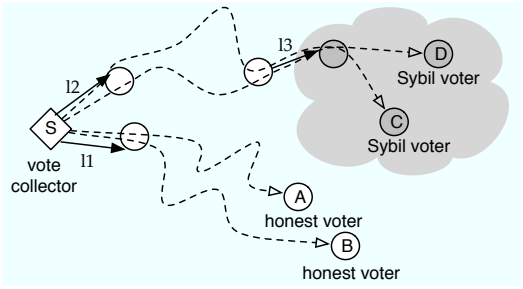

Figure 1: SumUp computes a set of approximate max-flow paths from the vote collector s to all voters (A,B,C,D). Straight lines denote trust links and curly dotted lines represent the vote flow paths along multiple links. Vote flow paths to honest voters are "congested" at links close to the collector while paths to Sybil voters are also congested at far-away attack edges.

a significant fraction of votes from honest voters.

A voting system can also let the vote collector provide *negative* feedback on malicious votes. In personalized vote aggregation, each collector gives feedback according to his personal taste. In global vote aggregation, the vote collector(s) should only provide objective feedback, e.g. negative feedback for positive votes on corrupted files. Such feedback is available for a very small subset of objects.

We describe the desired properties of a vote aggregation system. Let $G = (V, E)$ be a trust network with vote collector $s \in V$. V is comprised of an unknown set of honest users $V_h \subset V$ (including s) and the attacker controls all vertices in $V \setminus V_h$, many of which represent Sybil identities. Let e_A represent the number of attack edges from honest users in V_h to $V \setminus V_h$. Given that nodes in G cast votes on a specific object, a vote aggregation mechanism should achieve three properties:

1. Collect a large fraction of votes from honest users.
2. Limit the number of bogus votes from the attacker by e_A independent of the number of Sybil identities in $V \setminus V_h$.
3. Eventually ignore votes from nodes that repeatedly cast bogus votes using feedback.

4 Basic Approach

This section describes the intuition behind *adaptive vote flow* that SumUp uses to address the vote aggregation problem. The key idea of this approach is to appropriately assign link capacities to bound the attack capacity.

In order to limit the number of votes that Sybil identities can propagate for an object, SumUp computes a set of max-flow paths in the trust graph from the vote collector to all voters on a given object. Each vote flow consumes one unit of capacity along each link traversed. Figure 1 gives an example of the resulting flows from the collector s to voters A,B,C,D. When all links are assigned unit capacity, the attack capacity using the max-flow based approach is bounded by e_A.

The concept of max-flow has been applied in several reputation systems based on trust networks [3, 15]. When applied in the context of vote aggregation, the challenge is that links close to the vote collector tend to become "congested" (as shown in Figure 1), thereby limiting the total number of votes collected to be no more than the collector's node degree. Since practical trust networks are sparse with small median node degrees, only a few honest votes can be collected. We cannot simply enhance the capacity of each link to increase the number of votes collected since doing so also increases the attack capacity. Hence, a flow-based vote aggregation system faces the tradeoff between the maximum number of honest votes it can collect and the number of potentially bogus votes collected.

The *adaptive vote flow* technique addresses this tradeoff by exploiting two basic observations. First, the number of honest users voting for an object, even a popular one, is significantly smaller than the total number of users. For example, 99% of popular articles on Digg have fewer than 4000 votes which represents 1% of active users. Second, vote flow paths to honest voters tend to be only "congested" at links close to the vote collector while paths to Sybil voters are also congested at a few attack edges. When e_A is small, attack edges tend to be far away from the vote collector. As shown in Figure 1, vote flow paths to honest voters A and B are congested at the link l_1 while paths to Sybil identities C and D are congested at both l_2 and attack edge l_3.

The adaptive vote flow computation uses three key ideas. First, the algorithm restricts the maximum number of votes collected on an object to a value C_{max}. As C_{max} is used to assign the overall capacity in the trust graph, a small C_{max} results in less capacity for the attacker. SumUp can adaptively adjust C_{max} to collect a large fraction of honest votes on any given object. When the number of honest voters is $O(n^{\alpha})$ where $\alpha < 1$, the expected number of bogus votes is limited to $1 + o(1)$ per attack edge (Section 5.4).

The second important aspect of SumUp relates to *capacity assignment*, i.e. how to assign capacities to each trust link in order to collect a large fraction of honest votes and only a few bogus ones? In SumUp, the vote collector distributes C_{max} *tickets* downstream in a breadth-first search manner within the trust network. The capacity assigned to a link is the number of tickets distributed along the link plus one. As Figure 2 illustrates, the ticket distribution process introduces a *vote envelope* around the vote collector s; beyond the envelope all links have capacity 1. The vote envelope contains C_{max} nodes that can be viewed as entry points. There is enough capacity within the envelope to collect C_{max} votes from entry points. On the other hand, an attack edge beyond the envelope can propagate at most 1 vote regardless of the number of Sybil

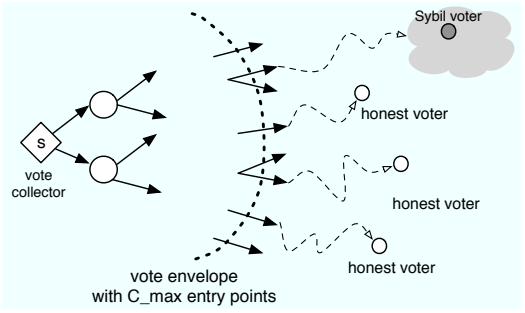

Figure 2: Through ticket distribution, SumUp creates a vote envelope around the collector. The capacities of links beyond the envelope are assigned to be one, limiting the attack capacity to be at most one per attack edge for adversaries outside this envelope. There is enough capacity within the envelope, such that nodes inside act like entry points for outside voters.

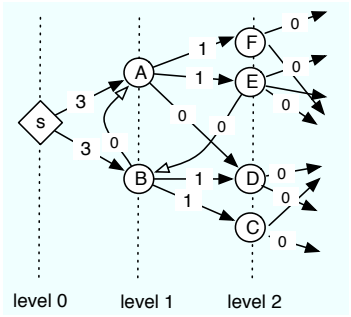

Figure 3: Each link shows the number of tickets distributed to that link from s (C_{max}=6). A node consumes one ticket and distributes the remaining evenly via its outgoing links to the next level. Tickets are not distributed to links pointing to the same level (B→A), or to a lower level (E→B). The capacity of each link is equal to one plus the number of tickets.

identities behind that edge. SumUp re-distributes tickets based on feedback to deal with attack edges within the envelope.

The final key idea in SumUp is to leverage user feedback to penalize attack edges that continuously propagate bogus votes. One cannot penalize individual identities since the attacker may always propagate bogus votes using new Sybil identities. Since an attack edge is always present in the path from the vote collector to a malicious voter [18], SumUp re-adjusts capacity assignment across links to reduce the capacity of penalized attack edges.

5 SumUp Design

In this section, we present the basic capacity assignment algorithm that achieves two of the three desired properties discussed in Section 3: (a) Collect a large fraction of votes from honest users; (b) Restrict the number of bogus votes to one per attack edge with high probability. Later in Section 6, we show how to adjust capacity based on feedback to deal with repeatedly misbehaved adversarial nodes.

We describe how link capacities are assigned given a particular C_{max} in Section 5.1 and present a fast algorithm to calculate approximate max-flow paths in Section 5.2. In Section 5.3, we introduce an additional optimization strategy that prunes links in the trust network so as to reduce the number of attack edges. We formally analyze the security properties of SumUp in Section 5.4 and show how to adaptively set C_{max} in Section 5.5.

5.1 Capacity assignment

The goal of capacity assignment is twofold. On the one hand, the assignment should allow the vote collector to gather a large fraction of honest votes. On the other hand, the assignment should minimize the attack capacity such that $C_A \approx e_A$.

As Figure 2 illustrates, the basic idea of capacity assignment is to construct a vote envelope around the vote collector with at least C_{max} entry points. The goal is to minimize the chances of including an attack edge in the envelope and to ensure that there is enough capacity within the envelope so that all vote flows from C_{max} entry points can reach the collector.

We achieve this goal using a *ticket distribution* mechanism which results in decreasing capacities for links with increasing distance from the vote collector. The distribution mechanism is best described using a propagation model where the vote collector is to spread C_{max} tickets across all links in the trust graph. Each ticket corresponds to a capacity value of 1. We associate each node with a level according to its shortest path distance from the vote collector, s. Node s is at level 0. Tickets are distributed to nodes one level at a time. If a node at level l has received t_{in} tickets from nodes at level $l-1$, the node consumes one ticket and re-distributes the remaining tickets evenly across all its outgoing links to nodes at level $l+1$, i.e. $t_{out} = t_{in} - 1$. The capacity value of each link is set to be one plus the number of tickets distributed on that link. Tickets are not distributed to links connecting nodes at the same level or from a higher to lower level. The set of nodes with positive incoming tickets fall within the vote envelope and thus represent the entry points.

Ticket distribution ensures that all C_{max} entry points have positive vote flows to the vote collector. Therefore, if there exists an edge-independent path connecting one of the entry points to an outside voter, the corresponding vote can be collected. We show in Section 5.4 that such a path exists with good probability. When C_{max} is much smaller than the number of honest nodes (n), the vote envelope is very small. Therefore, all attack edges reside outside the envelope, resulting in $C_A \approx e_A$ with high probability.

Figure 3 illustrates an example of the ticket distribution process. The vote collector (s) is to distribute C_{max}=6 tickets among all links. Each node collects tickets from its lower level neighbors, keeps one to itself and re-

distributes the rest evenly across all outgoing links to the next level. In Figure 3, s sends 3 tickets down each of its outgoing links. Since A has more outgoing links (3) than its remaining tickets (2), link A→D receives no tickets. Tickets are not distributed to links between nodes at the same level (B→A) or to links from a higher to lower level (E→B). The final number of tickets distributed on each link is shown in Figure 3. Except for immediate outgoing edges from the vote collector, the capacity value of each link is equal to the amount of tickets it receives plus one.

5.2 Approximate Max-flow calculation

Once capacity assignment is done, the task remains to calculate the set of max-flow paths from the vote collector to all voters on a given object. It is possible to use existing max-flow algorithms such as Ford-Fulkerson and Preflow push [4] to compute vote flows. Unfortunately, these existing algorithms require $O(E)$ running time to find each vote flow, where E is the number of edges in the graph. Since vote aggregation only aims to collect a large fraction of honest votes, it is not necessary to compute exact max-flow paths. In particular, we can exploit the structure of capacity assignment to compute a set of approximate vote flows in $O(\Delta)$ time, where Δ is the diameter of the graph. For expander-like networks, $\Delta = O(\log n)$. For practical social networks with a few million users, $\Delta \approx 20$.

Our approximation algorithm works incrementally by finding one vote flow for a voter at a time. Unlike the classic Ford-Fulkerson algorithm, our approximation performs a greedy search from the voter to the collector in $O(\Delta)$ time instead of a breadth-first-search from the collector which takes $O(E)$ running time. Starting at a voter, the greedy search strategy attempts to explore a node at a lower level if there exists an incoming link with positive capacity. Since it is not always possible to find such a candidate for exploration, the approximation algorithm allows a threshold (t) of non-greedy steps which explores nodes at the same or a higher level. Therefore, the number of nodes visited by the greedy search is bounded by $(\Delta + 2t)$. Greedy search works well in practice. For links within the vote envelope, there is more capacity for lower-level links and hence greedy search is more likely to find a non-zero capacity path by exploring lower-level nodes. For links outside the vote envelope, greedy search results in short paths to one of the vote entry points.

5.3 Optimization via link pruning

We introduce an optimization strategy that performs link pruning to reduce the number of attack edges, thereby reducing the attack capacity. Pruning is performed prior to link capacity assignment and its goal is to bound the in-degree of each node to a small value, d_{in_thres}. As a result, the number of attack edges is reduced if some adversarial nodes have more than d_{in_thres} incoming edges from honest nodes. We speculate that the more honest neighbors an adversarial node has, the easier for it to trick an honest node into trusting it. Therefore, the number of attack edges in the pruned network is likely to be smaller than those in the original network. On the other hand, pruning is unlikely to affect honest users since each honest node only attempts to cast one vote via one of its incoming links.

Since it is not possible to accurately discern honest identities from Sybil identities, we give all identities the chance to have their votes collected. In other words, pruning should never disconnect a node. The minimally connected network that satisfies this requirement is a tree rooted at the vote collector. A tree topology minimizes attack edges but is also overly restrictive for honest nodes because each node has exactly one path from the collector: if that path is saturated, a vote cannot be collected. A better tradeoff is to allow each node to have at most $d_{in\_thres} > 1$ incoming links in the pruned network so that honest nodes have a large set of diverse paths while limiting each adversarial node to only d_{in_thres} attack edges. We examine the specific parameter choice of d_{in_thres} in Section 7.

Pruning each node to have at most d_{in_thres} incoming links is done in several steps. First, we remove all links except those connecting nodes at a lower level (l) to neighbors at the next level ($l + 1$). Next, we remove a subset of incoming links at each node so that the remaining links do not exceed d_{in_thres}. In the third step, we add back links removed in step one for nodes with fewer than d_{in_thres} incoming links. Finally, we add one outgoing link back to nodes that have no outgoing links after step three, with priority given to links going to the next level. By preferentially preserving links from lower to higher levels, pruning does not interfere with SumUp's capacity assignment and flow computation.

5.4 Security Properties

This section provides a formal analysis of the security properties of SumUp assuming an expander graph. Various measurement studies have shown that social networks are indeed expander-like [13]. The link pruning optimization does not destroy a graph's expander property because it preserves the level of each node in the original graph.

Our analysis provides bounds on the expected attack capacity, C_A, and the expected fraction of votes collected if C_{max} honest users vote. The average-case analysis assumes that each attack edge is a random link in the graph. For personalized vote aggregation, the expectation is taken over all vote collectors which include all honest nodes. In the unfortunate but rare scenario where an adversarial node is close to the vote collector, we can use feedback to re-adjust link capacities (Section 6).

Theorem 5.1 *Given that the trust network G on n nodes is a bounded degree expander graph, the expected capacity per attack edge is* $\frac{E(C_A)}{e_A} = 1 + O(\frac{C_{max}}{n} \log C_{max})$

which is $1 + o(1)$ *if* $C_{max} = O(n^{\alpha})$ *for* $\alpha < 1$. *If* $e_A \cdot C_{max} \ll n$, *the capacity per attack edge is bounded by* 1 *with high probability.*

Proof Sketch Let L_i represent the number of nodes at level i with $L_0 = 1$. Let E_i be the number of edges pointing from level $i-1$ to level i. Notice that $E_i \geq L_i$. Let T_i be the number of tickets propagated from level $i-1$ to i with $T_0 = C_{max}$. The number of tickets at each level is reduced by the number of nodes at the previous level (i.e. $T_i = T_{i-1} - L_{i-1}$). Therefore, the number of levels with non-zero tickets is at most $O(log(C_{max}))$ as L_i grows exponentially in an expander graph. For a randomly placed attack edge, the probability of its being at level i is at most L_i/n. Therefore, the expected capacity of a random attack edge can be calculated as $1 + \sum_i (\frac{L_i}{n} \cdot \frac{T_i}{E_i}) < 1 + \sum_i (\frac{L_i}{n} \cdot \frac{C_{max}}{L_i}) = 1 + O(\frac{C_{max}}{n} \log C_{max})$. Therefore, if $C_{max} = O(n^{\alpha})$ for $\alpha < 1$, the expected attack capacity per attack edge is $1 + o(1)$.

Since the number of nodes within the vote envelope is at most C_{max}, the probability of a random attack edge being located outside the envelope is $1 - \frac{C_{max}}{n}$. Therefore, the probability that any of the e_A attack edges lies within the vote envelope is $1 - (1 - \frac{C_{max}}{n})^{e_A} < \frac{e_A \cdot C_{max}}{n}$. Hence, if $e_A \cdot C_{max} = n^{\alpha}$ where $\alpha < 1$, the attack capacity is bounded by 1 with high probability.

Theorem 5.1 is for expected capacity per attack edge. In the worse case when the vote collector is adjacent to some adversarial nodes, the attack capacity can be a significant fraction of C_{max}. Such rare worst case scenarios are addressed in Section 6.

Theorem 5.2 *Given that the trust network* G *on* n *nodes is a* d*-regular expander graph, the expected fraction of votes that can be collected out of* C_{max} *honest voters is* $\frac{d-\lambda_2}{d}(1 - \frac{C_{max}}{n})$ *where* λ_2 *is the second largest eigenvalue of the adjacency matrix of* G.

Proof Sketch SumUp creates a vote envelop consisting of C_{max} entry points via which votes are collected. To prove that there exists a large fraction of vote flows, we argue that the minimum cut of the graph between the set of C_{max} entry points and an arbitrary set of C_{max} honest voters is large.

Expanders are well-connected graphs. In particular, the Expander mixing lemma [19] states that for any set S and T in a d-regular expander graph, the expected number of edges between S and T is $(d - \lambda_2)|S| \cdot |T|/n$, where λ_2 is the second largest eigenvalue of the adjacency matrix of G. Let S be a set of nodes containing C_{max} entry points and T be a set of nodes containing C_{max} honest voters, thus $|S| + |T| = n$ and $|S| \geq C_{max}, |T| \geq C_{max}$. Therefore, the min-cut value between S and T is $= (d - \lambda_2)|S| \cdot |T|/n \geq (d - \lambda_2) \cdot C_{max}(n - C_{max})/n$. The number of vote flows between S and T is at least $1/d$ of the min-cut value because each vote flow only uses one of an honest voter's d incoming links. Therefore, the fraction of votes that can be collected is at least $(d - \lambda_2) \cdot C_{max}(n - C_{max})/(n \cdot d \cdot C_{max}) = \frac{d-\lambda_2}{d}(1 - \frac{C_{max}}{n})$. For well-connected graphs like expanders, λ_2 is well separated from d, so that a significant fraction of votes can be collected.

5.5 Setting C_{max} adaptively

When n_v honest users vote on an object, SumUp should ideally set C_{max} to be n_v in order to collect a large fraction of honest votes on that object. In practice, n_v/n is very small for any object, even a very popular one. Hence, $C_{max} = n_v \ll n$ and the expected capacity per attack edge is 1. We note that even if $n_v \approx n$, the attack capacity is still bounded by $O(\log n)$ per attack edge.

It is impossible to precisely calculate the number of honest votes (n_v). However, we can use the actual number of votes collected by SumUp as a lower bound estimate for n_v. Based on this intuition, SumUp adaptively sets C_{max} according to the number of votes collected for each object. The adaptation works as follows: For a given object, SumUp starts with a small initial value for C_{max}, e.g. $C_{max} = 100$. Subsequently, if the number of actual votes collected exceeds ρC_{max} where ρ is a constant less than 1, SumUp doubles the C_{max} in use and re-runs the capacity assignment and vote collection procedures. The doubling of C_{max} continues until the number of collected votes becomes less than ρC_{max}.

We show that this adaptive strategy is robust, i.e. the maximum value of the resulting C_{max} will not dramatically exceed n_v regardless of the number of bogus votes cast by adversarial nodes. Since adversarial nodes attempt to cast enough bogus votes to saturate attack capacity, the number of votes collected is at most $n_v + C_A$ where $C_A = e_A(1 + \frac{C_{max}}{n} \log C_{max})$. The doubling of C_{max} stops when the number of collected votes is less than ρC_{max}. Therefore, the maximum value of C_{max} that stops the adaptation is one that satisfies the following inequality:

$$n_v + e_A(1 + \frac{C_{max}}{n} \log C_{max}) < \rho C_{max}$$

Since $\log C_{max} \leq \log n$, the adaptation terminates with $C'_{max} = (n_v + e_A)/(\rho - \frac{\log n}{n})$. As $\rho \gg \frac{\log n}{n}$, we derive $C'_{max} = \frac{1}{\rho}(n_v + e_A)$. The adaptive strategy doubles C_{max} every iteration, hence it overshoots by at most a factor of two. Therefore, the resulting C_{max} found is $C_{max} = \frac{2}{\rho}(n_v + e_A)$. As we can see, the attacker can only affect the C_{max} found by an additive factor of e_A. Since e_A is small, the attacker has negligible influence on the C_{max} found.

The previous analysis is done for the expected case with random attack edges. Even in a worst case scenario where

some attack edges are very close to the vote collector, the adaptive strategy is still resilient against manipulation. In the worst case scenario, the attack capacity is proportional to C_{max}, i.e. $C_A = xC_{max}$. Since no vote aggregation scheme can defend against an attacker who controls a majority of immediate links from the vote collector, we are only interested in the case where $x < 0.5$. The adaptive strategy stops increasing C_{max} when $n_v + xC_{max} < \rho C_{max}$, thus resulting in $C_{max} \leq \frac{2n_v}{\rho - x}$. As we can see, ρ must be greater than x to prevent the attacker from causing SumUp to increase C_{max} to infinity. Therefore, we set $\rho = 0.5$ by default.

6 Leveraging user feedback

The basic design presented in Section 5 does not address the worst case scenario where C_A could be much higher than e_A. Furthermore, the basic design only bounds the number of bogus votes collected on a single object. As a result, adversaries can still cast up to e_A bogus votes on *every* object in the system. In this section, we utilize feedback to address both problems.

SumUp maintains a penalty value for each link and uses the penalty in two ways. First, we adjust each link's capacity assignment so that links with higher penalties have lower capacities. This helps reduce C_A when some attack edges happen to be close to the vote collector. Second, we eliminate links whose penalties have exceeded a certain threshold. Therefore, if adversaries continuously misbehave, the attack capacity will drop below e_A over time. We describe how SumUp calculates and uses penalty in the rest of the section.

6.1 Incorporating negative feedback

The vote collector can choose to associate negative feedback with voters if he believes their votes are malicious. Feedback may be performed for a very small set of objects-for example, when the collector finds out that an object is a bogus file or a virus.

SumUp keeps track of a penalty value, p_i, for each link i in the trust network. For each voter receiving negative feedback, SumUp increments the penalty values for all links along the path to that voter. Specifically, if the link being penalized has capacity c_i, SumUp increments the link's penalty by $1/c_i$. Scaling the increment by c_i is intuitive; links with high capacities are close to the vote collector and hence are more likely to propagate some bogus votes even if they are honest links. Therefore, SumUp imposes a lesser penalty on high capacity links.

It is necessary to penalize *all* links along the path instead of just the immediate link to the voter because that voter might be a Sybil identity created by some other attacker along the path. Punishing a link to a Sybil identity is useless as adversaries can easily create more such links. This way of incorporating negative feedback is inspired by Ostra [18]. Unlike Ostra, SumUp uses a customized flow network per vote collector and only allows the collector to incorporate feedback for its associated network in order to ensure that feedback is always trustworthy.

6.2 Capacity adjustment

The capacity assignment in Section 5.1 lets each node distribute incoming tickets evenly across all outgoing links. In the absence of feedback, it is reasonable to assume that all outgoing links are equally trustworthy and hence to assign them the same number of tickets. When negative feedback is available, a node should distribute fewer tickets to outgoing links with higher penalty values. Such adjustment is particularly useful in circumstances where adversaries are close to the vote collector and hence might receive a large number of tickets.

The goal of capacity adjustment is to compute a weight, $w(p_i)$, as a function of the link's penalty. The number of tickets a node distributes to its outgoing link i is proportional to the link's weight, i.e. $t_i = t_{out} * w(p_i) / \sum_{\forall i \in nbrs} w(p_i)$. The question then becomes how to compute $w(p_i)$. Clearly, a link with a high penalty value should have a smaller weight, i.e. $w(p_i) < w(p_j)$ if $p_i > p_i$. Another desirable property is that if the penalties on two links increase by the same amount, the ratio of their weights remains unchanged. In other words, the weight function should satisfy: $\forall p', p_i, p_j, \quad \frac{w(p_i)}{w(p_j)} = \frac{w(p_i+p')}{w(p_j+p')}$. This requirement matches our intuition that if two links have accumulated the same amount of additional penalties over a period of time, the relative capacities between them should remain the same. Since the exponential function satisfies both requirements, we use $w(p_i) = 0.2^{p_i}$ by default.

6.3 Eliminating links using feedback

Capacity adjustment cannot reduce the attack capacity to below e_A since each link is assigned a minimum capacity value of one. To further reduce e_A, we eliminate those links that received high amounts of negative feedback.

We use a heuristic for link elimination: we remove a link if its penalty exceeds a threshold value. We use a default threshold of five. Since we already prune the trust network (Section 5.3) before performing capacity assignment, we add back a previously pruned link if one exists after eliminating an incoming link. The reason why link elimination is useful can be explained intuitively: if adversaries continuously cast bogus votes on different objects over time, all attack edges will be eliminated eventually. On the other hand, although an honest user might have one of its incoming links eliminated because of a downstream attacker casting bad votes, he is unlikely to experience another elimination due to the same attacker since the attack edge connecting him to that attacker has also been eliminated. Despite this intuitive argument, there always exist pathological scenarios where link elimination affects some honest users, leaving them with no voting

Network	Nodes $\times 1000$	Edges $\times 1000$	Degree 50%(90%)	Directed?
YouTube [18]	446	3,458	2 (12)	No
Flickr [17]	1,530	21,399	1 (15)	Yes
Synthetic [24]	3000	24,248	6 (15)	No

Table 1: Statistics of the social network traces or synthetic model used for evaluating SumUp. All statistics are for the strongly connected component (SCC).

power. To address such potential drawbacks, we re-enact eliminated links at a slow rate over time. We evaluate the effect of link elimination in Section 7.

7 Evaluation

In this section, we demonstrate SumUp's security property using real-world social networks and voting traces. Our key results are:

1. For all networks under evaluation, SumUp bounds the average number of bogus votes collected to be no more than e_A while being able to collect >90% of honest votes when less than 1% of honest users vote.
2. By incorporating feedback from the vote collector, SumUp dramatically cuts down the attack capacity for adversaries that continuously cast bogus votes.
3. We apply SumUp to the voting trace and social network of Digg [1], a news aggregation site that uses votes to rank user-submitted news articles. SumUp has detected hundreds of suspicious articles that have been marked as "popular" by Digg. Based on manual sampling, we believe at least 50% of suspicious articles found by SumUp exhibit strong evidence of Sybil attacks.

7.1 Experimental Setup

For the evaluation, we use a number of network datasets from different online social networking sites [17] as well as a synthetic social network [24] as the underlying trust network. SumUp works for different types of trust networks as long as an attacker cannot obtain many attack edges easily in those networks. Table 1 gives the statistics of various datasets. For undirected networks, we treat each link as a pair of directed links. Unless explicitly mentioned, we use the YouTube network by default.

To evaluate the Sybil-resilience of SumUp, we inject $e_A = 100$ attack edges by adding 10 adversarial nodes each with links from 10 random honest nodes in the network. The attacker always casts the maximum bogus votes to saturate his capacity. Each experimental run involves a randomly chosen vote collector and a subset of nodes which serve as honest voters. SumUp adaptively adjusts C_{max} using an initial value of 100 and $\rho = 0.5$. By default, the threshold of allowed non-greedy steps is 20. We plot the average statistic across five experimental runs in all graphs. In Section 7.6, we apply SumUp on the real world voting trace of Digg to examine how SumUp can be used to resist Sybil attacks in the wild.

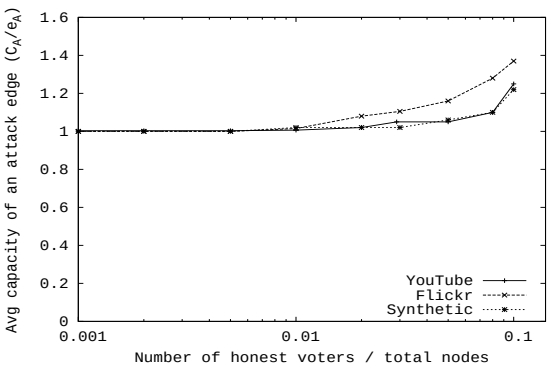

Figure 4: The average capacity per attack edge as a function of the fraction of honest nodes that vote. The average capacity per attack edge remains close to 1, even if $1/10$ of honest nodes vote.

7.2 Sybil-resilience of the basic design

The main goal of SumUp is to limit attack capacity while allowing honest users to vote. Figure 4 shows that the average attack capacity per attack edge remains close to 1 even when the number of honest voters approaches 10%. Furthermore, as shown in Figure 5, SumUp manages to collect more than 90% of all honest votes in all networks. Link pruning is disabled in these experiments. The three networks under evaluation have very different sizes and degree distributions (see Table 1). The fact that all three networks exhibit similar performance suggests that SumUp is robust against the topological details. Since SumUp adaptively sets C_{max} in these experiments, the results also confirm that adaptation works well in finding a C_{max} that can collect most of the honest votes without significantly increasing attack capacity. We point out that the results in Figure 4 correspond to a random vote collector. For an unlucky vote collector close to an attack edge, he may experience a much larger than average attack capacity. In personalized vote collection, there are few unlucky collectors. These unlucky vote collectors need to use their own feedback on bogus votes to reduce attack capacity.

Benefits of pruning: The link pruning optimization, introduced in Section 5.3, further reduces the attack capacity by capping the number of attack edges an adversarial node can have. As Figure 6 shows, pruning does not affect the fraction of honest votes collected if the threshold d_{in_thres} is greater than 3. Figure 6 represents data from the YouTube network and the results for other networks are similar. SumUp uses the default threshold (d_{in_thres}) of 3. Figure 7 shows that the average attack capacity is greatly reduced when adversarial nodes have more than 3 attack edges. Since pruning attempts to restrict each node to at most 3 incoming links, additional attack edges are excluded from vote flow computation.

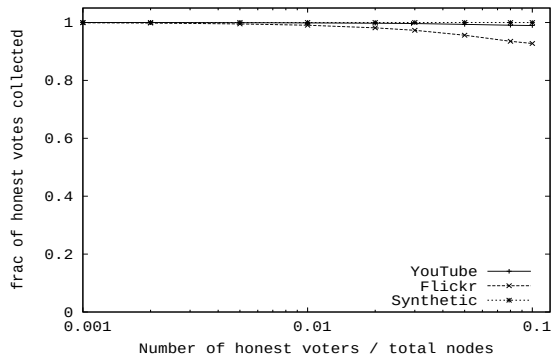

Figure 5: The fraction of votes collected as a function of fraction of honest nodes that vote. SumUp collects more than 80% votes, even $1/10$ honest nodes vote.

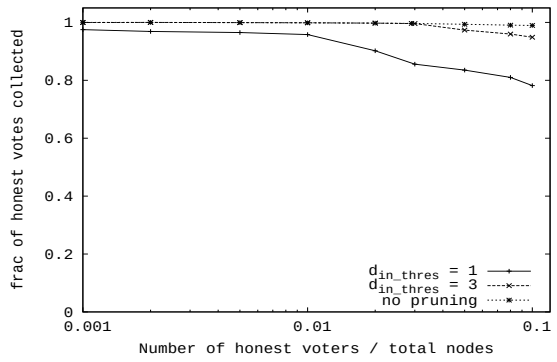

Figure 6: The fraction of votes collected for different d_{in_thres} (YouTube graph). More than 90% votes are collected when $d_{in\_thres} = 3$.

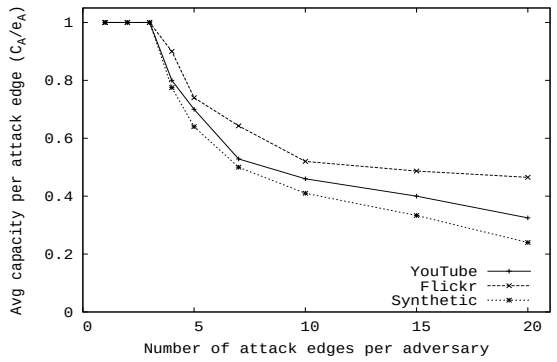

Figure 7: Average attack capacity per attack edge decreases as the number of attack edges per adversary increases.

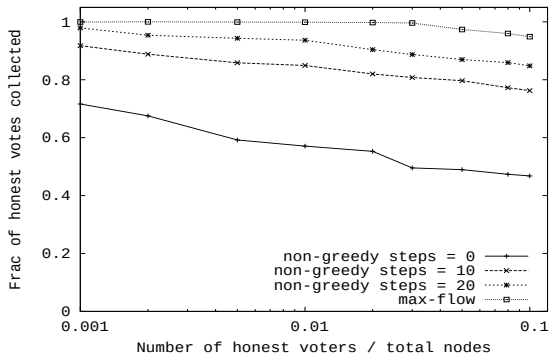

Figure 8: The fraction of votes collected for different threshold for non-greedy steps. More than 70% votes are collected even with a small threshold (10) for non-greedy steps.

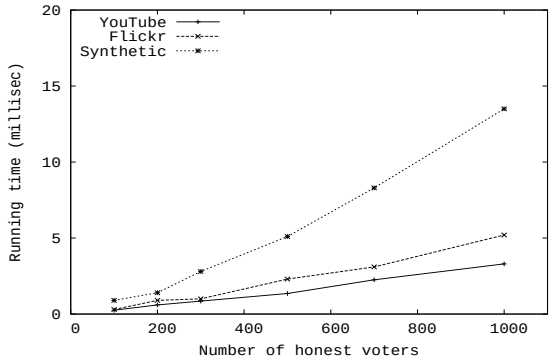

Figure 9: The running time of one vote collector gathering up to 1000 votes. The Ford-Fulkerson max-flow algorithm takes 50 seconds to collect 1000 votes for the YouTube graph.

7.3 Effectiveness of greedy search

SumUp uses a fast greedy algorithm to calculate approximate max vote flows to voters. Greedy search enables SumUp to collect a majority of votes while using a small threshold (t) of non-greedy steps. Figure 8 shows the fraction of honest votes collected for the pruned YouTube graph. As we can see, with a small threshold of 20, the fraction of votes collected is more than 80%. Even when disallowing non-greedy steps completely, SumUp manages to collect $> 40\%$ of votes.

Figure 9 shows the running time of greedy-search for different networks. The experiments are performed on a single machine with an AMD Opteron 2.5GHz CPU and 8GB memory. SumUp takes around 5ms to collect 1000 votes from a single vote collector on YouTube and Flickr. The synthetic network incurs more running time as its links are more congested than those in YouTube and Flickr. The average non-greedy steps taken in the synthetic network is 6.5 as opposed to 0.8 for the YouTube graph. Greedy-search dramatically reduces the flow computation time. As a comparison, the Ford-Fulkerson max-flow algorithm requires 50 seconds to collect 1000 votes

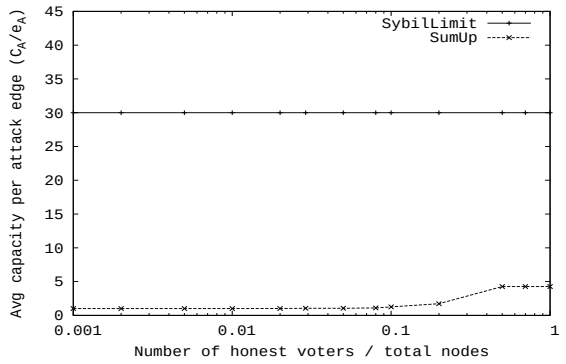

Figure 10: Average attack capacity per attack edge as a function of voters. SumUp is better than SybilLimit in the average case.

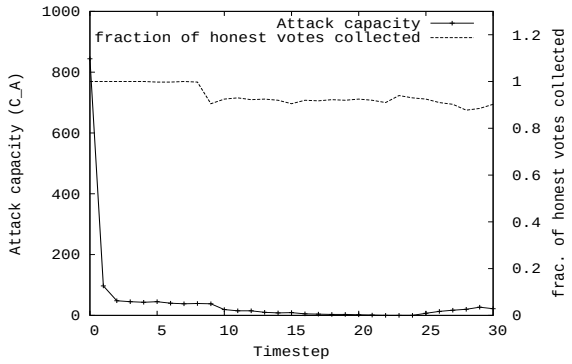

Figure 11: The change in attack capacity as adversaries continuously cast bogus votes (YouTube graph). Capacity adjustment and link elimination dramatically reduce C_A while still allowing SumUp to collect more than 80% of the honest votes.

for the YouTube graph.

7.4 Comparison with SybilLimit

SybilLimit is a node admission protocol that leverages the trust network to allow an honest node to accept other honest nodes with high probability. It bounds the number of Sybil nodes accepted to be $O(\log n)$. We can apply SybilLimit for vote aggregation by letting each vote collector compute a fixed set of accepted users based on the trust network. Subsequently, a vote is collected if and only if it comes from one of the accepted users. In contrast, SumUp does not calculate a fixed set of allowed users; rather, it dynamically determines the set of voters that count toward each object. Such dynamic calculation allows SumUp to settle on a small C_{max} while still collecting most of the honest votes. A small C_{max} allows SumUp to bound attack capacity by e_A.

Figure 10 compares the average attack capacity in SumUp to that of SybilLimit for the un-pruned YouTube network. The attack capacity in SybilLimit refers to the number of Sybil nodes that are accepted by the vote collector. Since SybilLimit aims to accept nodes instead of votes, its attack capacity remains $O(\log n)$ regardless of the number of actual honest voters. Our implementation of SybilLimit uses the optimal set of parameters ($w = 15$, $r = 3000$) we determined manually. As Figure 10 shows, while SybilLimit allows 30 bogus votes per attack edge, SumUp results in approximately 1 vote per attack edge when the fraction of honest voters is less than 10%. When all nodes vote, SumUp leads to much lower attack capacity than SybilLimit even though both have the same $O(\log n)$ asymptotic bound per attack edge. This is due to two reasons. First, SumUp's bound of $1 + \log n$ in Theorem 5.1 is a loose upper bound of the actual average capacity. Second, since links pointing to lower-level nodes are not eligible for ticket distribution, many incoming links of an adversarial nodes have zero tickets and thus are assigned capacity of one.

7.5 Benefits of incorporating feedback

We evaluate the benefits of capacity adjustment and link elimination when the vote collector provides feedback on the bogus votes collected. Figure 11 corresponds to the worst case scenario where one of the vote collector's four outgoing links is an attack edge. At every time step, there are 400 random honest users voting on an object and the attacker also votes with its maximum capacity. When collecting votes on the first object at time step 1, adaption results in $C_{max} = \frac{2n_v}{\rho - x} = 3200$ because $n_v = 400, \rho = 0.5, x = 1/4$. Therefore, the attacker manages to cast $\frac{1}{4}C_{max} = 800$ votes and outvote honest users. After incorporating the vote collector's feedback after the first time step, the adjacent attack edge incurs a penalty of 1 which results in drastically reduced C_A (97). If the vote collector continues to provide feedback on malicious votes, 90% of attack edges are eliminated after only 12 time steps. After another 10 time steps, all attack edges are eliminated, reducing C_A to zero. However, because of our decision to slowly add back eliminated links, the attack capacity doesn't remains at zero forever. Figure 11 also shows that link elimination has little effects on honest nodes as the fraction of honest votes collected always remains above 80%.

7.6 Defending Digg against Sybil attacks

In this section, we ask the following questions: Is there evidence of Sybil attacks in real world content voting systems? Can SumUp successfully limit bogus votes from Sybil identities? We apply SumUp to the voting trace and social network crawled from Digg to show the real world benefits of SumUp.

Digg [1] is a popular news aggregation site where any registered user can submit an article for others to vote on. A positive vote on an article is called a *digg*. A negative vote is called a *bury*. Digg marks a subset of submitted articles as "popular" articles and displays them on its front page. In subsequent discussions, we use the terms *pop-*

Number of Nodes	3,002,907
Number of Edges	5,063,244
Number of Nodes in SCC	466,326
Number of Edges in SCC	4,908,958
Out degree avg(50%, 90%)	10(1, 9)
In degree avg(50%, 90%)	10(2, 11)
Number of submitted (popular) articles 2004/12/01-2008/09/21	6,494,987 (137,480)
Diggs on all articles avg(50%, 90%)	24(2, 15)
Diggs on popular articles avg(50%, 90%)	862(650, 1810)
Hours since submission before a popular article is marked as popular. avg (50,%,90%)	16(13, 23)
Number of submitted (popular) articles with *bury* data available 2008/08/13-2008/09/15	38,033 (5,794)

Table 2: Basic statistics of the crawled Digg dataset. The strongly connected component (SCC) of Digg consists of 466,326 nodes.

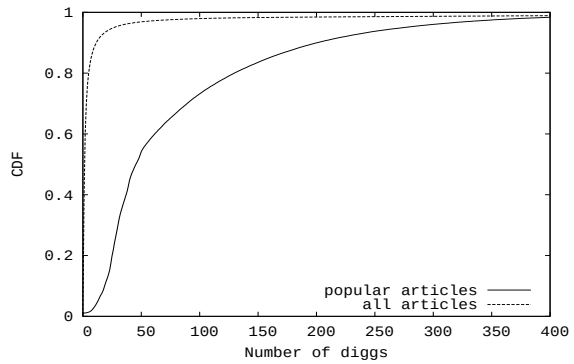

Figure 12: Distribution of diggs for all popular articles before being marked as popular and for all articles within 24 hours after submission.

ular or *popularity* only to refer to the popularity status of an article as marked by Digg. A Digg user can create a "follow" link to another user if he wants to browse all articles submitted by that user. We have crawled Digg to obtain the voting trace on all submitted articles since Digg's launch (2004/12/01-2008/09/21) as well as the complete "follow" network between users. Unfortunately, unlike diggs, bury data is only available as a live stream. Furthermore, Digg does not reveal the user identity that cast a bury, preventing us from evaluating SumUp's feedback mechanism. We have been streaming bury data since 2008/08/13. Table 2 shows the basic statistics of the Digg "follow" network and the two voting traces, one with bury data and one without. Although the strongly connected component (SCC) consists of only 15% of total nodes, 88% of votes come from nodes in the SCC.

There is enormous incentive for an attacker to get a submitted article marked as popular, thus promoting it to the front page of Digg which has several million page views per day. Our goal is to apply SumUp on the voting trace to reduce the number of successful attacks on the popularity marking mechanism of Digg. Unfortunately, unlike experiments done in Section 7.2 and Section 7.5, there is no ground truth about which Digg users are adversaries. Instead, we have to use SumUp itself to find evidence of attacks and rely on manual sampling and other types of data to cross check the correctness of results.

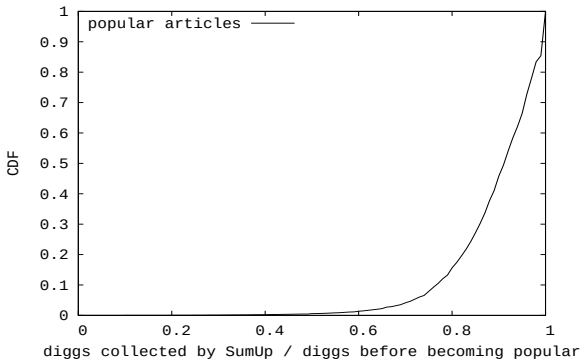

Figure 13: The distribution of the fraction of diggs collected by SumUp over all diggs before an article is marked as popular.

Digg's popularity ranking algorithm is intentionally not revealed to the public in order to mitigate gaming of the system. Nevertheless, we speculate that the number of diggs is a top contributor to an article's popularity status. Figure 12 shows the distribution of the number of diggs an article received before it was marked as popular. Since more than 90% of popular articles are marked as such within 24 hours after submission, we also plot the number of diggs received within 24 hours of submission for all articles. The large difference between the two distributions indicates that the number of diggs plays an important role in determining an article's popularity status.

Instead of simply adding up the actual number of diggs, what if Digg uses SumUp to collect all votes on an article? We use the identity of Kevin Rose, the founder of Digg, as the vote collector to aggregate all diggs on an article before it is marked as popular. Figure 13 shows the distribution of the fraction of votes collected by SumUp over all diggs before an article is marked as popular. Our previous evaluation on various network topologies suggests that SumUp should be able to collect at least 90% of all votes. However, in Figure 13, there are a fair number of popular articles with much fewer than the expected fraction of diggs collected. For example, SumUp only manages to collect less than 50% of votes for 0.5% of popular articles. We hypothesize that the reason for collecting fewer than the expected votes is due to real world Sybil attacks.

Since there is no ground truth data to verify whether

Threshold of the fraction of collected diggs	20%	30%	40%	50%
# of suspicious articles	41	131	300	800
Advertisement	5	4	2	1
Phishing	1	0	0	0
Obscure political articles	2	2	0	0
Many newly registered voters	11	7	8	10
Fewer than 50 total diggs	1	3	6	4
No obvious attack	10	14	14	15

Table 3: Manual classification of 30 randomly sampled suspicious articles. We use different thresholds of the fraction of collected diggs for marking suspicious articles. An article is labeled as having many new voters if $> 30\%$ of its votes are from users who registered on the same day as the article's submission date.

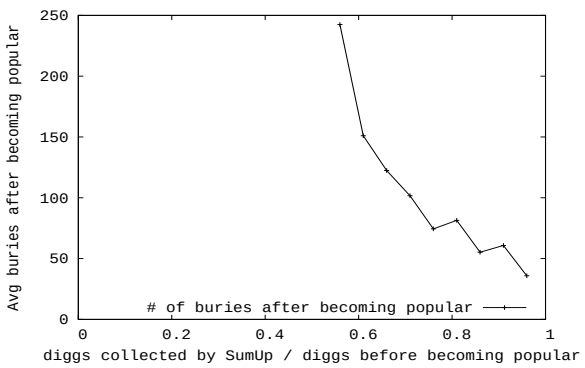

Figure 14: The average number of buries an article received *after* it was marked as popular as a function of the fraction of diggs collected by SumUp *before* it is marked as popular. The Figure covers $5,794$ popular articles with bury data available.

few collected diggs are indeed the result of attacks, we resort to manual inspection. We classify a popular article as suspicious if its fraction of diggs collected is less than a given threshold. Table 3 shows the result of manually inspecting 30 random articles out of all suspicious articles. The random samples for different thresholds are chosen independently. There are a number of obvious bogus articles such as advertisements, phishing articles and obscure political opinions. Of the remaining, we find many of them have an unusually large fraction (>30%) of new voters who registered on the same day as the article's submission time. Some articles also have very few total diggs since becoming popular, a rare event since an article typically receives hundreds of votes after being shown on the front page of Digg. We find no obvious evidence of attack for roughly half of the sampled articles. Interviews with Digg attackers [10] reveal that, although there is a fair amount of attack activities on Digg, attackers do not usually promote obviously bogus material. This is likely due to Digg being a highly monitored system with fewer than a hundred articles becoming popular every day. Instead, attackers try to help paid customers promote normal or even good content or to boost their profiles within the Digg community.

As further evidence that a lower than expected fraction of collected diggs signals a possible attack, we examine Digg's *bury* data for articles submitted after 2008/08/13, of which 5794 are marked as popular. Figure 14 plots the correlation between the average number of bury votes on an article *after* it became popular vs. the fraction of the diggs SumUp collected before it was marked as popular. As Figure 14 reveals, the higher the fraction of diggs collected by SumUp, the fewer bury votes an article received after being marked as popular. Assuming most bury votes come from honest users that genuinely dislike the article, a large number of bury votes is a good indicator that the article is of dubious quality.

What are the voting patterns for suspicious articles? Since 88% diggs come from nodes within the SCC, we expect only 12% of diggs to originate from the rest of the network, which mostly consists of nodes with no incoming follow links. For most suspicious articles, the reason that SumUp collecting fewer than expected diggs is due to an unusually large fraction of votes coming from outside the SCC component. Since Digg's popularity marking algorithm is not known, attackers might not bother to connect their Sybil identities to the SCC or to each other. Interestingly, we found 5 suspicious articles with sophisticated voting patterns where one voter is linked to many identities (~ 30) that also vote on the same article. We believe the many identities behind that single voter are likely Sybil identities because those identities were all created on the same day as the article's submission. Additionally, those identities all have similar usernames.

8 SumUp in a Decentralized Setting

Even though SumUp is presented in a centralized setup such as a content-hosting Web site, it can also be implemented in a distributed fashion in order to rank objects in peer-to-peer systems. We outline one such distributed design for SumUp. In the peer-to-peer environment, each node and its corresponding user is identified by a self-generated public key. A pair of users create a trust link relationship between them by signing the trust statement with their private keys. Nodes gossip with each other or perform a crawl of the network to obtain a complete trust network between any pair of public keys. This is different from Ostra [18] and SybilLimit [26] which address the harder problem of decentralized routing where each user only knows about a small neighborhood around himself in the trust graph. In the peer-to-peer setup, each user naturally acts as his own vote collector to aggregate votes and compute a personalized ranking of objects. To obtain all votes on an object, a node can either perform flooding (like in Credence [25]) or retrieve votes stored in a dis-

tributed hash table. In the latter case, it is important that the DHT itself be resilient against Sybil attacks. Recent work on Sybil-resilient DHTs [5, 14] addresses this challenge.

9 Conclusion

This paper presented SumUp, a content voting system that leverages the trust network among users to defend against Sybil attacks. By using the technique of adaptive vote flow aggregation, SumUp aggregates a collection of votes with strong security guarantees: with high probability, the number of bogus votes collected is bounded by the number of attack edges while the number of honest votes collected is high. We demonstrate the real-world benefits of SumUp by evaluating it on the voting trace of Digg: SumUp detected many suspicious articles marked as "popular" by Digg. We have found strong evidence of Sybil attacks on many of these suspicious articles.

Acknowledgments

We thank Alan Mislove and Krishna Gummadi for making their social network traces publicly available. We are grateful to Krishna Gummadi (our shepard), Haifeng Yu, Aditya Dhananjay, Michael Paik, Eric Hielscher, and the anonymous reviewers whose comments have helped us improve the paper. This work is supported by NSF CAREER Award CNS-0747052.

References

[1] Digg. http://www.digg.com.

[2] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. In *WWW* (1998).

[3] CHENG, A., AND FRIEDMAN, E. Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems* (New York, NY, USA, 2005), ACM, pp. 128–132.

[4] CORMEN, T., LEISERSON, C., AND RIVEST, R. *Introduction to Algorithms*. The MIT Press, 1985.

[5] DANEZIS, G., LESNIEWSKI-LAAS, C., KAASHOEK, M. F., AND ANDERSON, R. Sybil-resistant dht routing. In *European Symposium On Research In Computer Security* (2008).

[6] DOUCEUR, J. The sybil attack. In *1st International Workshop on Peer-to-Peer Systems* (2002).

[7] FENG, Q., AND DAI, Y. Lip: A lifetime and popularity based ranking approach to filter out fake files in p2p file sharing systems. In *IPTPS* (2007).

[8] GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. Propagation of trust and distrust. In *WWW* (2004).

[9] HSIEH, H. Doonesbury online poll hacked in favor of MIT. In *MIT Tech* (2006).

[10] INVESPBLOG. An interview with digg top user, 2008. `http://www.invesp.com/blog/social-media/an-interview-with-digg-top-user.html`.

[11] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 640–651.

[12] KLEINBERG, J. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms* (1998).

[13] LESKOVEC, J., LANG, K., DASGUPTA, A., AND MAHONEY, M. W. Statistical properties of community structure in large social and information networks. In *7th international conference on WWW* (2008).

[14] LESNIEWSKI-LAAS, C. A sybil-proof one-hop dht. In *1st Workshop on Social Network Systems* (2008).

[15] LEVIEN, R., AND AIKEN, A. Attack-resistant trust metrics for public key certification. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998* (Berkeley, CA, USA, 1998), USENIX Association, pp. 18–18.

[16] LIANG, J., KUMAR, R., XI, Y., AND ROSS, K. Pollution in p2p file sharing systems. In *IEEE Infocom* (2005).

[17] MISLOVE, A., MARCON, M., GUMMADI, K., DRUSCHEL, P., AND BHATTACHARJEE, S. Measurement and analysis of online social networks. In *7th Usenix/ACM SIGCOMM Internet Measurement Conference (IMC)* (2007).

[18] MISLOVE, A., POST, A., DRUSCHEL, P., AND GUMMADI, K. P. Ostra: Leveraging trust to thwart unwanted communication. In *NSDI'08: Proceedings of the 5th conference on 5th Symposium on Networked Systems Design & Implementation* (2008).

[19] MOTWANI, R., AND RAGHAVAN, P. *Randomized Algorithms*. Cambridge University Press, 1995.

[20] NG, A., ZHENG, A., AND JORDAN, M. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence (IJCAI)* (2001).

[21] RICHARDSON, M., AGRAWAL, R., AND DOMINGOS, P. Trust management for the semantic web. In *Proceedings of the 2nd Semantic Web Conference* (2003).

[22] RILEY, D. Techcrunch: Stat gaming services come to youtube, 2007. `http://www.techcrunch.com/2007/08/23/myspace-style-profile-gaming-comes-to-youtube`.

[23] SOVRAN, Y., LIBONATI, A., AND LI, J. Pass it on: Social networks stymie censors. In *Proc. of the 7th International Workshop on Peer-to-Peer Systems (IPTPS)* (Feb 2008).

[24] TOIVONEN, R., ONNELA, J.-P., SARAMÄKI, J., HYVÖNEN, J., AND KASKI, K. A model for social networks. *Physica A Statistical Mechanics and its Applications 371* (2006), 851–860.

[25] WALSH, K., AND SIRER, E. G. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation* (Berkeley, CA, USA, 2006), USENIX Association, pp. 1–1.

[26] YU, H., GIBBONS, P., KAMINSKY, M., AND XIAO, F. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy* (2008).

[27] YU, H., KAMINSKY, M., GIBBONS, P. B., AND FLAXMAN, A. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2006), ACM, pp. 267–278.

[28] YU, H., SHI, C., KAMINSKY, M., GIBBONS, P. B., AND XIAO, F. Dsybil: Optimal sybil-resistance for recommendation systems. In *IEEE Security and Privacy* (2009).

[29] ZHANG, H., GOEL, A., GOVINDAN, R., AND MASON, K. Making eigenvector-based reputation systems robust to collusions. In *Proc. of the Third Workshop on Algorithms and Models for the Web Graph* (2004).

[30] ZIEGLER, C.-N., AND LAUSEN, G. Propagation models for trust and distrust in social networks. *Information Systems Frontiers 7*, 4-5 (2005), 337–358.