Fighting Censorship with Algorithms

Mohammad Mahdian

Yahoo! Research, Santa Clara, CA, USA.
mahdian@alum.mit.edu
WWW home page: http://www.mahdian.org

Abstract. In countries such as China or Iran where Internet censorship is prevalent, users usually rely on proxies or anonymizers to freely access the web. The obvious difficulty with this approach is that once the address of a proxy or an anonymizer is announced for use to the public, the authorities can easily filter all traffic to that address. This poses a challenge as to how proxy addresses can be announced to users without leaking too much information to the censorship authorities. In this paper, we formulate this question as an interesting algorithmic problem. We study this problem in a static and a dynamic model, and give almost tight bounds on the number of proxy servers required to give access to n people k of whom are adversaries. We will also discuss how trust networks can be used in this context.

1 Intrduction

Today, Internet is playing an ever-increasing role in social and political movements around the globe. Activists connect and organize online and inform ordinary citizen (and the rest of the world) of the news of arrests and crackdowns and other news that the political powers do not want to be spread. In particular, Web 2.0 has broken the media monopoly and has given voice to dissidents and citizen journalists with no access to traditional media outlets. The role that Twitter, Facebook, YouTube, CNN's iReport and many other websites and blogs have played in the recent events in Iran is a great example of this [9, 10].

Threatened by this paradigm, repressive regimes have tried hard to control and monitor their citizen's access to the web. Sophisticated filtering and surveillance technologies are developed or purchased by governments such as China or Iran to disallow access to certain blacklisted websites (See Figure 1) or monitor the activities of users [19]. Blacklisted websites include news websites such as BBC or CNN, Web 2.0 websites, many blogs, and even Wikipedia. Internet censorship activities are documented in great detail by organizations such as Reporters Without Borders [18] or the OpenNet Initiative [16, 5].

At the same time, users in such countries and supporters of free online speech have looked for *circumvention technologies* to get around Internet censorship. These range from secure peer-to-peer networks like Freenet [1, 2] to anonymous traffic routing softwares like The Onion Router (Tor) [6] to web proxy systems such as Psiphon [17]. These tools sometimes use cryptographic protocols to provide security, but to elude detection and filtering, the common way is to route

Fig. 1. A typical Internet browsing session in Iran involves multiple encounters with pages like the above. The text in Persian reads: "According to the laws of the Islamic Republic of Iran and directives from the judiciary access to this website is forbidden."

traffic through intermediaries not known to the censorship authorities. For example, when a user of Psiphon requests a webpage, this request is routed through a proxy. The proxy is a computer outside the censored country, and therefore can access the requested webpage and send it back to the user.[1] This can be any computer on the net whose owner is willing to contribute to the anti-censorship network by installing the Psiphon software. Usually, this is done by people outside the censored country who have friends or relatives inside the censored country. They simply install the software on their machines (with full-time Internet connectivity), and send an invitation (which is basically a link through which the recipient can access the Internet) to their friends and family.[2]

This model works well for users who have friends outside the censored territory. The challenge is to provide censorship-free Internet access to those who have no direct connection to the outside world. In the case of Iran, sometimes US-sponsored radio stations such as Voice of America or Radio Farda broadcast URLs of anonymizers on their program. The obvious problem with this approach

[1] This is a simplification of how Psiphon operates. In reality (starting from version 2.0), Psiphon is a 2-hop proxy system, where the first hop (the in-proxy) can be any computer outside the censored country that runs the Psiphon software and simply forwards the requests, and the second hop (the managed Psiphon server) is a dedicated computer that serves and manages Psiphon requests. For the purpose of our discussions, the simplified view is enough.

[2] Freenet's "darknet" mode operates similarly.

is that the censorship authorities also listen to these radio stations and quickly add the announced URL to their blacklist.

This motivates the main problem studied in this paper: how can a set of proxies be distributed among n users, k of whom adversaries (agents of censorship), in such a way that all legitimate users can have access to a proxy that is not filtered. We give a more precise definition of our model in the next section, and define a static (one-shot) version of this problem, as well as a (more realistic) dynamic problem. As we observe in Section 3, the static problem is equivalent to a previously studied network design problem. The main contribution of this paper is our solution for the dynamic model, which will be presented in Section 4. In Section 5, we discuss the role of trust networks in building a user base, and how this affects the algorithmic problem studied in this paper. We conclude with a number of open problems and conjectures.

Related work. To the best of our knowledge, this is the first systematic study of methods for proxy distribution. As we observe in Section 3, the static version of this problem is essentially equivalent to finding union-free families of sets [13]. In coding theory, these objects are known as *superimposed codes* [12, 8], and have a wide range of applications (see, for example, [3, 4, 11]). The static case is also closely related to the combinatorial group testing literature [7]. The dynamic problem, however, does not fit within the group testing framework, as in our problem the adversary can strategically delay compromising a proxy.

2 The model

Consider a population of n users, k of whom are adversaries, and the remaining $n-k$ are legitimate users. We do not know the identities of the adversaries, but throughout the paper we assume that $k \ll n$. We have a set of at most m *keys* (representing proxies or anonymizers) that we can distribute among these users in any way we want. If a key is given to an adversarial user, she can *compromise* the key, meaning that from that point on, the key will be unusable. Our goal is to distribute the keys in such a way that at the end, every legitimate user has at least one uncompromised key.

This problem can be studied in a static (one shot) model, or in a dynamic (adaptive) model. The static model is a one-round problem: we give each user a set of keys at once, and then the adversarial users compromise the keys they have received. In the dynamic model, there are multiple rounds. In each round we distribute a number of keys to users who have no uncompromised key. The next round starts when one or more of the keys given to an adversarial user is compromised. Note that in this model the adversarial users do not have to compromise a key as soon as they see them; instead, they can strategically decide when to compromise a key. Also, observe that in this model we can assume without loss of generality that in any round each user who has no uncompromised key receives only one key (i.e., there is no point in giving more than one uncompromised key to a user at any time).

In both of the above models, if the number m of available keys is at least n, the problem is trivially solvable: simply give each user her personal key. Our objective is to find the smallest value of m for which the problem is solvable. As we will see in the next sections, it is indeed possible to solve the problem with only a sublogarithmic number of keys.

Variants of the problem. Several variants of our model can also be considered. For example, what if instead of requiring all legitimate users to have access at the end of the algorithm, we only require a $1 - \epsilon$ fraction? Also, the dynamic model can be defined not as an adversarial model but as a stochastic one: instead of having a bound k on the number of adversarial nodes, we can assume that each node is adversarial with probability p. As we will remark at the end of Section 4, both of these variants are easily solvable using the same algorithms that we propose for the main problem (with $\log(1/\epsilon)$ replacing $\log n$ for the first variant, and pn replacing k for the second).

A more challenging variant of the problem concerns situations where the nodes can invite other users to join, forming a trust network structure. This gives rise to a challenging algorithmic problem that will be discussed in Section 5.

3 Static key distribution

The static key distribution problem is equivalent to designing a bipartite graph between the set of n users and the set of m available keys. An edge between a user and a key means that the key is given to that user. k of the user nodes in this graph are adversaries, and all of the neighbors of these nodes will be compromised. After that, a user is *blocked* if all its adjacent keys are compromised.

This is precisely equivalent to a secure overlay network design problem studied by Li et al. [14], although the motivating application and therefore the terminology in that problem are different from ours. Users in our problem are equivalent to Access Points (APs) there, and keys are equivalent to servelets.

The result in [14] that is relevant to our case is Theorem 6 (the adversarial model). That theorem exploits the connection between the solutions of our problem to k-union-free families of sets [13], and proves the following (restated using our terminology):

Theorem 1. *(Restatement of Theorem 6 in [14]) For every m and k, the maximum value of n for which the key distribution problem has a solution is at least $(1 - \frac{k^k}{(k+1)^{k+1}})^{-m/(k+1)}$ and at most $O(k2^{m/k}m^{-1/(2k)})$.*

The lower bound in this theorem is proved using the probabilistic method (giving each key to each user independently with probability $1/(k+1)$), and the upper bound is proved using Sperner's theorem in extremal set theory. A simple calculation from the above theorem gives the following.

Theorem 2. *There is a solution for the static key distribution problem with at most $O(k^2 \log n)$ keys. Furthermore, no key distribution scheme with fewer than $\Omega(k \log(n/k))$ keys can guarantee access to all legitimate users.*

4 Dynamic key distribution

In this section, we study the key distribution problem in the dynamic model, i.e., when the algorithm can respond to a compromised key by providing new keys for affected users. The quantity of interest here is the expected number of keys required before until *every* legitimate user gets access. The following theorem shows that $O(k \log(n/k))$ keys suffice.

Theorem 3. *The dynamic key distribution problem has a solution with at most* $k(1 + \lceil log_2(n/k) \rceil)$ *keys.*

Proof. Consider the following algorithm: In the first round, all n users are divided into k groups of almost equal size, and each group is given a distinct key. After this, any time a key is compromised, the users that have that key (if there is more than one such user) are divided into two groups of almost equal size, and each group is given a new key (i.e., a key that has not been used before). This completes the description of the algorithm.

One can visualize the above algorithm as a tree, where the root corresponds to the set of all n users, and each node corresponds to a subset of users. The root has k children and other nodes have 2. The sets corresponding to the children of a node comprise an almost balanced partition of the corresponding set. In this view, any time the key corresponding to a node is compromised, we move one level down in the tree.

We now show that this procedure uses at most $k(1 + \lceil log_2(n/k) \rceil)$ keys. Consider the k adversarial users. At any point in time, each such node is contained in one group. At the end of the first round, the size of this group is n/k, and after that, every time this adversary compromises a key, the size of this group is divided in half. Therefore, each of the k adversaries can compromise at most $\lceil log_2(n/k) \rceil$ keys. Putting this together with the fact that we started with k keys in the first round gives us the result.

It is not hard to show that in the above algorithm, dividing the users initially into k groups and then into 2 groups in subsequent rounds is essentially the best possible for this style of algorithms. That is, we cannot asymptotically improve the bound given in the above theorem by devising a different partitioning scheme. This might lead one to believe that the bound in the above theorem is asymptotically optimal. However, this is not true. As the following theorem shows for the case of $k = 1$, it is possible to solve the problem with a sublogarithmic ($O(\log n / \log \log n)$) keys. The idea is to "reuse" keys that are already given to people in other branches. The following theorem will later guide us (and will serve as a basis of induction) to a solution for general k with sublogarithmic dependence on n.

Theorem 4. *For* $k = 1$*, the dynamic key distribution problem has a solution with* $O(\frac{\log n}{\log \log n})$ *keys.*

Proof. Let $\ell = \lceil \frac{\log n}{\log \log n} \rceil$. As in the proof of the previous theorem, the algorithm proceeds in a tree-like fashion. At first, the set of all users is divided into ℓ groups of almost equal size, and a distinct key is given to each set. Once a key is compromised, we would know that the single adversary is among the users in the group that has that key. We call this group the *suspicious group*. Since $k = 1$, we know that all users in remaining groups are legitimate users; we call these users *trusted* users. We divide the suspicious group into ℓ subgroups of almost equal size. But instead of giving each subgroup its distinct new key (as was the case in the proof of Theorem 3), we give one subgroup a new key, and give the remaining $\ell - 1$ subgroups the other $\ell - 1$ keys that are already in use by trusted users.

Similarly, when another one of the keys fail, we would know which subgroup the adversary belongs to; so now only users in that subgroup are suspicious, and the remaining $\ell - 1$ subgroups become trusted. Again, we divide the suspicious subgroup into ℓ almost equal-sized subsubgroups. One of these subsubgroups is given a new key, and the remaining $\ell - 1$ are given the keys already in use. We also need to give keys to trusted users whose key is compromised (since there could be trusted users that use the same key as the users in the suspicious subgroup). We give these users an arbitrary uncompromised key already in the system. This process continues until the size of the suspicious group reaches 1, at which point we stop.

Since in each round, the algorithm divides the size of the suspicious group by ℓ, and in each round exactly one key is compromised, the total number of compromised key in this algorithm is at most $\log_\ell n$. This, together with the fact that there are precisely $\ell - 1$ keys that remain uncompromised, shows that the total number of keys used by our algorithm is at most

$$\ell - 1 + \frac{\log n}{\log \ell} \leq \frac{\log n}{\log \log n} + \frac{\log n}{\log \lceil \log n / \log \log n \rceil} = O(\frac{\log n}{\log \log n}).$$

This completes the proof.

We are now ready to give a sublogarithmic bound for the number of required keys for general k.

Theorem 5. *For any k, there is a solution for the dynamic key distribution problem that uses $O(k^2 \log n / \log \log n)$ keys in expectation.*

Proof. First, note that we can assume without loss of generality that $k < \log \log n$, since for larger values of k, Theorem 3 already implies the desired bound. We use induction on k. The $k = 1$ case is already solved in Theorem 4. For $k > 1$, we proceed as follows. The structure of the first stage of our algorithm is similar to the algorithm in the proof of Theorem 4: we fix $\ell = \lceil \frac{\log n}{\log \log n} \rceil$, and start by dividing the users into ℓ groups of almost equal size (to do this, we can put each user in one of the ℓ groups uniformly at random). At any point in this stage, there are precisely ℓ uncompromised keys in the system. Once a key is compromised, we replace it by a new key, split *all* groups that have the

compromised key into ℓ groups and give each group one of the ℓ available keys. In other words, once a key is compromised, it is replaced by a new one and each user who was using that key receives a *random* key among the ℓ available keys. This process is repeated for R rounds, where R will be fixed later.

The sketch of the rest of the proof is as follows: for any legitimate user, in each round the probability that the user receives a key shared by an adversary is at most k/ℓ (since there are k adversaries and ℓ keys available, and key assignments are random). Therefore, in expectation, after R rounds, this user has had at most Rk/ℓ compromised keys. However, at least one of the k adversaries has had at least R/k compromised keys in this stage (since each compromised key is given to an adversary). By setting the right value for R and using the Chernoff bound, we will be able to show that the set of users that have had at least R/k compromised keys contains at least one adversary and almost surely no legitimate user. Given this, we can give each user in this set her personal key, and solve the problem recursively (with at least one fewer adversary) for the remaining set of users.

We now formalize the proof. Consider an arbitrary legitimate user u, and define a sequence of random events that determine the number of times this user's key is compromised. Every time u is given a new key, there is probability of at most k/ℓ (independent of all the previous events) that she receives a key that one of the adversaries in the system already has. If this event occurs, we wait until this key is compromised (if at all) and a new key is given to u; otherwise, the next event corresponds to the next time one of the adversaries compromises a key. At that time, one or more adversaries will receive other random keys. The next event, whose probability can also be bounded by k/ℓ is that one of the keys given to the adversaries is the one u already has. We proceed until the end of the R rounds. This gives us a sequence of events at most R, each with probability k/ℓ of occuring independent of the previous events, and the number of time u's key is compromised can be bounded by the number of events that occur in this sequence. Therefore, this number can be bounded by the sum of R Bernoulli random variables, each with probability k/ℓ of being one. Let this sum be denoted by X. We have $\mu := \mathrm{Exp}[X] = Rk/\ell$. By the Chernoff bound (Theorem 4.1 in [15]) with $1 + \delta = \ell/k^2$ we have:

$$\Pr[X > R/k] = \Pr[X > (1+\delta)\mu] < \left[\frac{e^\delta}{(1+\delta)^{1+\delta}}\right]^\mu < \left(\frac{ek^2}{\ell}\right)^{R/k}.$$

Setting $R = ck \log n / \log\log n$ for a large constant c, the above probability can be bounded by:

$$\exp\left(\frac{c\log n}{\log\log n}\log(\frac{ek^2\log\log n}{\log n})\right) < \exp(-\frac{c}{2}\log n) = n^{-c/2},$$

where the inequality follows from the fact that $k < \log\log n$ and therefore asymptotically, $\log(\frac{ek^2\log\log n}{\log n}) < -\frac{1}{2}\log\log n$. By the above inequality, we know that the probability that the set S of users whose key is compromised at least R/k times contains u is at most $n^{-c/2}$. Therefore, the expected cardinality of this set

is at most $k + n^{1-c/2}$, and hence we can afford to give each user in this set her personal proxy (just in the case of the unlikely event that there is a legitimate user in this set). On the other hand, with probability 1 there is at least one adversary in S. Hence, by removing S, our problem reduces to one with fewer adversaries. We use induction to solve the remaining problem.

The expected total number of keys required by the above procedure is at most ℓ (the initial keys) plus R (the number of keys that replace a compromised key) plus $k + n^{1-c/2}$ (for the set S), plus keys that are needed for the recursive procedure. It is easy to see that this number is $O(k^2 \log n / \log\log n)$.

Our last result is a lower bound that shows that the upper bound in the above theorem is tight upto a factor of k, i.e., as a function of n, the optimal solution to the dynamic key distribution problem grows precisely as $\log n / \log\log n$. This theorem is proved using a simple entropy argument.

Theorem 6. *Any solution to the dynamic key distribution problem requires at least* $\Omega(\frac{k \log(n/k)}{\log k + \log\log n})$ *keys.* 

Proof. Consider an oblivious adversary: initially k random users are adversarial, and in each stage a random adversarial user compromises her key. Let ℓ denote the number of keys used by the algorithm. Since the algorithm eventually proves access to all legitimate users, it must be able to identify the set of all adversarial users. There are $\binom{n}{k}$ such sets, and they are all equally likely. The information that the algorithm receives in each round is the index of the key that is compromised. This index has one of the ℓ values, and therefore can be written in $\log \ell$ bits. Since the total number of rounds is at most ℓ, all the information that the algorithm receives can be written as an $\ell \log \ell$ bit binary sequence. Since this information is enough to find the k adversarial nodes, we must have

$$\ell \log \ell \geq \log \binom{n}{k} = \Omega(k \log(n/k)).$$

A simple calculation from the above inequality implies the lower bound.

Variants of the problem. It is not hard to see that if we only require access for a $1 - \epsilon$ fraction of the legitimate users, the upper bounds in the above theorems can be improved significantly. Namely, Theorem 3 can be adapted to solve the problem with only $O(k \log(1/\epsilon))$ keys.

Also, the upper bounds in this section do not require a precise knowledge of the value of k. Therefore, for the stochastic version of the problem, the same upper bounds hold with k replaced by pn. Furthermore, since the lower bound proof only uses a simple randomized adversary, a similar lower bound holds for the stochastic model. The details of these proofs are left to the final version of the paper.

5 Trust Networks

A common way to build a user base in a country under censorship is through personal trust relationships: the system starts with a few trusted users, and then each trusted user will be allowed to invite new users whom she trusts to the system. In this way, the set of users grows like a *trust network*, a rooted tree on the set of users with the initial trusted users as the children of the root, and edges indicating trust relationships. In fact, newer versions of the Psiphon system rely on an invitation-based method very similar to this to build a user base and a tree structure of trusts among them [17].

Using trust networks to build the user base poses an additional risk to the system: if an adversary infiltrates the network, she can invite new adversarial users (perhaps even fake accounts controlled by herself) to the network, increasing the value of k for the algorithms in the previous section to an unacceptable level. In this section, we formulate this problem theoretically. We will leave it as an open problem for the most part; but for $k = 1$, we give non-trivial solutions, exhibiting that the general problem is interesting and possibly tractable.

The model. We have a population of n users that are nodes of a rooted tree (called the trust network) T. We assume that the depth of the tree T is small (e.g., at most $O(\log n)$).[3] The adversary controls at most k nodes in this tree *and all nodes descending from them.* Our objective is to design a dynamic key distribution scheme that eventually guarantees that each legitimate user receives an uncompromised key. The challenge is to do this with the minimum number of keys.

In the following theorem, we show that for $k = 1$, it is possible to solve this problem with $O(\log n)$ keys. We leave the problem for $k > 1$ as an open question.

Theorem 7. *For $k = 1$, there is a solution for the dynamic key distribution problem on a trust network that uses at most $O(\log n)$ keys.*

Proof (Proof Sketch). We use a binary division method similar to the one used in Theorem 3, except here we keep the divisions consistent with the trust tree T. Recall that the algorithm in the proof of Theorem 3 maintains a *suspicious* group of users, and every time a new key is compromised, it divides the suspicious group into two subgroups of almost equal size, giving a distinct key to each subgroup. We do the same, except we maintain the following invariant: at any point, the *suspicious group* consists of some children $u_1, \ldots, u_r$ ($r \geq 2$) of a node u (we call u the root of the suspicious group), and all descendants of u_i's. Of course we cannot be sure that u and its ancestors are legitimate, but if any of them is an adversary, we have already achieved the goal of giving access to all legitimate users. Therefore, we treat them as if they are legitimate. However, we

[3] This assumption is necessary. For example, if the trust structure is a long path, it is easy to see that the problem has no non-trivial solution, even for $k = 1$. Furthermore, this is a realistic assumption since trust networks are generally small-world networks and have small diameter.

need to be careful about the possibility that the are adversarial, and will try to get our algorithm to use too many keys *after* all legitimate nodes already get access. This can occur if we see the key that is given to nodes that our algorithm already assumes to be legitimate is compromised. In this case, we simply give new personal keys to all the nodes on the path from the root of the tree to the root of the suspicious subtree, and stop the algorithm.

It is not hard to show that since the number of nodes in the suspicious group reduces by a constant factor in each iteration, and the depth of T is at most logarithmic, the total number of keys used by the above algorithm is at most logarithmic in n.

6 Conclusion and Open Problems

In this paper we studied a key distribution problem motivated by applications in censorship circumvention technologies. The algorithms we give for the problem are simple and intuitive, and therefore have a chance at being useful in practice. From a theoretical point of view, still a few questions remain open:

- In the dynamic model, what is the optimal dependence of the number of required keys on k? There is a gap of roughly $O(k)$ between our best lower and upper bounds. Our conjecture is that (at least for k up to $O(\log n)$), $O(k \log n / \log \log n)$ is the right answer for this problem.
  A similar question can be asked for the static problem, but given the connection between this problem and a long-standing open problem in extremal set theory on k-union-free families of sets [13], this problem is unlikely to have a simple solution.
- Our upper bound in Theorem 5 is on the *expected number* of keys used, whereas the weaker bound in Theorem 3 holds with probability 1. Is it possible to prove a bound similar to Theorem 5 with probability 1? This is mainly a theoretical curiosity, since the bound in Theorem 5 holds not only in expectation but also with high probability.
- The key distribution problem with trust networks for $k > 1$.

Acknowledgments. I would like to thank Olgica Milenkovic for pointing me to the literature on superimposed codes. Also, I am thankful to Nicole Immorlica and Abie Flaxman for discussions on an earlier version of this problem.

References

1. Ian Clarke, Oskar Sandberg, Brandon Wiley, Theodore W. Hong, *"Freenet: A Distributed Anonymous Information Storage and Retrieval System,"* in Designing Privacy Enhancing Technologies, Lecture Notes in Computer Science 2009, H. Federrath, ed., pp. 46–66 Springer-Verlag, Berlin, 2001.
2. Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, Brandon Wiley, *"Protecting Free Expression Online with Freenet,"* IEEE Internet Computing, pp. 40–49, January/February, 2002.

3. A. E. Clementi, A. Monti, and R. Silvestri, *Selective families, superimposed codes, and broadcasting on unknown radio networks*. In Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 709-718, 2001.
4. W. Dai and O. Milenkovic, *Weighted Superimposed Codes and Integer Compressive Sensing*, submitted to IEEE Trans. on Inform. Theory, June 2008.
5. Ronald J. Deibert, John G. Palfrey, Rafal Rohozinski, and Jonathan Zittrain (editors), *ACCESS DENIED: The Practice and Politics of Internet Filtering*, The MIT Press, 2008.
6. R. Dingledine, N. Mathewson, and P. Syverson, *Tor: the second-generation onion router*. In Proceedings of the 13th Conference on USENIX Security Symposium, pp. 21–21, 2004.
7. Ding-Zhu Du and Frank K. Hwang, *Combinatorial group testing and its applications*, World Scientific Publishing Co, 2000.
8. Arkadii D'yachkov, Vladimir Lebedev, Pavel Vilenkin, and Sergey Yekhanin, *Cover-free families and superimposed codes: constructions, bounds and applications to cryptography and group testing*, Proceedings of International Symposium on Information Theory (ISIT), p. 117, 2001.
9. Joe Fay, *Iran's revolution will not be televised, but could be tweeted*, The Register, June 16, 2009. available at `http://www.theregister.co.uk/2009/06/16/iran_twitter/`
10. Lev Grossman, *Iran Protests: Twitter, the Medium of the Movement*, Time, June 17, 2009. available at `http://www.time.com/time/world/article/0,8599,1905125,00.html`
11. P. Indyk, *Deterministic Superimposed Coding with Applications to Pattern Matching*. In Proceedings of the 38th Annual Symposium on Foundations of Computer Science, p. 127, 1997.
12. W. Kautz and R. Singleton, *Nonrandom binary superimposed codes*, IEEE Transactions on Information Theory, Volume 10, No 4, pp. 363-377, 1964.
13. D. Kleitman and J. Spencer. *Families of k-independent sets*. Discrete Mathematics, 6:255262, 1973.
14. Li Li, Mohammad Mahdian, and Vahab Mirrokni, *Secure Overlay Network Design*, Algorithmica, Volume 57, No 1, pp. 82–96, May 2010.
15. Rajeev Motwani and Prabhakar Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
16. OpenNet Initiative, ONI reports and articles. available at `http://opennet.net/reports`
17. Psiphon, `http://psiphon.ca`, Developed at the Citizen Lab at the University of Toronto.
18. Reporters Without Borders, *Internet Enemies, 2009 edition*, March 12, 2009. available at `http://www.rsf.org/IMG/pdf/Internet_enemies_2009_2_.pdf`
19. Christopher Rhoads and Loretta Chao, *Iran's Web Spying Aided By Western Technology*, The Wall Street Journal, page A1, June 22, 2009.