Certificate Recommendations to Improve the Robustness of Webs of Trust

Qinglin Jiang, Douglas S. Reeves, and Peng Ning
Cyber Defense Lab
Department of Electrical and Computer Engineering and Computer Science
North Carolina State University, Raleigh, NC 27695-8207 USA
Email: {qjiang, reeves, pning}@ncsu.edu

Abstract. Users establish distributed webs of trust by exchanging certificates. These webs of trust allow users to communicate securely, without the necessity of a central, trusted keyserver. Distributed webs of trust are unfortunately susceptible to attack by malicious users, who issue false certificates.

Previously, it has been suggested that multiple, redundant paths in the certificate graph provide assurance that certificates are not false. We use this insight to propose certificate *recommendations*, which will result in certificate graphs that provide high assurance. In addition, we can identify certificates which do not increase assurance, and are thus candidates for revokation.

We illustrate these concepts on actual PGP keyrings. We show that current PGP keyrings are susceptible to attack. Their robustness against false certficates can be dramatically increased by the addition of a modest number of additional certificates. Furthermore, many of the existing certificates can be eliminated with no reduction in the assurance offered by the keyrings. The result is keyrings with fewer certificates, and with far greater ability to discriminate true and false certificates. The performance of our methods is also discussed.

Keywords: Authentication, certificates, PGP keyrings, graph connectivity.

1 Introduction

Authenticating a user's public key is a very basic requirement for public key cryptography. Failure to authenticate key information can easily lead to security failures. In large-scale systems, key authentication is usually provided by public key infrastructures, such as X.509 [20] and PGP [21]. Generally speaking, in these systems a public key is authenticated by means of certificates. A certificate is a signed message in which an authority speaks about a user's public key.

Obviously, the correctness of a user's public key information in a certificate relies on the certificate issuer or authority. In X.509, certificates can only be issued by an entity referred to as a certificate authority, or CA. A CA is usually secured and trusted, so that it is safe to believe all certificates contain true information. Systems like this are referred to as *hierarchical trust* systems, with CAs as roots of the hierarchy.

In PGP, each user becomes an authority, and issues certificates to each other. Systems like this are referred to as *web of trust* systems. In such systems, it is unrealistic to expect every user to be

fully secure and trustworthy. If users can be malicious, or their computers can be compromised, it is risky to accept all the certificates they provide without question. This lack of confidence in the information provided by webs of trust is a major reason why their rate of adoption has been slow. On the other hand, there are many advantages to the web of trust approach. The distributed nature of webs of trust makes them scalable and robust, and they are much lower cost than hierarchical trust systems (a CA is generally very expensive to build, configure, and administer).

For these reasons, a method of increasing the reliability of web of trust systems would be very useful. Several researchers have suggested the use of redundancy, i.e., requiring the agreement of multiple users to authenticate key information. Our investigation of existing PGP keyrings shows that most of them do not have sufficient redundancy to provide assurance about the authenticity of key information.

Previous work has mainly addressed the issue of how to *measure* assurance. In this paper, our focus is on how to *increase* assurance in web of trust systems. We present algorithms for solving two problems in distributed webs of trust. The first problem is how to augment or *enhance* webs of trust in an efficient way, to increase assurance in the authenticity of key information. The output of our algorithm is a set of recommendations to users about additional certificates to issue. The second problem is how to *reduce* the web of trust by identifying the essential certificates. Essential certificates are those which contribute to a higher level of assurance. The output of the algorithm for this problem is a set of recommendations to users about certificates which are safe to revoke, without decreasing the total assurance.

The paper is organized as follows. Section 2, discusses related work on using redundancy to measure the assurance of user-provided key information. Section 3 defines the research problems and our assumptions. The criteria for evaluating the proposed solutions are also presented. Section 4 describes an algorithm for efficiently improving the assurance of user-provided key information, by enhancing the web of trust. Section 5 presents a method for identifying unnecessary certificates which can be revoked, while keeping a desired level of assurance for the user-provided key information. Section 6 illustrates the performance of our two solutions on both actual and generated webs of trust (PGP keyrings). Section 7 discusses the issue of user willingness to comply with recommendations, and presents results for one form of user preferences. The final section concludes our work, and presents some open problems.

2 Related Work

First we briefly mention some existing public key infrastructures. X.509 [20] has a hierarchical structure composed of many CAs. It has been frequently used in many commercial applications, such as electronic commerce and enterprise applications. PGP [21] is distributed, and is popular in less commercial applications, such as secure email. Some other public key infrastructures, such as SPKI/SDSI [9] and PolicyMaker [5] mainly focus on access control issues. They differ from X.509 and PGP in that they bind public keys directly to access control policies, instead of to identities.[1]

In the web of trust system, each user maintains a set of certificates, which we call a certificate repository. All methods of measuring the assurance of user-provided key information rely on the measurement of certificate chains in the certificate repository. Many of these methods require a trust profile to be established for each user. Based on these trust profiles, the assurance of user-provided key information is computed. For example, [19] and [15] compute the assurance using a single certificate chain. Multiple certificate chains are used by [4] and [14] to increase the level of

[1] In SPKI, "local names" are akin to roles or group names, rather than individual identities.

assurance. In [16], insurance, which can be seen as another way to reduce risk, is used in in the trust calculations.

Another approach is presented in [17]. This method does not require the establishment of user trust profiles. Instead, the authors suggest computing the assurance of user-provided key information by counting the number of public key-independent certificate chains. The paper showed that if at most n public keys are compromised, a public key must be true if there are at least $n+1$ public key-independent certificate chains certifying it. In this paper we adopt the model of [17] for measuring the assurance of user-provided key information. We believe this model is more suitable for the case of malicious users. It is much easier to bound the number of users who may be malicious, than to specify precisely how trustworthy each user is.

In all these methods, multiple certificate chains provide a higher level of assurance. However, while these methods provide a way to measure the assurance of user-provided key information, they do not address how to improve this assurance.

An analysis of the characteristics of PGP keyrings has been done in [6]. This paper does not, however, measure the assurance of PGP keyrings directly. We will show that PGP provides very poor assurance on user-provided key information. This poor assurance is part of the motivation for this paper.

Enhancing the assurance of user-provided key information requires some users to issue additional, recommended certificates. We hope the user would comply with this request in most cases, but recognize that users have preferences about the issuance of certificates. For instance, some users may wish to limit the number of certificates they issue, or may only agree to issue certificates to selected individuals. An incentive for users to issue the recommended certificates is that the global level of assurance will increase by doing so. This assurance represents a level of protection for the entire community against malicious users and false certificates, and thus represents a benefit to all.

The next section presents the definitions and assumptions on which our methods are based. We also define precisely the problem to be solved, and the criteria for assessing the proposed solution.

3 Problem Statement

A *user* is an entity in our system represented by an *identity*, such as the names "Bob" and "Alice". We assume in this paper that each user legitimately has exactly one, unique identity, and associated with that identity is exactly one, unique public key. The set of all users is represented by U.

A *public key certificate* is a triple $\langle x, k_x, s(k_y)\rangle$, where x is an identity, k_x is a public key, and $s(k_y)$ is a digital signature (using the private key correponding to k_y) over the combination of x and k_x. User y is termed the *issuer* of the certificate. For the purposes of this paper, we assume that all certificates are public and are collected in one or more known locations, or *repositories*.[2]

Given a certificate $c = \langle x, k_x, s(k_y)\rangle$, if x agrees that k_x is its unique public key, then c is termed a *true certificate*. Otherwise, c is termed a *false certificate*, and k_x is termed a *false key* for user x. A non-malicious user w who knows some set of identity/key bindings are true will issue certificates for these bindings, signed with her public key k_w. A user y who knowingly issues a false certificate is called a *malicious user*.

Accepting a false certificate $\langle x, k_x, s(k_y)\rangle$ generated by y as true would undermine the foundation of security. Because y knows the private key corresponding to k_x, all messages encrypted with k_x

[2]The issue of how users distribute certificates among themselves, when there are no public repositories, is out of the scope of this paper.

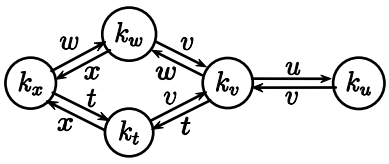


Figure 1: A sample certificate graph

would be disclosed to y. Also, the ability to authenticate x's messages would be compromised, since y would be able to generate digital signatures signed with k_x.

A *certificate chain* is a sequence of certificates where:

1. the starting certificate, which is called the *tail* certificate, is known to be true;
2. each certificate contains a public key that can be used to verify the digital signature associated with the next certificate in the sequence; and,
3. the ending certificate, which is called the *head* certificate, contains an identity/key binding of interest. This identity/key binding is called the *target* of the certificate chain.

Certificate chains are the basic means of bootstrapping trust among a community of users in a web of trust system.

To prove that a certificate is true, we use a result from [17], summarized below. Two certificate chains are *public key-independent* if they have no keys in common, other than their heads.

Theorem 1 *Given $n+1$ public key-independent certificate chains with the same target, if there are at most n malicious users, then the head certificates must be true.*

This result is the basic insight needed to prove that certificates are true. Intuitively, this theorem states that a certificate must be true if there are more than n independent sources of information to show it is true in a system with at most n malicious users.

In the rest of this paper we use *certificate graphs* to represent the collection of certificates. A certificate graph G consists of a set V of vertexes and a set E of arcs (directed edges). In this graph, a vertex labeled k in V represents the public key k. There is an arc labeled x from vertex k_y to vertex k_x if and only if there exists a certificate $\langle x, k_x, s(k_y)\rangle$. That is, the key k_y is used to sign a certificate binding x to k_x. A certificate chain is represented by a directed path in the certificate graph, starting from a key known to be true.

From the above, if there are $n+1$ vertex-disjoint paths in the certificate graph, each starting from a known true key and all having the same target, the head certificates of these paths must be true as long as there are at most n malicious users. The number of vertex disjoint paths can be computed by a maximum flow algorithm using an edge-splitting technique[1]. This algorithm runs in $O(|V||E|log(\frac{|V|}{|E|}))$ time, where $|V|$ is the number of vertexes and $|E|$ is the number of edges.

Figure 1 is a sample certificate graph. In this graph, there are ten arcs correponding to ten certificates. Suppose for this web of trust there is a single malicious user. Users can determine the true keys/certificates using Theorem 1 in the following way. For user x, k_w and k_t are true keys because they are certified by x directly. k_v is also determined to be true because there are two vertex-disjoint paths from k_x to k_v. However, user x is unable to determine if k_u is true or not, because there is only one vertex-disjoint path from k_x to k_u. Following the same idea, user u is only able to determine that k_v is a true key. She is not able to determine any other keys are true.

We now introduce the notion of *assurance* in a certificate graph. Let $K_x(G)$ represent the set of all *true* keys reachable from vertex k_x in the certificate graph G. Let $K_x^T(G,n)$ represent the set of keys which can be proved to be true by the above method (or which are assumed to be true), assuming initially only k_x and the keys it directly certifies are true, and that there are no more than n malicious users. Then the *assurance for user x for the certificate graph G in the presence of no more than n malicious users* is defined to be

$$\mathcal{A}(x,G,n) \;=\; \frac{\mid K_x^T(G,n) \mid}{\mid K_x(G) \mid}$$

We define the total, or aggregate, assurance of a certificate graph G for all users in the presence of no more than n malicious users to be

$$\mathcal{A}(G,n) \;=\; \frac{\sum_{x \in U} \mid K_x^T(G,n) \mid}{\sum_{x \in U} \mid K_x(G) \mid}$$

Assurance can thus range from a minimum of 0% (worst) to 100% (best).

We can now state the problem addressed by this paper. The first goal ("Goal #1") is the following. Given a certificate graph G and a value n representing the maximum number of malicious users, add the minimum number of arcs (i.e., add certificates) to this graph so that $\mathcal{A}(G,n) = 100\%$. Secondly ("Goal #2"), given a graph G and value n for which $\mathcal{A}(G,n) = 100\%$, remove the maximum possible number of arcs (i.e., revoke certificates) from G without reducing $\mathcal{A}(G,n)$ to less than 100%.

Certificates are created and revoked by users, and in most systems those users are not under central control. For this reason, we refer to the certificates added and revoked by a solution to the above problem as certificate *recommendations*. We return to the issue of user preferences and their impact on recommendations at the end of this paper. The next section presents a method for accomplishing the first goal.

4 Enhancing Certificate Graphs

From the preceding discussion, assurance is increased when the number of vertex-disjoint paths between pairs of vertexes is increased. A graph G is said to be *q-connected* if there does not exist a set of q or fewer vertexes whose removal disconnects the graph. Menger's theorem states that a graph is q-connected iff every pair of vertexes is joined by at least q vertex-disjoint paths [12]. The key requirement in achieving goal #1 is therefore to make the certificate graph G $n+1$-connected, where n is the maximum number of malicious nodes. We denote the connectivity between two nodes u and v as $q(u,v)$.

The connectivity $q(u,v)$ between u and v can be computed by a maximum flow algorithm [1] in time $\mathrm{O}(|V||E|\log(\frac{|V|}{|E|}))$, where $|V|$ is the number of vertexes and $|E|$ is the number of edges. Let $q(G)$ represent the minimum connectivity of any pair of vertexes in graph G, i.e.,

$$q(G) = min\{q(u,v)|\text{ ordered pair } u,v,(u,v) \notin E\}.$$

$q(G)$ can be easily determined by computing the maximum flow for every ordered pair of vertexes in G and taking the minimum. However, we propose the algorithm below as a faster means of computing $q(G)$. This algorithm follows an approach from [1] for computing edge connectivity of a directed graph. The set $P(v)$ of predecessors of vertex v is defined as $P(v) = \{u|(u,v) \in E\}$, and the set $O(v)$ of successors of vertex v is defined as $O(v) = \{u|(v,u) \in E\}$.

Algorithm 1 q-connectivity algorithm

input: digraph G

output: vertex connectivity $q(G)$

1: select an arbitrary vertex u
2: compute $c_1 = \min\{q(u,v) \mid v \in V - \{u\},\ (u,v) \notin E\}$
3: compute $c_2 = \min\{q(v,u) \mid v \in V - \{u\},\ (v,u) \notin E\}$
4: compute $c_3 = \min\{q(x,y) \mid x \in P(u),\ y \in O(u),\ (x,y) \notin E\}$
5: $k(G)$=min$\{c_1, c_2, c_3\}$

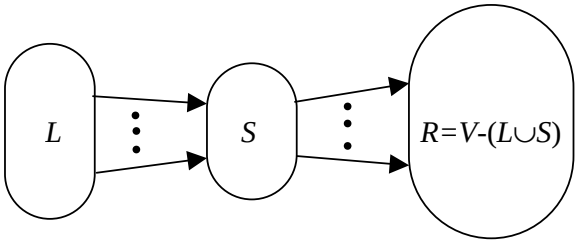


Figure 2: Graph $G = (V, E)$ and a minimum vertex-separator S

Algorithm 1 requires computing the maximum flow only O($2|V|$) times. We now explain algorithm 1.

Let u and v be a pair of distinct vertexes in G. If $(u,v) \notin E$, let $q(u,v)$ represent the least number of vertexes, chosen from $V - \{u,v\}$, whose deletion from G would disconnect every path from u to v. The vertex connectivity $q(G)$ of a digraph is the least cardinality $|S|$ of a vertex set $S \subset V$ whose deletion from G would either make $G - S$ disconnected. Such a set S is called a *minimum vertex-separator*.

Suppose by some method we have found a pair of vertexes u and v such that $q(u,v) = q(G)$. We consider the following abstraction of a directed graph G and an arbitrary minimum vertex-separator S in G. In Figure 2, S is a minimum vertex-separator whose deletion would destroy all paths from u to v. L contains u and all vertexes reachable from u after deletion of S. R contains the remaining vertexes. Clearly, for all $w \in L$ and all $z \in R$, $q(w,z) = q(u,v) = q(G)$.

For an arbitrary vertex x, consider the following three possibilities for the location of x.

1. $x \in L$ (Line 2 in Algorithm 1). If there is at least one vertex y in R, then $q(G) = q(x,y)$. We simply compute the connectivity from x to all other vertexes. At least one vertex must be in R.
$$q(G) = min\{q(x,y) | (y \in V - x) \wedge (y \text{ is not adjacent to } x \text{ in } G)\}.$$

2. $x \in R$ (Line 3 in Algorithm 1). If there is at least one vertex y in L, then $q(G) = q(y,x)$. We simply compute the connectivity from all other vertexes to x. At least one vertex must be in L.
$$q(G) = min\{q(y,x) | (y \in V - x) \wedge (y \text{ is not adjacent to } x \text{ in } G)\}.$$

3. $x \in S$ (Line 4 in Algorithm 1). In this case, at least one of x's predecessors must be in L, and at least one of x's successors must be in R. We simply compute the vertex connectivity from all of x's precedessors to all of x's successors and choose the minimum one.

The problem of increasing the connectivity of a graph by adding a minimal number of arcs is referred to as the *graph connectivity augmentation* problem. Some work on this problem includes [10, 3, 2]. These papers provide optimal solutions to this problem. They are theoretically feasible but in practice, they may not be appropriate. For example, [10] relies on the ellipsoid method which is well known to be impractical. [2] requires a running time of $O(|V|^6 f(k))$ where $|V|$ is the number of vertexes, $f(k)$ is a super-exponential function of k and k is the connectivity.

To efficiently solve this problem practically, we propose a heuristic for the graph augmentation problem. Given is a certificate graph G and the desired graph connectivity q. The algorithm outputs a graph G' which is *enhanced* or augmented to have a connectivity of at least q. Before describing the algorithm we introduce some notation. Let $G = (V, E)$ represent a digraph(directed graph) without loops or multiple edges. The *in-degree* $deg_{in}(v)$ of a vertex v in V is defined as the number of arcs in G whose head is v, while the *out-degree* $deg_{out}(v)$ of v is defined as the number of arcs in G whose tail is v. The *minimum degree* of a graph G is represented as $\delta(G)$, and is defined as

$$\delta(G) = min\{deg_{in}(v), deg_{out}(v) | v \in V\}$$

The heuristic has two phases, and is shown below as Algorithm 2. In Phase 1, arcs are added between vertexes with the smallest in- or out-degrees. The purpose is to increase $\delta(G)$ to reach q by adding a minimum number of arcs. In Phase 2, each vertex's in- and out-degree is at least q but $q(G)$ may not reach q. To help $q(G)$ to reach q, arcs are directly added between pairs of vertexes with connectivity lower than q.

Algorithm 2 *graph connectivity augmentation algorithm*

input: $G(V, E)$, q
output: $G'(V, E')$

1: $G'(V, E') = G(V, E)$
Phase 1
2: construct an arc list $L = \{(u, v) \mid u \in V, v \in V, (u, v) \notin E', (deg_{out}(u) < q \vee deg_{in}(v) < q)\}$ ordered by SUM$(deg_{out}(u) + deg_{in}(v))$
3: **while** $\delta(G') < q$ **do**
4: choose the first arc e in L
5: add arc e to E'
6: update L
7: **end while**
Phase 2
8: run algorithm 1 and construct the arc list $L = \{(u, v) \mid q(u, v) < q,\ u \in V,\ v \in V\}$ ordered by $q(u, v)$.
9: **while** $q(G') < q$ **do**
10: choose the first arc e in L
11: add e to E'
12: run algorithm 1 and construct the arc list $L = \{(u, v) \mid q(u, v) < q,\ u \in V,\ v \in V\}$ ordered by $q(u, v)$.
13: **end while**

Phase 1 requires a running time of $O(|V|)$. Phase 2 requires $O(|V|^2)$ calls to algorithm 1. The worst case complexity of algorithm 2 is $O(|V|^3|E|log(\frac{|E|}{|V|}))$. In our experiments described in section 6 we will show the average case execution time is very reasonable.

With this method, we can now address the goal of increasing the assurance of a certificate graph to 100%. For a maximum number of malicious users n, the certificate graph is enhanced or augmented using the above method, with $q = n + 1$. The method is guaranteed to terminate, and the enhanced certificate graph is guaranteed to have an assurance of 100%. The arcs added by the method represent recommendations of certificates to add to the certificate graph.

We now present a method for accomplishing goal #2 (remove the maximum possible number of arcs from G without reducing $\mathcal{A}(G, n)$ to less than 100%).

5 Reducing Certificate Graphs

There are multiple reasons for reducing the number of certificates to the minimum possible for a desired level of assurance. In some environments (e.g., mobile terminals) the certificates may be distributed among the users, and each user may have a limited amount of storage. Some of the methods of analyzing assurance, and determining reliability of key information, run in time that is determined by the number of certificates. In addition, some users may prefer to issue only a limited number of certificates. For all of these reasons, we believe reducing certificate graph size is important.

The problem of removing the maximum number of edges from a graph while maintaining a given connectivity q is referred to as the *minimum q-vertex connected spanning subgraph problem*. A minimum q-vertex connected spanning subgraph for a digraph G is a spanning subgraph whose number of arcs is minimum and which is still strongly connected after removing any fewer than q vertexes. This problem has been shown to be NP-Complete [11]. Various heuristics have been developed to solve this problem (see, for instance, [7]). We have developed our own straightforward heuristic for solving this problem in polynomial time.

Algorithm 3 *minimum q vertex-connected spanning subgraph algorithm*

input: q-vertex connected digraph G, q
output: q-vertex connected minimum spanning subgraph G'

1: $Q'(V, E') = Q(V, E)$
2: **for** each arc $e = (u, v)$ in E' **do**
3: **if** $deg_{out}(u) > q \wedge deg_{in}(v) > q$ **then**
4: remove e from E'
5: **if** $q(G') < q$ (Algorithm 1) **then**
6: restore e to E'
7: **end if**
8: **end if**
9: **end for**

Algorithm 3 requires $O(|E| - q|V|)$ calls to algorithm 1.

With this method, we can address the goal of reducing the number of certificates to the minimum possible for a given level of assurance. For a maximum number of malicious users n, the certificate graph is reduced using the above algorithm, with $q = n+1$. The method is guaranteed to terminate with a graph for which assurance remains at 100%. The arcs removed by the method represent recommendations about which certificates are safe to revoke with no detriment to the assurance.

We have described heuristic methods for accomplishing our two goals. In the next section, we examine the performance of these heuristics on actual webs of trust: PGP keyrings. In the process, we will also make some observations about the assurance provided by these PGP keyrings.

6 Experimental Results

The effectiveness and practicality of the proposed methods depends on the data to which it is applied. The best examples of webs of trust that are widely available are PGP keyrings[8]. We therefore used these for purposes of experimental validation.

There are many PGP keyservers, for example, pgp.cc.gatech.edu and wwwkeys.ch.pgp.net. The typical size of PGP keyring at each server is around 27,000 keys. [3]

The PGP keyrings we use in experiments are downloaded from these keyservers and the keyanalyze [18] tool was used to extract many strongly-connected components. In the many extracted strongly-connected components, the size of each keyring range from 20 to 16449 keys. The PGP keyrings used in this experiment are listed in figure 3.

KR25	Pgp keyrings with 25 keys and 208 certificates
KR105	Pgp keyrings with 105 keys and 245 certificates
KR588	Pgp keyrings with 588 keys and 5237 certificates
KR1226	Pgp keyrings with 1226 keys and 8779 certificates

Figure 3: PGP keyrings used in this experiment

While KR588 and KR1226 are not the largest keyrings, they exercise the capabilities of our methods thoroughly, and it is possible to run large numbers of experiments on variations of these keyrings. Experiments were also run for those smaller keyrings, with similar results; these experiments are omitted for space reasons.

We first analyzed the assurance of the 4 actual keyrings in figure 3 for varying values of n, the number of malicious users. The results are shown in Figure 4. Two of these keyrings exhibit very low assurance values for $n > 1$. A third keyring (KR588) has better assurance, but also drops to a low level for $n > 5$. Only the smallest keyring, KR25, shows a fairly high level of assurance. This analysis provides strong evidence that large actual webs of trust do not achieve high levels of assurance. We believe this demonstrates the need for recommendations to increase the robustness of webs of trust to attack by malicious users.

In the second set of experiments, we attempted to increase the assurance of the actual keyrings KR588 and KR1226. As mentioned, these are some of the larger connected components in the PGP keyring repositories. We used algorithm 2 to increase its assurance to 100% for varying levels of n. The performance of the proposed method is shown in Figure 5.

The experimental results show a consistent behavior when our certificate graph augmentation heuristic is applied. Namely, there is an initial slow increase in assurance, followed by a much more rapid, almost linear increase, until an assurance of 100% is reached. Our explanation is the following. When adding certificates for a target value of n (number of malicious users), the certificates initially added increase more or less uniformly the connectivity up to a level of n. There is thus little change in the $q = n + 1$-connectivity of the graph. Once this uniform level has been reached however, every certificate added increases the $q = n+1$-connectivity of the graph, resulting in the linear increase in assurance. The final, negligible slower rate of increase in some cases to reach an assurance of 100% is due to phase 2 of the algorithm.

We next analyzed the performance of our heuristic for adding certificates to the graph. We compared the performance with a method which randomly adds arcs to the graph. Any reasonable

[3] A good place to retrieve PGP keyserver information is PGP-keyserver-folk.

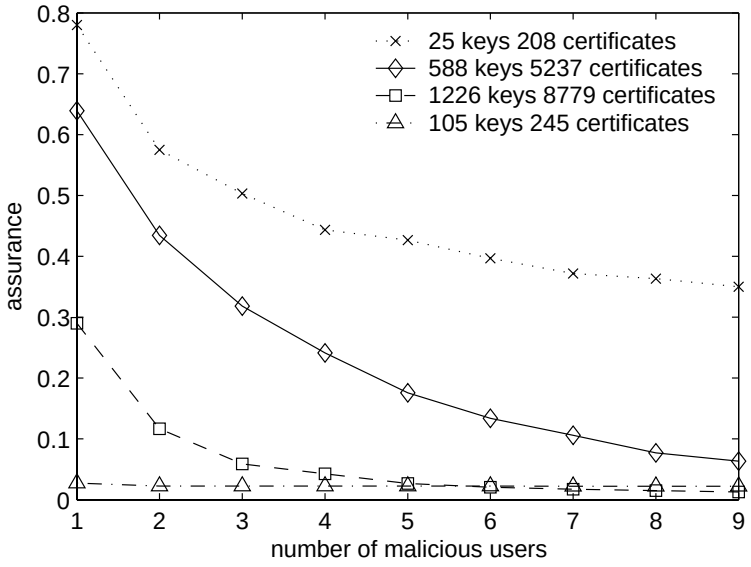
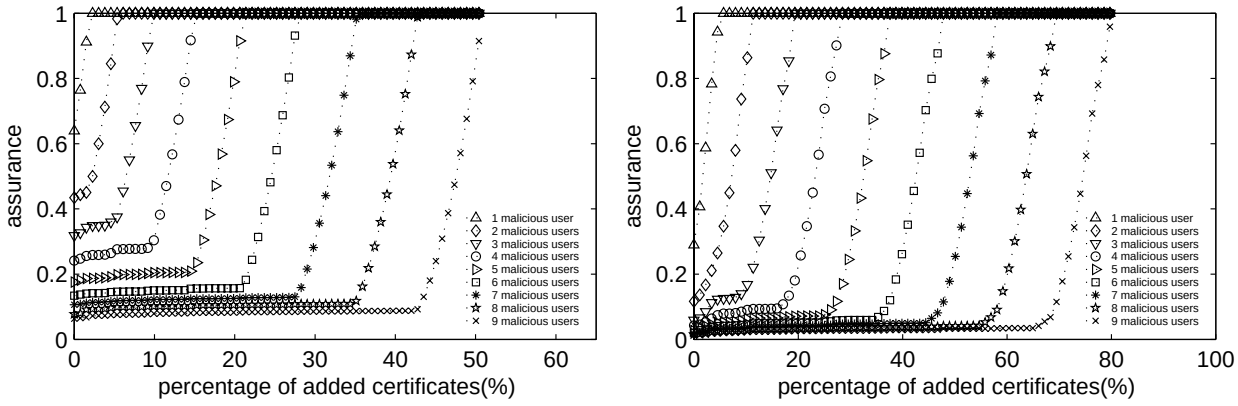


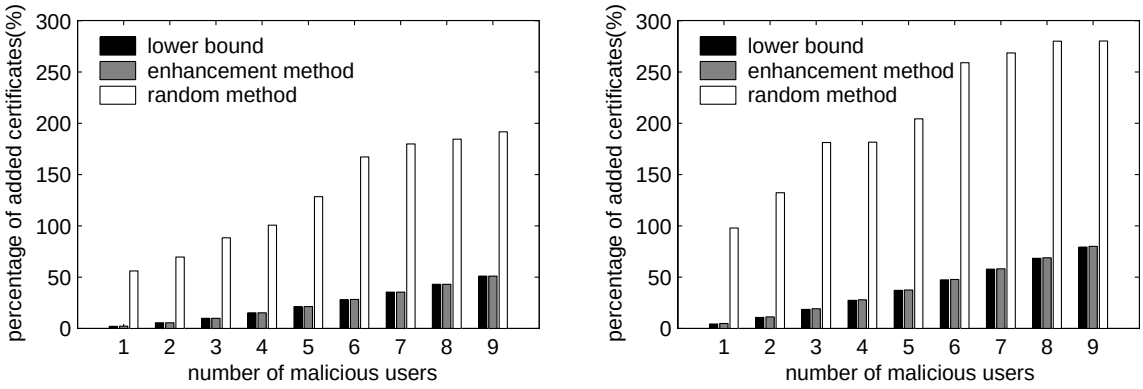
Figure 4: Assurance of some actual PGP keyrings



(a) 588 keys and 5237 certificates

(b) 1226 keys and 8779 certificates

Figure 5: Performance of algorithm 2 on two PGP keyrings



(a) 588 keys and 5237 certificates

(b) 1226 keys and 8779 certificates

Figure 6: Comparison of the lower bound, the enhancement method and the random method for two PGP keyrings

heuristic should of course perform better than this method. We also computed a lower bound for the number of arcs which must be added by *any* method in order to achieve $q = n+1$-connectivity. By the $n+1$-degree requirement for connectivity, it is straightforward to see that the minimum number of arcs that must be added is at least

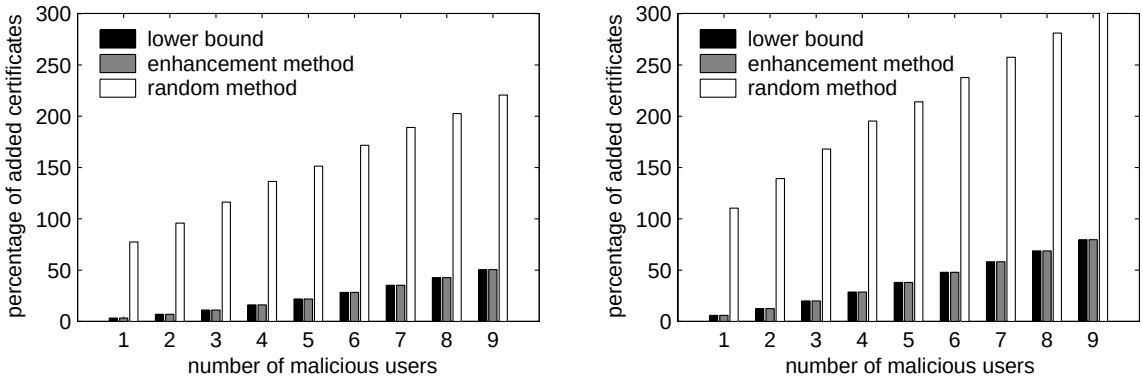
$$max(\sum_{v \in V} |def^q_{in}(v)|, \sum_{v \in V} |def^q_{out}(v)|),$$

where $\text{def}^q_{in}(v)$ is the in-degree *deficiency* of vertex v, i.e., the difference between q and $\deg_{in}(v)$, and $\text{def}^q_{out}(v)$ is similarly defined for the out-degree. The performance of our method (number of arcs added) relative to the random method, and to this lower bound, is shown in Figure 6 for the actual keyrings KR588 and KR1226. It is clear that our heuristic performs much better than the random method. In addition, our method is very close to optimal for these two keyrings; in all cases, the difference between the lower bound and our method was no greater than 0.7%.

To increase confidence that these results are representative, we also synthetically generated keyrings using the model of [6]. This keyring generator will take the degree distribution of a given certificate graph and then generate many synthetic certificate graphs which have similiar degree distributions. In this experiment, we generate two types of keyrings, using a degree distribution of KR588 and KR1226, respectively. For each type of keyring, 50 synthetic keyrings were generated, based on the method of [6]; the variations in these 50 synthetic keyrings were due solely to the use of different random number seeds. The results are shown in Figure 7. Each value shown in this graph is the average of 50 synthetic instances, and has a 99% confidence interval of ± 5 certificates. The results show our method consistently performs close to optimal, and much better than a random method.

We next investigate the performance of our heuristic for computing the minimum $q = n + 1$-vertex connected spanning subgraph for PGP keyrings. In this case, we compare with another lower bound. For a $q = n + 1$-vertex connected digraph, a trivial lower bound is just the product of the minimum vertex degree and the number of vertexes, i.e., $(n + 1) * \mid V \mid$. Figure 8 shows the results of our experiments on the actual keyrings KR588 and KR1226.

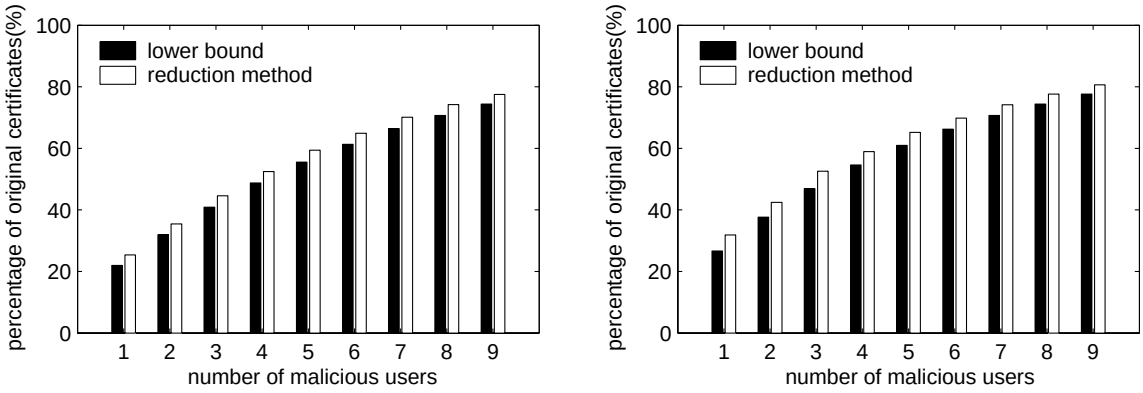
These results show that the performance of the heuristic algorithm 3 is also close to optimal.



(a) 588 keys and 5237 certificates

(b) 1226 keys and 8779 certificates

Figure 7: Comparison of the lower bound, the enhancement method and the random method for 50 synthesized keyrings, for each of two actual keyrings



(a) 588 keys and 5237 certificates

(b) 1226 keys and 8779 certificates

Figure 8: Comparison between the lower bound and the reduction method for two PGP keyrings

These experiments also yield useful insights into existing PGP keyrings. First, it may be seen that the "cost" (additional certificates) to achieve 100% assurance for a small number of malicious users is small (e.g., less than 10% additional certificates with no more than 3 malicious users). However, the overhead grows steadily, and for a large number of malicious users (e.g., greater than 10) it may be expected this overhead is substantial.

The reduction experiment, however, reveals that most of the certificates in existing PGP keyrings are unnecessary, at least from the standpoint of providing 100% assurance against a small number of malicious users. The total number of certificates in these PGP keyrings need not increase from their current number until the number of malicious users being tolerated is at least 10. However, to achieve this benefit will require users to generate a different set of certificates than they are currently generating.

Our experiments also show the average case running time for algorithm 2. It requires no more than $\frac{2|V|}{100}$ calls to algorithm 1 in phase 2. The average case running time of algorithm 2 is $\frac{1}{50}(|V|^2|E|log(\frac{|E|}{|V|}))$. We measured the running time of our heuristics on a PC with a 2.4GHz Pentium IV processor and 512MB of memory. For the PGP keyring KR588, in all cases algorithm 2 requires less than 2 minutes to achieve an assurance level of 100%. For this same keyring, the reduction heuristic (algorithm 3) requires less than 30 minutes in all cases. For the keyring KR1226, the corresponding times are 30 minutes (enhancement) and 4 hours (reduction). If these algorithms are executed off-line and infrequently, and on PGP keyrings of similar size, these running times are acceptable. If the method needs to be executed interactively, however, or for much larger PGP keyrings, speedups in processing will be necessary.

7 User Preferences (Constraints)

In our discussion of the problem we termed the addition and revocation of certificates *recommendations*. This is an acknowledgment that certificates are generated by people, frequently for personal reasons. It may be expected that some users will be more willing than others to follow the recommendations generated by automated methods, such as those presented in this paper.

We propose that user preferences be incorporated into the method for enhancing certificate graphs, in the form of constraints. The many forms such preferences can take is outside the scope of this short discussion. We only intend to indicate one approach and its impact in this paper. For these purposes, we propose to model user preferences as *buddy lists*. A buddy list for user u is the set of other users for whom she is willing to issue a certificate.

Incorporating such a constraint into algorithm 2 is simple. In this algorithm, when building the candidate arc list L, we simply exclude those arcs (u, v) where v is not in u's buddy list. One consequence of buddy lists (and of user preferences in general) is that it is no longer possible to guarantee that the enhancement method will always terminate with an assurance level of 100%.

To investigate the impact of user preferences, we ran the following experiment. We artificially generated a buddy list for each user. The sizes of user buddy lists followed a power-law distribution, with exponent α = -1, a specified minimum value b, and a maximum value $\mid V \mid/10$. The power-law distribution has been frequently found in many networks, and research shows the relationships between persons in a social network follows power law[13]. The minimum value is enforced by simply discarding any randomly generated value less than b, and similarly for the maximum value. The selection of buddies is uniformly distributed among the other users. We generated 9 different sets of user preferences (sets of buddy lists), for values of b from 2 to 10. For each set of user preferences, we measured the achievable assurance with our method while also varying the number of malicious users from 2 to 10. The results are shown in Figure 9.

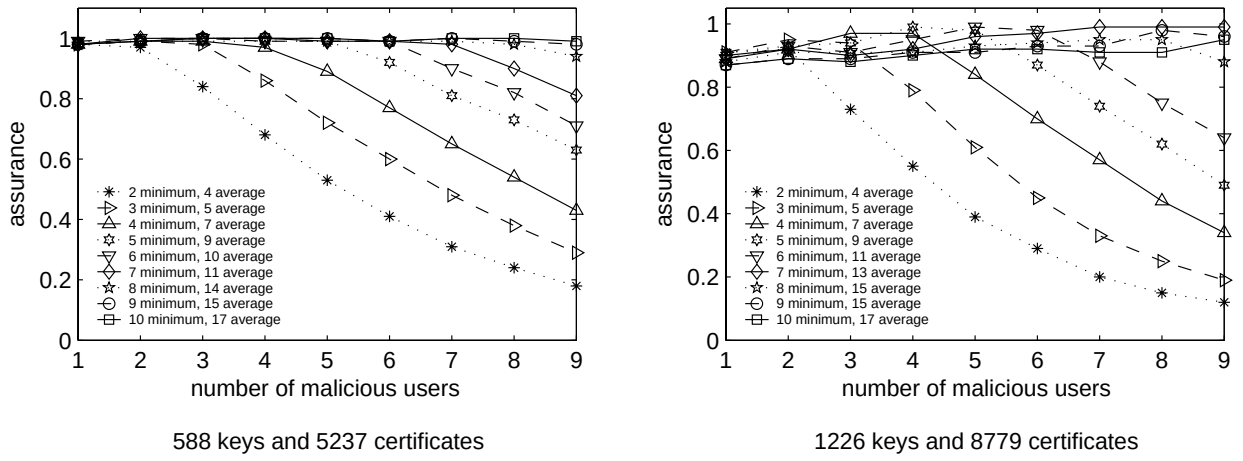


Figure 9: Achievable assurance for different buddy list sizes

From this graph it can be seen that the assurance level is roughly 100% until the number of malicious users exceeds the minimum size of the buddy list. Beyond that point, assurance decreases roughly linearly with the increase in the number of malicious users; the rate of decrease declines as the minimum size of the buddy list goes up. There is little impact on the running time of the algorithm to incorporate user preferences such as buddy lists. The more significant impact is that users must be more flexible in their preferences (e.g. expand their buddy lists) to achieve high assurance in the face of increasing number of malicious users.

8 Conclusions and Future Work

In this paper we described how distributed webs of trust can be made resistant to false certificates issued by malicious users. Our approach requires increasing the vertex connectivity of a certificate (directed) graph, which is a known problem with optimal (but complex) solutions. We presented a heuristic for this problem which runs in worse case $O(|V|^3|E|log(\frac{|E|}{|V|}))$ time and average case of $\frac{1}{50}(|V|^2|E|log(\frac{|E|}{|V|}))$.

It is possible to minimize the number of certificates necessary to maintain a specified degree of robustness by reducing the certificate graph. This is equivalent to finding a minimum q-vertex-connected spanning subgraph, for which optimal (but complex) algorithms are also known. We presented a heuristic for this problem which runs in $O((|E| - q|V|)|V|^2|E|log\frac{|E|}{|V|})$ time.

We applied these algorithms to current PGP keyrings. Our analysis shows that current PGP keyrings are highly vulnerable to false certificates issued by malicious users. Experimental results indicate that the robustness of PGP keyrings can be greatly increased with only a small increase in the number of certificates. In addition, the results show that there is a great deal of useless redundancy in current PGP keyrings that can be eliminated, reducing the number of certificates with no loss in robustness against false certificates.

Finally, we addressed the issue of user behavior and the constraints that adds. We demonstrated by experiments that PGP keyrings can be made resistant to false certificates as long as user "buddy lists" are larger than the number of malicious users.

Future work will focus on how to integrate this method with PGP keyrings, in the form of

recommendations to users about the certificates they are issuing. We are also extending this method to the case of users who may hold multiple public keys.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[2] Tibor Jordn Andrs Frank. Directed vertex-connectivity augmentation. *Mathematical Programming*, 84(3):537–553, 1999.

[3] Andrs A. Benczr. Pushdown-reduce: an algorithm for connectivity augmentation and poset covering problems. *Discrete Applied Mathematics*, 129(2-3):233–262, 2003.

[4] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *Proceeding of the 3rd European Symposium on Research in Computer Security (ESORICS 94)*, pages 3–18, 1994.

[5] M. Blaze and J. Feigenbaum. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland CA USA, 6-8 May 1996.

[6] Srdjan Capkun, Levente Buttyn, and Jean-Pierre Hubaux. Small worlds in security systems: an analysis of the pgp certificate graph. In *Proceedings of the 2002 workshop on New security paradigms*, pages 28–35. ACM Press, 2002.

[7] Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000.

[8] Darnell D. Pgp or pki? the future of internet security. *EDI Forum: The Journal of Electronic Commerce*, 12(1):59–62, 1999.

[9] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. RFC 2693: SPKI certificate theory, September 1999.

[10] A. Frank and T. Jordan. Minimal edge-coverings of pairs of sets. *Journal of Combinatorial Theory, Series B*, 65(1):73–110, 1995.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman & Co., 1979.

[12] Frank Harary. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969.

[13] Adamic LA, Lukose RM, Puniyani AR, and Huberman BA. Search in power-law networks. *PHYSICAL REVIEW E*, 64(4), 2001.

[14] Ueli Maurer. Modelling a public-key infrastructure. In *Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS 96)*, pages 324–350, 1996.

[15] S. Mendes and C. Huitema. A new approach to the X.509 framework: Allowing a global authentication infrastructure without a global trust model. In *Proceedings of the Symposium on Network and Distributed System Security, 1995*, pages 172–189, San Diego, CA , USA, Feb 1995.

[16] M. Reiter and S. Stubblebine. Toward acceptable metrics of authentication. In *IEEE Symposium on Security and Privacy*, pages 10–20, 1997.

[17] M. Reiter and S. Stubblebine. Resilient authentication using path independence. *IEEE Transactions on Computers*, 47(12), December 1998.

[18] M. Drew Streib. Keyanalyze - analysis of a large OpenPGP ring. http://www.dtype.org/keyanalyze/.

[19] Anas Tarah and Christian Huitema. Associating metrics to certification paths. In Yves Deswarte, Gérard Eizenberg, and Jean-Jacques Quisquater, editors, *Computer Security - ESORICS 92, Second European Symposium on Research in Computer Security, Toulouse, France, November 23-25, 1992, Proceedings*, volume 648 of *Lecture Notes in Computer Science*, pages 175–189. Springer Verlag, 1992.

[20] Int'l Telecommunications Union/ITU Telegraph & Tel. ITU-T recommendation X.509: The directory: Public-key and attribute certificate frameworks, Mar 2000.

[21] Philip Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, Mass., 1995.