Certificate Recommendations to Improve the Robustness of Web of Trust*

Qinglin Jiang, Douglas S. Reeves, and Peng Ning

Cyber Defense Lab
Departments of Computer Science and Electrical and Computer Engineering
N. C. State University Raleigh, NC 27695-8207 USA
{qjiang,reeves,pning}@ncsu.edu

Abstract. Users in a distributed system establish webs of trust by issuing and exchanging certificates amont themselves. This approach does not require a central, trusted keyserver. The distributed web of trust, however, is susceptible to attack by malicious users, who may issue false certificates. In this work, we propose a method for generating certificate *recommendations*. These recommendations guide the users in creating webs of trust that are highly robust to attacks. To accomplish this we propose a heuristic method of graph augmentation for the certificate graph, and show experimentally that it is close to optimal. We also investigate the impact of user preferences and non-compliance with these recommendations, and demonstrate that our method helps identify malicious users if there are any.

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

1 Introduction

Authenticating a user's public key is a very basic requirement for public key cryptography. In large-scale systems, this function is usually provided by public key infrastructures, such as X.509 [25] and PGP [27]. Generally speaking, in these systems public keys are 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 it is safe to believe all certificates contain truthful 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 work is partially supported by the U.S. Army Research Office under grant DAAD19-02-1-0219, and by the National Science Foundation under grant CCR-0207297.

For this reason, a method of measuring the robustness of web of trust systems would be very useful. Very intuitively, several researchers have suggested the use of redundancy, i.e., the agreement of multiple users to authenticate the same key information. Following this method, investigation of existing PGP keyrings shows that most of them do not provide a high degree of assurance in the validity of key information[14]. These previous works have mainly addressed the issue of how to *measure* robustness. In this paper, our focus is on how to *enhance* robustness in web of trust systems.

The paper is organized as follows. Section 2 discusses related work on using redundancy to measure the robustness of web of trust. Section 3 defines the research problems and our assumptions. Section 4 describes an algorithm for efficiently enhance the robustness of web of trust. Section 5 illustrates the performance of our solutions on both actual and generated web of trust using experiments. Section 6 investigates the impact of users' willingness and users' preference on the recommendations, respectively. Section 7 discusses and shows how our recommendation method may be used to help users to detect attacks and identify malicious users. 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 [25] is centralized and has a hierarchical structure composed of many CAs. PGP [27] is distributed, and is popular in personal communications. Some other public key infrastructures, such as SPKI/SDSI [8], Delegation Networks [2] and PolicyMaker [5] mainly focus on access control issues. Most of the methods [24, 18, 4, 17], or so called "trust metrics" measure the robustness of web of trust based on the trust profile established for each user. A new concept of insurance is introduced in [21] and a maximum flow approach is presented in [16]. A different approach is presented in [22] which computes the assurance by counting the number of public key-independent certificate chains. The method of [14] is similar to [22] but investigates the case that each user may possess multiple public keys. It also shows practical PGP keyrings provides poor assurance on users' key information and investigates certificate conflicts. Certificate conflicts were first pointed out to be very important in [22] and is analyzed in [14]. However, while these methods provide a way to measure the robustness of web of trust, they do not address how to enhance it.

The next section presents the definitions and assumptions of the research problem.

3 Problem Statement

First we briefly introduce some notions in web of trust systems. A *user* is an entity who participates in web of trust and who will receive our recommendations. He or she may be represented by a *name*, such as "Bob" and "Alice". The set

of all names is denoted by U. A *public key certificate* is a signed name-to-key binding represented by a triple $\langle x, k_x, s(k_y)\rangle$, where x is a name, k_x is a public key, and $s(k_y)$ is a digital signature (generated using the private key corresponding to k_y) over the binding of x and k_x. The set of all certificates are usually stored and collected in one or more public known locations, i.e. *keyservers* or *repositories*.[1]

In all methods of authenticating public keys [24, 18, 4, 17, 21, 22, 16, 14], multiple certificate chains represent high assurance for the name-to-key binding in question. More specifically and accurately, [22] has shown multiple disjoint certificate chains is an essential measurement of the assurance of name-to-key bindings.

In the rest of this paper we use *certificate graphs* as the model to represent the collection of certificates. A certificate graph G is a simple graph and 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 in a known correct name-to-key binding. [2]

From the above, if there are q vertex-disjoint paths in the certificate graph, each starting from user x's own key and all ending at the same target name-to-key binding, the *assurance* of the target name-to-key binding for x is defined to be q. That is, x would be able to say that the target name-to-key binding is resistant to attacks on less than q keys as shown in [22, 16]. The number of vertex disjoint paths can be computed by a maximum flow algorithm using an vertex-splitting technique[1]. There are various maximum flow algorithm available, such as those in [1].

we introduce the following notions in a certificate graph. Let $K_x(G)$ represent the set of all *correct* name-to-key bindings whose keys are reachable from vertex k_x(x's own key) in the certificate graph G. Let $K_x^T(G, l)$ represent the set of name-to-key bindings whose assurance are greater or equal to a predefined threshold l (for user x, binding x/k_x and the name-to-key bindings x directly certifies are correct, thus their assurance is set to be infinite). Then for user x the *robustness* of the certificate graph G at assurance l is defined to be

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

We define the total, or aggregate, robustness of a certificate graph G at assurance l for all users to be

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

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

[2] The set of known correct name-to-key bindings for each user is different. It is a general knowledge that user x's known correct name-to-key bindings include her own binding x/k_x and those that are directly certified by herself.

It is straightforward to see this robustness represents the percentage of name-to-key bindings whose assurance is at least l, thus robustness can range from a minimum of 0% (worst) to 100% (best).

We can now formally state the problem addressed by this paper. Given a directed certificate graph G and a value l representing a desired level of assurance, add the minimum number of arcs (i.e., additional certificates) to this graph so that robustness $\mathcal{A}(G, l)$ of whole certificate graph reaches 100%. It is worthwhile to minimize the number of additional certificates because it may reduce users' efforts that are required to yield the desired robust keyrings.

The next section presents a method for accomplishing the goal.

4 Enhancing Certificate Graphs

From the preceding discussion, assurance is enhanced when the number of vertex-disjoint paths between pairs of vertexes is increased. By Menger's theorem [12], vertex-disjoint paths correspond to graph connectivity. Specifically, a graph is q-connected iff every pair of vertexes is joined by at least q vertex-disjoint paths, or to say, there does not exist a set of q or fewer vertexes whose removal disconnects the graph [12]. The key requirement in achieving our goal is therefore to make the directed certificate graph G q-connected, where $q \geq l$ and l is the desired level of assurance for all users and all name-to-key bindings.

The problem of increasing the connectivity of a graph by adding a minimal number of edges is referred to as the *graph connectivity augmentation* problem. There are a number of solutions to this problem and a quite complete survey has been done both in [20] and [9]. Among the numerous solutions to this problem, [10, 3, 11] are the candidates to augment the vertex-connectivity in directed graphs. Unfortunately these may not be the practical solutions. For example, [10] relies on the ellipsoid method [19] which is well known to be impractical. [3] points out its practical efficient implementations require oracle choices. [11] requires a very expensive 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 required connectivity.

To efficiently solve this problem, we propose a heuristic for the graph augmentation problem. First we introduce some notations. Let $G = (V, E)$ be the given certificate graph. The *in-degree* $\Delta_i(v)$ of a vertex v in V is defined as the number of arcs in G whose terminal is v, while the *out-degree* $\Delta_o(v)$ of v is defined as the number of arcs in G whose initial is v. Let $\lambda_i^q(v)$ be the in-degree *q-deficiency* of vertex v, i.e., the minimum value to be added to make $\Delta_i(v)$ greater than or equal to q, and $\lambda_o^q(v)$ is similarly defined for the out-degree. The *minimum degree* of a graph G is represented as $\delta(G)$, and is defined as

$$\delta(G) = min\{\Delta_i(v), \Delta_o(v) | v \in V\}$$

The heuristic has two phases, and is shown 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 G may not be q-connected. To make G q-connected, algorithm 1 is used to detect critical pairs of vertexes whose connectivities are lower than q, then arcs are directly added between them.

Algorithm 1 q-connectivity algorithm

input: digraph G
output: vertex connectivity $q(G)$

1: select a vertex u with the minimum $\{\Delta_i(u) + \Delta_o(u)\}$
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: $q(G)$=min$\{c_1, c_2, c_3\}$

Algorithm 2 *graph connectivity augmentation algorithm*

input: $G(V,E)$, l
output: $G'(V,E')(l - connected)$

1: $G'(V,E') = G(V,E), q = l$
Phase 1
2: construct an arc list $L = \{(u,v) \mid u \in V, v \in V, (u,v) \notin E', (\Delta_o(u) < q \vee \Delta_i(v) < q)\}$ ordered by SUM$(\Delta_o(u) + \Delta_i(v))$
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 update L
13: **end while**

Due to space reasons, we only briefly explain algorithm 1 and 2. Details of these algorithms may be found in [13]. First we explain algorithm 1. Denote the vertex connectivity between two nodes u and v as $q(u,v)$. The vertex connectivity $q(u,v)$ between u and v can be computed by a maximum flow algorithm [1] using vertex-splitting techniques in time $O(|V||E| \log(\frac{|V|}{|E|}))$. The vertex connectivity $q(G)$ of G, is the minimum vertex 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 algorithm 1 as a faster means of computing $q(G)$. This algorithm follows some insights from an approach in [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's main purpose is to check $q(G)$ by checking the connectivity of a set of $2|V|+q^2$ pairs of vertexes. The connectivity of each pair of vertexes is checked using maximum flow algorithm and algorithm 1 requires to run maximum flow $2|V| + q^2$ times.

In algorithm 2, phase 1 is to make $\delta(G) \geq q$ by adding a minimum number of arcs. It is simple to see that in a q-connected graph, any vertex's in- and out-degree must be at least q (We refer to this as the $q - degree$ requirement). Thus the number of arcs needed to be added to make $\delta(G) \geq q$ is at least

$$max(\sum_{v \in V} |\lambda_i^q(v)|, \sum_{v \in V} |\lambda_o^q(v)|).$$

Our heuristic successfully achieve this goal by adding an arc from the vertex with the minimum out-degree to the vertex with the minimum in-degree each time. In phase 2, if algorithm 1 tells us G is already q-connected, then algorithm 2 can simply stop because the goal of making G q-connected is already completed. Otherwise, algorithm 1 would tell us which pair of vertexes has a connectivity lower than q, and our heuristic's choice is to simply add an arc directly between them.

The complexity of our heuristic is discussed as follows. Phase 1 only needs to check and sort each vertex's degree and requires $O(|V|log(|V|))$ time. In phase 2, suppose a total number of ε additional arcs are needed to make the graph q-connected, then algorithm 1 needs to be called ε times. Algorithm 1 itself needs to run maximum flow $2|V| + q^2$ times. The running time of phase 2 is $O(\varepsilon \cdot |V|^2|E| \log(\frac{|V|}{|E|}))$ which is also the running time for algorithm 2.

With this method, we can now address the goal of increasing the robustness of a certificate graph to 100%. For any arbitrary level of assurance l, the certificate graph is enhanced or augmented to be q-connected($q \geq l$) using the above method. The method is guaranteed to terminate, and the enhanced certificate graph is guaranteed to have an robustness of 100%. The arcs added by the method represent recommendations of certificates to add to the certificate graph. These recommendations may be directly sent to users by email[3] or be provided by running web service.

We have described heuristic methods for accomplishing our goal. In the next section, we examine the performance of the heuristic on actual web of trust: PGP keyrings.

[3] In PGP, each user's email address can be directly obtained from their certificates.

5 Experimental Results

The effectiveness and practicality of the proposed methods depend on the data to which it is applied. The best examples of web of trust that are widely available are PGP keyrings[7]. We therefore used these for purposes of experimental validation. The PGP keyrings we use in experiments are downloaded from PGP keyservers and the keyanalyze [23] tool was used to extract many strongly-connected components. The PGP keyrings used in this experiment are listed in table 1.

KR25	PGP keyrings with 67 keys and 209 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
SKR588	Synthetic keyrings simulating KR588
SKR1226	Synthetic keyrings simulating KR1226

Table 1. PGP keyrings used in this experiment

In the first set of experiments, we attempted to enhance the robustness of the actual keyrings KR588 and KR1226. We used algorithm 2 to enhance its robustness to 100% for varying levels of assurance l. We compared the performance with a method which randomly adds arcs to the graph. Any reasonable heuristic should of course perform better than the random method. We also computed a lower bound for the number of arcs which must be added by *any* method in order to achieve q-connectivity. It is very intuitive to see that by the q-degree requirement, the minimum number of arcs that must be added to make G q-connected is at least

$$max(\sum_{v \in V} |\lambda_i^q(v)|, \sum_{v \in V} |\lambda_o^q(v)|).$$

The performance of our method (total number of arcs after addition) relative to the random method, and to this lower bound, is shown in Table 2 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 is no greater than 0.7%.

To increase confidence that these results are representative, we also synthetically generated keyrings using the model of [6]. In this experiment, we generate two sets of keyrings, i.e. SKR588 and SKR1226. For each set of keyring, 50 synthetic keyrings were generated, based on the method of [6]. The results are shown in Table 3. Each value shown in this graph is the average of 50 synthetic instances and has been rounded to the nearest integer, 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.

PGP keyrings	Different methods	Total number of arcs needed at different levels of assurance								
		$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$
KR588	lower bound	5343	5523	5754	6029	6351	6706	7085	7484	7907
	our method	5349	5524	5754	6030	6352	6710	7086	7484	7907
	random method	8168	8877	9864	10505	11959	13991	14654	14901	15267
KR1226	lower bound	9147	9720	10403	11187	12034	12931	13846	14781	15744
	our method	9203	9769	10458	11223	12061	12960	13876	14827	15796
	random method	17369	20385	24679	24709	26717	31507	32350	33352	33372

Table 2. Number of certificates needed to reach 100% robustness for PGP keyrings

Synthetic keyrings	Different methods	Total number of arcs needed at different levels of assurance								
		$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$	$l=8$	$l=9$	$l=10$
SKR588	lower bound	5409	5598	5814	6077	6376	6713	7077	7469	7880
	our method	5409	5598	5815	6077	6377	6714	7078	7469	7881
	random method	9289	10255	11324	12376	13157	14219	15132	15841	16787
SKR1226	lower bound	9286	9874	10528	11284	12102	12973	13874	14808	15763
	our method	9287	9875	10530	11285	12103	12974	13875	14809	15763
	random method	18466	20989	23524	25923	27565	29635	31379	33448	35217

Table 3. Number of certificates needed to reach 100% robustness for synthetic keyrings

These experiments also yield useful insights into existing PGP keyrings. First, it may be seen that the "cost" (additional certificates) to achieve 100% robustness for a lower level of assurance is small (e.g., less than 10% additional certificates with no more than assurance 4). However, the overhead grows steadily, and for a higher level of assurance (e.g., greater than 10) it may be expected this overhead is substantial.

Our experiments also show the practical running time for algorithm 2. Practical experiment data tells us that algorithm 1 is called by algorithm 2 for only a few times. In all cases, ε is no greater than $\frac{|V|}{50}$. The heuristic is implemented in Java and C and the test environment is Linux Fedora Core 1 on a PC with a 2.4GHz Pentium IV processor and 512MB of memory. For the PGP keyring KR588 and KR1226, in all cases algorithm 2 requires less than 2 and 30 minutes to complete, respectively.

6 User Preferences (Constraints)

In our discussion of the problem we termed the addition of certificates *recommendations*. This is an acknowledgment that certificates are generated by people, frequently for personal reasons. 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 two approaches and show that our heuristic is feasible even under such constraints. For these purposes, we pro-

pose to model user preferences as *user compliance* and *buddy lists*, respectively. By user compliance, we mean some users may choose to follow only part of our recommendations instead of all of them. In buddy lists, a user would not issue certificates to users other than those on her buddy list. One consequence of user compliance and 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 a robustness of 100%. Experiments for these two types of user preferences have been done and are shown as below.

User compliance. To investigate how user compliance may affect the performance of our method, we ran the following experiments. First we applied algorithm 2 and produced all the recommendations that are needed to reach a 100% robustness at assurance levels from 2 to 10. With all these recommendations, a fraction t of users are randomly chosen to follow all of our recommendations and the rest users will randomly choose to follow a part(from 0% to 99.99%) of our recommendations. The results are shown in Figure 1 for different values of t.

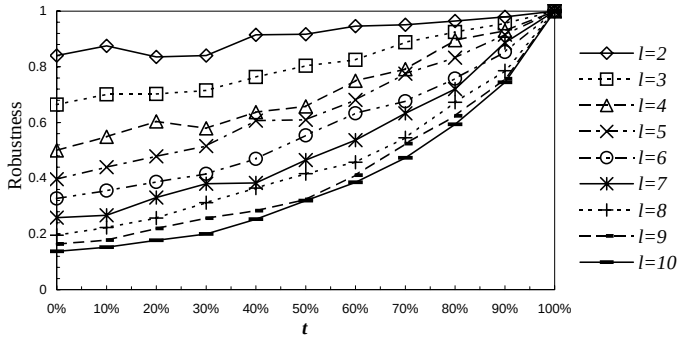

Fig. 1. Robustness of KR588 under different percentage of users' compliance

From this figure, we can see that the robustness is always higher than t for small levels of assurance($l < 5$). When l goes higher, the robustness can't keep up with t. One of the reasons is that for lower l, the number of recommendations is small, thus the users who don't comply with recommendations have little impairment on the resulting robustness of keyrings.

Buddy lists. Practical data of buddy lists, such as MSN, Yahoo Messenger and AOL Messenger are not available to us for obvious reasons. Instead, we artificially generated the buddy lists by utilizing the small world graph generator of [6]. Small world phenomena has been frequently found in social networks [26], World Wide Webs [15] and PGP web of trust [6]. Specifically, the users' buddy lists may be represented by a small world graph, where a vertex represents a user and an arc represents the buddy relationship. 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 robustness with our method at different levels of assurance from 2 to 10. Incorporating the constraint of buddy

lists 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. The results for algorithm 2 with buddy lists constraints are shown in Figure 2.

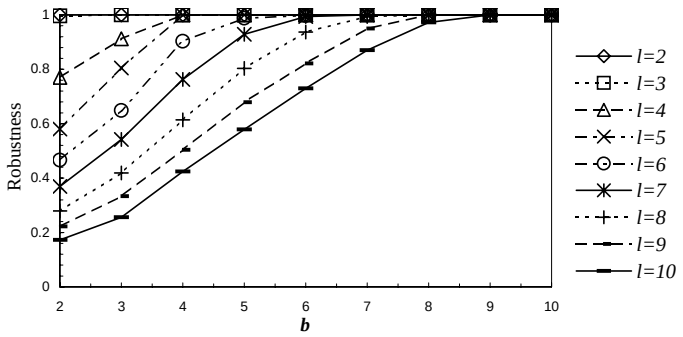

Fig. 2. Robustness of KR588 under different buddy lists

From this figure it can be seen that robustness increase roughly linearly with the increase of b, the minimum size of buddy list. The rate of increase of robustness decreases as b goes up. And a 100% robustness is achieved once b becomes the same as the level of assurance.

7 Certificate conflicts and suspect set

Certificate conflict was first caught attention in [22] and being analyzed in [14]. It is defined to be a pair of certificates in which the two name-to-key bindings have the same name but different keys. The definition is also extended to the case that each user may have multiple keys by associating a sequence number with each key[14]. Certificate conflicts may be used to detect malicious behavior and it is possible to construct *suspect set* to identify malicious users [14]. A suspect set is defined to be a set of names in which at least one belongs to a malicious user. Or to say, there is at least one malicious user in each suspect set. For details on how suspect sets may be constructed and used, we refer to [14].

One impact of our recommendation method is that it may be used for purposes of constructing suspect sets. To see how our recommendation method may be used to serve this purpose, we ran the following experiment on KR588. Note this experiment is just an illustration of our method's application to identify malicious users. No proof or data about the general effectiveness or optimality of the method are presented. First we randomly chose 9 colluding malicious users and each of them certifies an incorrect name-to-key binding to a victim newly added to the keyring. Second we apply our recommendation method to this keyring with $l = 10$. Note in this experiment, all malicious users are not supposed follow our recommendations simply because they may not like to do so. The results

are as following. Before applying our method, no certificate conflicts can be detected and no suspect set can be constructed. After applying our method, all users are able to detect the conflicts and over 99% users are able to construct suspect sets with size as small as three. If users may spend more efforts on the certificate conflicts, e.g. by contacting the victim, and find out which certificate contains the incorrect name-to-key binding, then over 99% users would be able to construct 9 suspect sets with size one. Or to say, almost all users would be able to identify all the 9 malicious users.

8 Conclusions and Future Work

In this paper we described how distributed web of trust can be made more robust against malicious attacks. The problem is modeled as a graph theory problem, i.e. increasing vertex connectivity of a certificate (directed) graph, which is a known problem with optimal but expensive solutions. Our heuristic for this problem runs fairly efficient and 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, we addressed the issue of user behavior and the constraints that adds. We investigate the impact of users' compliance and users' preference of our recommendations on the robustness of PGP keyrings. Moreover, the application of our method may also serve as a determent to malicious attacks because of user's ability to detect these attacks and identify 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.

References

1. R. Ahuja, T. Magnanti, and J. Orlin. *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J., 1993.
2. Tuomas Aura. On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, pages 14–26, Rockport, MA, June 1998. IEEE Computer Society Press.
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. Matt Blaze, Joan Feigenbaum, and Jack Lacy. 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. Darnell D. Pgp or pki? the future of internet security. *EDI Forum: The Journal of Electronic Commerce*, 12(1):59–62, 1999.

8. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. RFC 2693: SPKI certificate theory, September 1999.
9. A. Frank. Connectivity augmentation problems in network design. *Mathematical Programming: State of the Art 1994*, pages 34–63, 1994.
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. Andras Frank and Tibor Jordan. Directed vertex-connectivity augmentation. *Mathematical Programming*, 84(3):537–553, 1999.
12. Frank Harary. *Graph Theory*. Addison-Wesley, Reading, Mass., 1969.
13. Qinglin Jiang, Douglas S. Reeves, and Peng Ning. Certificate recommendations to improve robustness of webs of trust. Technical Report TR-2004-04, Department of Computer Science, N.C. State University, January 2004.
14. Qinglin Jiang, Douglas S. Reeves, and Peng Ning. Improving robustness of PGP by conflict detection. In *The Cryptographers Track at the RSA Conference 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 194–207. Springer, 2004.
15. Jon Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM Press, 2000.
16. R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *Proceedings of the Seventh USENIX Security Symposium*, 1998.
17. 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.
18. 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.
19. M.Grotschel, L.Lovasz, and A.Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169 –197, 1981.
20. Hiroshi Nagamochi and Toshihide Ibaraki. Graph connectivity and its augmentation: applications of ma orderings. *Discrete Appl. Math.*, 123(1-3):447–472, 2002.
21. M. Reiter and S. Stubblebine. Toward acceptable metrics of authentication. In *IEEE Symposium on Security and Privacy*, pages 10–20, 1997.
22. M. Reiter and S. Stubblebine. Resilient authentication using path independence. *IEEE Transactions on Computers*, 47(12), December 1998.
23. M. Drew Streib. Keyanalyze - analysis of a large OpenPGP ring. http://www.dtype.org/keyanalyze/.
24. Anas Tarah and Christian Huitema. Associating metrics to certification paths. In *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.
25. Int'l Telecommunications Union/ITU Telegraph & Tel. ITU-T recommendation X.509: The directory: Public-key and attribute certificate frameworks, Mar 2000.
26. D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440, 1998.
27. Philip Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, Mass., 1995.