Vulnerability Analysis of Certificate Graphs

Eunjin (EJ) Jung and Mohamed Gouda
1 University Station C0500
TAY 2.124 Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
Email:{ejung,gouda}@cs.utexas.edu

Abstract

A certificate system can be represented by a directed graph, called a *certificate graph*, where each node represents a user that has a public key and a private key and each edge (u, v) represents a certificate that is signed by the private key of u and contains the public key of v. Two types of damage can be done in a certificate graph when the private key of a node u in the graph is revealed to an adversary: explicit and implicit. The explicit damage is that the adversary can impersonate node u to other nodes in the graph (until it is known to other nodes that the private key of u is revealed). The implicit damage is that the adversary can impersonate nodes other than u to other nodes in the graph. In this paper, we define a metric called *vulnerability* that measures the scope of explicit and implicit damage that may occur in a certificate graph when the private key of a node in the graph is revealed to an adversary. Using this metric, we analyze the vulnerability of different classes of certificate graphs. For example, in the case of (m, k)-star certificate graphs, the vulnerability is $1 - \frac{k-1}{2mk}$, whereas in the case of (d, h)-tree certificate graphs, the vulnerability is approximately $1 - \frac{h}{d^h}$. For the same number of nodes, (m, k)-star certificate graphs can be made less vulnerable than (d, h)-tree certificate graphs. We present three algorithms that compute the vulnerability of an arbitrary certificate graph, and use these algorithms to show that certificate dispersal and stricter acceptance criteria reduce the vulnerability of certificate graphs.

1 Introduction

Consider a system where each user u has a public key $b.u$ and a private key $r.u$. When a user u wants to securely communicate with another user v, user u needs to find out the public key $b.v$ of user v (after an initial handshake using public and private keys, users can set up a shared key for further secure communication). In a large-scale distributed system, one can expect that a user may not know the public keys of all other users. Instead, a user can learn the public keys of other users via certificates when he or she needs to securely communicate with them. A certificate $\langle u, v, b.v\rangle r.u$ contains the identity of the issuer u, the identity of the subject v, and the public key of the subject $b.v$, and is signed by the private key $r.u$ of the issuer u. Any user who knows the public key of node u can learn the public key $b.v$ of node v via this certificate $\langle u, v, b.v\rangle r.u$. When user u needs to securely communicate with another user w, and does not have a certificate $\langle u, w, b.w\rangle r.u$, user u will seek a sequence of certificates from u to w to find the public key of node v. This sequence of certificates is called the *certificate chain* from u to w, where the first certificate is issued by u, and the subject of the last certificate is w, and each certificate in this chain can be verified using the public key of the previous certificate in the chain. For example, if user v has issued a certificate

$\langle v, w, b.w\rangle r.v$, then user u can verify each certificate in this chain $\langle u, v, b.v\rangle r.u\langle v, w, b.w\rangle r.v$ from u to w and obtain the public key of user w. Pretty Good Privacy (PGP) [17] is an example of such certificate systems.

The certificates issued by different users in a system can be represented by a directed graph, called the *certificate graph* of the system. Each node u in the certificate graph represents a user u with a public key $b.u$ and a private key $r.u$ in the system. Each directed edge from node u to node v in the certificate graph represents a certificate $\langle u, v, b.v\rangle r.u$. A certificate chain from a node u to a node v is a simple path from node u to node v in a certificate graph. For nodes u and v in a certificate graph G, if u wishes to securely send messages to v, then u seeks a path from u to v in G. (There are systems where u seeks a set of paths from u to v, which will be discussed in Section 8.)

In a certificate graph, two types of damage can occur when the private key $r.u$ of a node u is revealed to an adversary: explicit and implicit. The explicit damage is that the adversary can impersonate node u to other nodes until it is known to other nodes that the private key $r.u$ of u is revealed to the adversary. The implicit damage is that the adversary can impersonate nodes other than u to other nodes in the system by signing forged certificates with the revealed private key $r.u$ of node u.

As an example, consider the certificate graph in Fig. 1. If node a wishes to send a secure message to node g in this certificate graph, then node a needs to find a certificate chain from node a to node g. In the certificate graph in Fig. 1, there is one certificate chain from node a to node g, $(a, d), (d, e), (e, g)$.

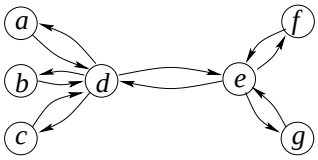

Figure 1: An example of a certificate graph

Assume that the private key $r.d$ of node d is revealed to an adversary. The adversary can encrypt and decrypt messages using $r.d$ to impersonate node d to any other node in the graph. This impersonation of node d is explicit damage. Assume that node a does not know that $r.d$ is revealed. The adversary can create a new public and private key pair, $b.g'$ and $r.g'$, and sign a forged certificate $\langle d, g, b.g'\rangle r.d$ with the revealed private key $r.d$ of node d. Denote this forged certificate as (d, g'). The certificate chain $(a, d), (d, g')$ shows the public key $b.g'$ created by the adversary to node a as if it belonged to node g. This impersonation of node g is implicit damage.

The explicit and implicit damage that can be brought into a certificate graph when the private key of a node u is revealed to an adversary is called the *vulnerability* of node u. For example, if the private key $r.d$ of node d in Fig. 1 is revealed to an adversary, then the adversary can impersonate node d to all other nodes in the graph without forging any certificates. In addition to impersonating node d, the adversary can impersonate nodes a, b, c to nodes e, f, g by signing forged certificates (d, a'), (d, b'), and (d, c') with the revealed private key $r.d$ of node d. Also, the adversary can impersonate nodes e, f, g to nodes a, b, c by forging certificates (d, e'), (e', f'), and (e', g').

We have identified a metric to measure the damage from this type of attacks. We call this metric *"vulnerability"* of certificate graphs. As discussed in detail in this paper, a metric of the vulnerability of certificate graphs is useful in several applications. First, this metric can be used to determine which certificate graphs are less vulnerable and which ones are more vulnerable. Second,

this metric can be used to determine which criteria for accepting public keys from certificate chains are better. Third, this metric can be used to balance between the resilience against impersonation attacks and storage cost.

In the following sections, we formally define the vulnerability metrics of nodes and of certificate graphs, and present theorems that show vulnerabilities of several certificate graphs with different requirements. Also, we present three algorithms to compute the vulnerability of an arbitrary certificate graph. Using these algorithms, we investigate the effect of graph topology, certificate dispersal, and acceptance criteria on the vulnerability of certificate graphs. Then we discuss the vulnerability when many private keys are revealed to an adversary. We present a brief summary of related work and end with concluding remarks.

2 Vulnerability of Certificate Graphs

Let G be a certificate graph and d be a node in G. We assume that each node in G stores the certificates it issues. Assume that the private key $r.d$ of node d is revealed to an adversary. The adversary can use the revealed private key to encrypt and decrypt any messages as if the adversary were node d, and so it can impersonate node d to all other nodes from which there are certificate chains to d. Also, the adversary can use $r.d$ to impersonate a node dst, other than d, to another node src, also other than d in G, by performing the following three steps.

i. The adversary creates a private key $r.dst'$ and its corresponding public key $b.dst'$. Later, the adversary will pretend that these keys are the public and private keys of node dst.

ii. The adversary uses the revealed private key $r.d$ of node d to issue a forged certificate $\langle d, dst, b.dst' \rangle r.d$. This forged certificate is denoted (d, dst').

iii. The adversary provides node src with the certificate chain that consists of a chain of correct certificates from src to d and the forged certificate (d, dst'). From this chain, node src can wrongly deduce that the public key $b.dst'$ created by the adversary is the public key of node dst. Any message sent by the adversary that is encrypted with the matching private key $r.dst'$ will be authenticated by node src as if it were sent by node dst.

Note that this scenario of the adversary would work only if G has a certificate chain from src to d that does not contain any certificate issued by dst and G has no certificate (src, dst).

The next theorem states a necessary and sufficient condition for an adversary to impersonate node dst to another node src in a certificate graph where the private key $r.d$ of some node d is revealed to an adversary.

Theorem 1 *Let G be a certificate graph and src and dst be any two distinct nodes in G. Let d be a node in G whose private key $r.d$ is revealed to an adversary. The adversary can impersonate node dst to node src if and only if $src \neq d$, G has a certificate chain from src to d that does not contain any certificates issued by node dst, and one of the following two conditions holds.*

i. $dst = d$, or

ii. G has no certificate (src, dst)

Proof for if If $dst = d$, then the adversary can use the revealed private key $r.d$ of node d to encrypt and decrypt any message as if it were node d and impersonate node dst to node src. If $dst \neq d$, then G has no certificate (src, dst), so src does not know the correct public key of dst. Now the adversary can sign a forged certificate (d, dst') with the revealed private key $r.d$ of d. There is a certificate chain from src to d that does not contain any certificates issued by dst, so the adversary can add the forged certificate (d, dst') to the correct certificate chain from src to d and present the certificate chain from src to dst' to node src. If node src does not know that the private key of node d is revealed to the adversary, then src will not notice that the certificate (d, dst') is forged and accept the public key in (d, dst') as the valid public key of dst.

Proof for only if In order to prove the "only if" part, we prove the contraposition of the "only if" part, i.e. if any of the following three conditions holds, then the adversary cannot impersonate dst to src:

i. $src = d$

ii. Any certificate chain from src to dst in G, if it exists, contains a certificate issued by dst.

iii. $dst \neq d$ and G has certificate (src, dst).

■

Let G be a certificate graph and d be a node in G. Assume that the private key $r.d$ of node d is revealed to an adversary. The *vulnerability of node d*, denoted $V(d)$, is the number of node pairs (src, dst) where the adversary can impersonate node dst to node src divided by the number of node pairs (src, dst) where $src \neq dst$ and $src \neq d$ in G. More formally,

$$V(d) = \frac{|IMP(d)|}{(n-1)^2},$$

where $IMP(d) = \{(src, dst)|$ the adversary can impersonate dst to src using $r.d\}$ and n is the number of nodes in G.

The following theorem gives tight upper and lower bounds on the vulnerability of a node in a certificate graph.

Theorem 2 *i. For a node d in any certificate graph G, we have*

$$\frac{|\{src|G \text{ has a chain from } src \text{ to } d\}|}{(n-1)^2} \leq V(d) \leq 1$$

ii. There exists node d in some certificate graph G, where

$$V(d) = 1$$

iii. There exists node d in some certificate graph G, where

$$V(d) = \frac{|\{src|G \text{ has a certificate chain from } src \text{ to } d\}|}{(n-1)^2}$$

Proof of i By Theorem 1, when a private key of node d is revealed to an adversary, the adversary cannot impersonate a node dst to another node src if $src = dst$ or $src = d$. For a given node d, the most number of node pairs (src, dst) such that $src \neq dst$ and $src \neq d$ is $(n-1)^2$. Therefore, the upper bound of $V(d)$ is 1. Also, since the adversary knows $r.d$, the adversary can always impersonate node d to every node that has a certificate chain from itself to d. (This is the scope of explicit damage.) Therefore, the number of node pairs (src, d) where G has a certificate chain from src to d divided by $(n-1)^2$ is the lower bound.

Proof of ii Consider the certificate graph in Fig. 2. When the private key of the center node is revealed to an adversary, the adversary can impersonate any node dst to any other node src, where src is not the center node. There are 8 nodes that can be src, and for each src node among them, there are 8 other nodes that can be impersonated to src. Therefore, the number of node pairs (src, dst) where the adversary can impersonate dst to src is $8 \times 8 = 64$, and $n = 9$. Therefore, the vulnerability of the center node is 1.

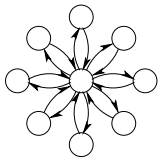

Figure 2: The $(8, 1)$-star certificate graph

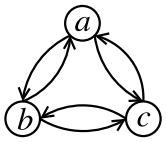

Figure 3: An example of fully connected certificate graph

Proof of iii In the certificate graph in Fig. 3, every node has issued certificates to all other nodes in the graph. If the private key of node c is revealed to an adversary, the adversary can impersonate only node c to nodes a and b, since node a already knows the correct public key of node b in the certificate (a, b) and node b knows the correct public key of node a in the certificate (b, a). So the vulnerability of node c is $\frac{2}{2^2}=\frac{1}{2}$, which meets the lower bound. In fact, the vulnerability of any node in a fully connected certificate graph meets the lower bound. ■

Let G be a certificate graph, then the vulnerability of graph G, denoted $V(G)$, is defined as follows:

$$V(G) = \max_{d \in G} V(d)$$

3 Vulnerability of Special Certificate Graphs

In this section, we give three theorems that show the vulnerability of three special classes of certificate graphs: n-loops, (m, k)-stars, and (d, h)-trees. In many certificate systems, for example PGP, certificate graphs are not planned in advance and certainly not designed. Rather, they are developed in an ad-hoc manner depending on which users decide to issue certificates for which other users. However, if we do have the luxury of planning and designing certificate graphs, then we can choose the best among these special classes according to the system requirements. n-loop certificate graphs are useful when the certificate graph needs to be strongly-connected but the number of certificates needs to be minimized. (m, k)-star certificate graphs are useful when a trusted certificate authority (center node) is available. (d, h)-tree certificate graphs are useful in hierarchical systems.

The following three theorems compute the vulnerabilities of three special classes of certificate graphs. The theorems show that n-loop certificate graphs are less vulnerable than $(m, 2)$-star certificate graphs for $n \geq 4$. On the other hand, $(2, h)$-tree certificate graphs are less vulnerable than n-loop certificate graphs for $n > 10$. The comparison results are discussed in more detail in the end of this section.

An n-loop certificate graph is a certificate graph that has n nodes arranged in a unidirectional ring. Fig. 4 shows the 8-loop certificate graph.

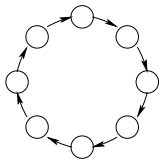

Figure 4: The 8-loop certificate graph

Theorem 3 *The vulnerability of an n-loop certificate graph is* $1 - \frac{n-2}{2(n-1)}$.

Proof Label each node $0 \cdots n-1$. Assume that the private key of node j is revealed to an adversary. The adversary can impersonate node k to node i if $k = j$, or if $\nexists(i, k)$ and there is a path from node i to node j that does not contain node k. Therefore, to node $j-1$, the adversary can impersonate nodes $j, j +_n 1, \cdots, j +_n (n-2)$. To node $j-2$, the adversary can impersonate nodes $j, j +_n 1, \cdots, j +_n (n-3)$. After considering each node, the number of (src, dst) pairs in which the adversary can impersonate node dst to node src is $\frac{n(n-1)}{2}$. The vulnerability of node j is $\frac{n(n-1)}{2(n-1)(n-1)} = 1 - \frac{n-2}{2(n-1)}$. This holds for any node j in this graph, so the vulnerability of an n-loop certificate graph is $1 - \frac{n-2}{2(n-1)}$. ■

An (m, k)-star certificate graph is a certificate graph that consists of m unidirectional rings that share one center node and each ring has k unshared nodes. Fig. 5 shows the $(4, 2)$-star certificate graph.

Theorem 4 *The vulnerability of an* (m, k)*-star certificate graph is* $1 - \frac{k-1}{2mk}$.

Proof The vulnerability of a graph is the maximum vulnerability of every node in the graph. In an (m, k)-star certificate graph, the center node has the highest vulnerability. Now let us compute the vulnerability of the center node. Label the k nodes in a satellite ring from $1 \cdots k$ and the center node as node 0. There is an edge from node i to node $i +_{k+1} 1$, where $0 \leq i \leq k$. When the private key of the center node is revealed to an adversary, the adversary can impersonate to node 1 any node in the graph except for the nodes $2 \cdots k$ in the same satellite ring. To node 2, the adversary can impersonate any node in the graph except for the nodes $3 \cdots k$ in the same satellite ring. As

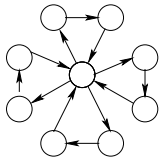

Figure 5: The $(4, 2)$-star certificate graph

a result, the adversary can impersonate $\sum_{i=1}^{k}(mk-(k-i))$ pairs for each satellite ring. So the vulnerability of the center node is

$$\begin{aligned} V(center) &= \frac{1}{(mk)^2}(m\sum_{i=1}^{k}(mk-(k-i))) \\ &= \frac{1}{(mk)^2}\left(mk(mk-k)+\frac{mk(k+1)}{2}\right) \\ &= \frac{1}{mk}\left((mk-k)+\frac{k+1}{2}\right) \\ &= \frac{2mk-2k+k+1}{2mk} \\ &= \frac{2mk-k+1}{2mk} \\ &= 1-\frac{k-1}{2mk} \end{aligned}$$

Therefore, the vulnerability of an (m,k)-star certificate graph is $1-\frac{k-1}{2mk}$. ■

A (d,h)-tree certificate graph is a complete tree certificate graph with degree d and height h, where there is an edge from each parent node to each of its children nodes and an edge from each child node to its parent node. Fig. 6 is an example of a (d,h)-tree certificate graph, where $d=3$ and $h=2$.

Theorem 5 *The vulnerability of a (d,h)-tree certificate graph is $1-\frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2}-\frac{d}{(d-1)(n-1)}-\frac{d^2}{(d-1)(n-1)^2}$, approximately $1-\frac{h}{d^h}$.*

Proof The vulnerability of a graph is the maximum of vulnerability of all nodes in the graph. In a (d,h)-tree certificate graph, the root node has the highest vulnerability. The vulnerability of the root node can be computed as follows. Consider a node i in level h. When the private key of the root node is revealed to an adversary, the adversary can impersonate any node to node i except the $(h-1)$ nodes on the certificate chain from node i to the root node. On the other hand, for a node j in level $h-1$, the adversary can impersonate any node to node j except its d children nodes and the $(h-2)$ nodes on the certificate chain from node j to the root node. So, the adversary can impersonate $(n-1-(h-2+d))$ nodes to node j. Similarly, for a node in level l, where $l<h$, the adversary can impersonate $(n-1-(l-1+d))$ nodes to the node. As a result, the vulnerability

of the root node is :

$$
\begin{aligned}
V(root) &= \frac{1}{(n-1)^2}\left(\sum_{i=1}^{h-1} d^i(n-1-(i-1+d)) + d^h(n-1-(h-1))\right) \\
&= \frac{1}{(n-1)^2}\left((n-d)\sum_{i=1}^{h-1} d^i - \sum_{i=1}^{h-1} id^i + (n-h)d^h\right) \\
&= \frac{1}{(n-1)^2}\left(n\sum_{i=1}^{h} d^i - \sum_{i=1}^{h-1} d^{i+1} - \sum_{i=1}^{h} id^i\right) \\
&= \frac{1}{(n-1)^2}\left(n(n-1) - \frac{d^2(d^{h-1}-1)}{d-1} - \frac{hd^{h+1}-n+1}{d-1}\right) \\
&= \frac{n}{n-1} - \frac{d^{h+1}-d^2+hd^{h+1}-n+1}{(d-1)(n-1)^2} \\
&= 1 - \frac{1}{n-1} - \frac{d^{h+1}-d^2+hd^{h+1}-n+1}{(d-1)(n-1)^2} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{1}{n-1} - \frac{d^2}{(d-1)(n-1)^2} - \frac{1}{(d-1)(n-1)} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{d-1+1}{(d-1)(n-1)} - \frac{d^2}{(d-1)(n-1)^2} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{d}{(d-1)(n-1)} - \frac{d^2}{(d-1)(n-1)^2} \\
&\simeq \{\text{since } n \text{ is large}\}\, 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} \\
&\simeq \{\text{since } n = \frac{d^{h+1}-1}{d-1} \simeq d^h\}\, 1 - \frac{d^{h+1}(1+h)}{(d-1)(d^h)^2} \\
&= \frac{1}{(n-1)^2}\left(n(n-1) - \frac{d^2(d^{h-1}-1)}{d-1} - \frac{hd^{h+1}-n+1}{d-1}\right) \\
&= \frac{n}{n-1} - \frac{d^{h+1}-d^2+hd^{h+1}-n+1}{(d-1)(n-1)^2} \\
&= 1 - \frac{1}{n-1} - \frac{d^{h+1}-d^2+hd^{h+1}-n+1}{(d-1)(n-1)^2} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{1}{n-1} - \frac{d^2}{(d-1)(n-1)^2} - \frac{1}{(d-1)(n-1)} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{d-1+1}{(d-1)(n-1)} - \frac{d^2}{(d-1)(n-1)^2} \\
&= 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} - \frac{d}{(d-1)(n-1)} - \frac{d^2}{(d-1)(n-1)^2} \\
&\simeq \{\text{since } n \text{ is large}\}\, 1 - \frac{d^{h+1}+hd^{h+1}}{(d-1)(n-1)^2} \\
&\simeq \{\text{since } n = \frac{d^{h+1}-1}{d-1} \simeq d^h\}\, 1 - \frac{d^{h+1}(1+h)}{(d-1)(d^h)^2} \\
&\simeq \{\text{since } \frac{h+1}{d-1} \simeq \frac{h}{d}\}\, 1 - \frac{hd^{h+1}}{d(d^h)^2} \\
&= 1 - \frac{h}{d^h}
\end{aligned}
$$

Therefore, the vulnerability of a (d, h)-tree certificate graph is approximately $1 - \frac{h}{d^h}$. ■

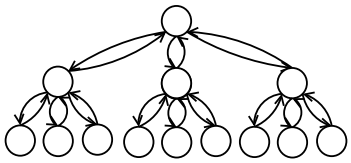

Figure 6: The $(3, 2)$-tree certificate graph

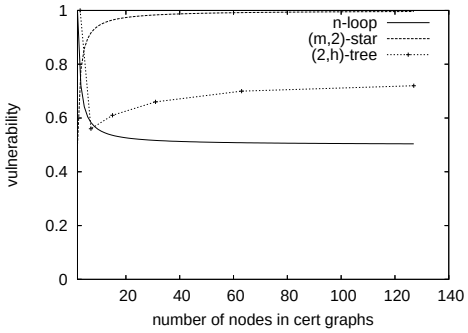

Figure 7: Comparison of three special graphs

Fig. 7 shows the vulnerabilities of three special certificate graphs, n-loops, $(m, 2)$-stars, and $(2, h)$-trees as functions of the number of nodes in each graph. From this graph, it is clear that n-loops are less vulnerable than $(m, 2)$-stars and $(2, h)$-trees. This metric of vulnerability can be used to show which certificate graph is less vulnerable.

4 Vulnerability of Arbitrary Certificate Graphs

In the previous section we computed the vulnerability of three special classes of certificate graphs. We now present Algorithm 1 that computes the vulnerability of an arbitrary certificate graph.

By Theorem 1, if G has a path from node src to node d that does not contain node dst, then the adversary can impersonate dst to src when the private key of node d is revealed to it (and G has no certificate (src, dst)). To find every node src that has a path to node d that does not contain node dst, Algorithm 1 removes node dst and its incoming and outgoing edges from G and sees which nodes are still connected to d. Consider the example certificate graph G in Fig. 1. In Fig. 8(a), node a and its incoming and outgoing edges are removed from G. There are paths from nodes b, c, e, f, g to node d in Fig. 8(a). Therefore, if the private key of node d is revealed to an adversary, then the adversary can impersonate node a to nodes b, c, e, f, g. On the other hand, without node e and its incoming and outgoing edges, there are no paths from nodes f, g to node d as shown in Fig. 8(b). Therefore, when the private key of d is revealed to an adversary, the adversary cannot impersonate node e to nodes f, g.

Figure 8: Computing vulnerability of the example graph

For a given certificate graph G, Algorithm 1 computes a transitive closure C_{dst} without using any incoming and outgoing edges of node dst for each node dst in G (lines 3-4). C_{dst} contains an edge (src, d) if and only if there is a path from src to d that does not contain dst and G has no certificate (src, dst) (line 5). In other words, if there is an edge (src, d) in C_{dst}, then an adversary can impersonate dst to src when the private key of node d is revealed to the adversary.

ALGORITHM 1 : Vulnerability of a certificate graph

INPUT: a certificate graph G with n nodes
OUTPUT: vulnerability of G

STEPS:
1: **for** $dst = 0$ **to** $n - 1$
2: $C_{dst} := G$
3: **remove** all the incoming and outgoing edges of node dst from C_{dst}
4: C_{dst} := transitive closure of C_{dst}
5: **if** G has an edge (src, dst), **then remove** (src, dst) from C_{dst}
6: **endfor**
7: C := transitive closure of G
8: **for** $d = 0$ **to** $n - 1$
9: $V(d) := \sum_{dst \in G}$(the in-degree of node d in C_{dst})
+ the in-degree of node d in C
10: **endfor**
11: **return** $\max_{d \in G} \frac{V(d)}{(n-1)^2}$

To compute the vulnerability of a node d in G, Algorithm 1 finds all the node pairs (src, dst) in G such that G has a path from src to d that does not contain dst and has no certificate (src, dst). For each node dst in G, the in-degree of node d in the transitive closure C_{dst} is the number of node pairs (src, dst) in G that satisfies the condition. So the sum of the in-degree of node d in the transitive closure C_{dst} for each node dst in G shows the scope of the implicit damage of the revealed private key of node d.

In the example certificate graph G in Fig. 1, when the private key of d is revealed to an adversary, the adversary can impersonate node d to any other user in G. To compute this explicit damage of the revealed private key of node d, Algorithm 1 also computes a transitive closure C of G (line 7). C contains an edge (src, d) if and only if there is a path from src to d in G. In other words, if there is an edge (src, d) in C, then the adversary can impersonate d to src using the revealed private key of node d. Therefore, the in-degree of node d in the transitive closure C of G shows the scope of the explicit damage of the revealed private key of node d.

Using these transitive closures, Algorithm 1 computes the vulnerability of each node d in a given certificate graph G, and then returns the maximum as the vulnerability of the certificate graph.

In this algorithm, the most expensive step is line 4. The cost of computing a transitive closure of a certificate graph with n nodes is $O(n^3)$, and we need to compute $(n + 1)$ transitive closures. Therefore, the complexity of this algorithm is $O(n^4)$.

5 Effect of Topology on Vulnerability

The vulnerability of a certificate graph is affected by the topology of the graph. For example, the $(4, 2)$-star certificate graph in Fig. 5 has vulnerability $\frac{15}{16}$, whereas the $(8, 1)$-star certificate graph in Fig. 2 has vulnerability 1. Therefore, these two certificate graphs, despite having the same number of nodes and the same connectivity, have different vulnerabilities.

In Fig. 9, we show the effect of topology on vulnerability of star certificate graphs. Theorem 4

gives the vulnerability of (m, k)-star certificate graphs. However, if we keep the same number of nodes in the star certificate graph but change the value of k, not every satellite ring can have exactly k nodes. We put k nodes in as many rings as possible, and leave the remaining nodes in the last ring. We ran Algorithm 1 on the star certificate graphs with 100 nodes where k, the maximum number of nodes in each satellite ring, changes from 1 to 99. Fig. 9 shows that the vulnerability decreases as k increases.

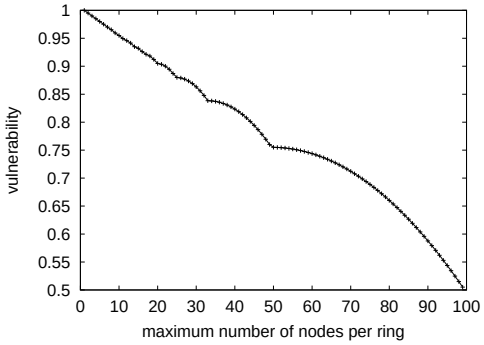

Figure 9: Vulnerability of Star Certificate Graphs

In Fig. 10, we show the effect of topologies on vulnerability of tree certificate graphs. Theorem 5 gives the vulnerability of (d, h)-tree certificate graphs. However, if we keep the same number of nodes in the certificate graph but change the value of d, the resulting tree may not be complete. In those trees, we pack the leaf nodes to the left. We ran Algorithm 1 on the tree certificate graphs with 100 nodes where d, the degree of tree, changes from 2 to 99. Fig. 10 shows that the vulnerability increases as d increases.

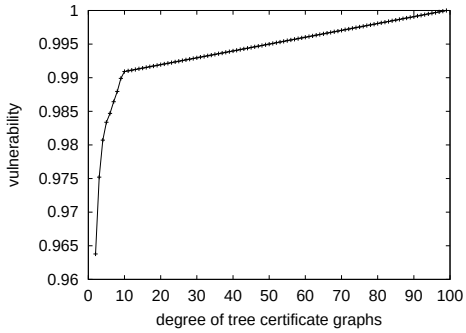

Figure 10: Vulnerability of Tree Certificate Graphs

6 Effect of Dispersal on Vulnerability

In a certificate graph where certificate chains are used to find a public key, nodes may store a few certificates in their local storage to expedite the search for a public key [6, 7, 16]. In particular, *certificate dispersal* D of a certificate graph G [7, 16] assigns a set of certificates $D.u$ to each node u so that if G has a certificate chain from node u to node v, then $D.u \cup D.v$ contains all the certificates in the certificate chain. If certificate dispersal is applied, then when a node u wishes to securely communicate with a node v, then node u will look for a public key of node v in $D.u$ first before it sends out a query to node v for more certificates. Therefore, if node u already has a

$D.a$	$= \{(a,d),(d,a)\}$,
$D.b$	$= \{(b,d),(d,b)\}$,
$D.c$	$= \{(c,d),(d,c)\}$,
$D.d$	$= \{\}$,
$D.e$	$= \{(d,e),(e,d)\}$,
$D.f$	$= \{(d,e),(e,d),(e,f),(f,e)\}$,
$D.g$	$= \{(d,e),(e,d),(e,g),(g,e)\}$

Table 1: Optimal dispersal of certificate graph in Fig. 1

$D.a$	$= \{(a,d),(d,b),(d,c),(d,e),(e,f),(e,g)\}$,
$D.b$	$= \{(b,d),(d,a),(d,c),(d,e),(e,f),(e,g)\}$,
$D.c$	$= \{(c,d),(d,a),(d,b),(d,e),(e,f),(e,g)\}$,
$D.d$	$= \{(d,a),(d,b),(d,c),(d,e),(e,f),(e,g)\}$,
$D.e$	$= \{(e,d),(d,a),(d,b),(d,c),(e,f),(e,g)\}$,
$D.f$	$= \{(f,e),(e,d),(d,a),(d,b),(d,c),(e,g)\}$,
$D.g$	$= \{(g,e),(e,d),(d,a),(d,b),(d,c),(e,f)\}$

Table 2: Full-tree dispersal of certificate graph in Fig. 1

certificate that has node v as the subject of the certificate, the adversary cannot impersonate node v to node u by issuing forged certificates. In other words, the vulnerability of a certificate graph is not only determined by the topology of the certificate graph, but also affected by the dispersal of the certificate graph. For example, the optimal certificate dispersal of the certificate graph in Fig. 1 is shown in Table 1.

When no dispersal is deployed. If the private key of node d is revealed to an adversary, then the adversary can impersonate nodes d, e, f, g to nodes a, b, c, and impersonate nodes a, b, c, d to nodes e, f, g. However, when we assign all the certificates in an outgoing spanning tree rooted at node x to the set $D.x$, if the private key of node d is revealed to an adversary, then the adversary can impersonate only node d to all other nodes, so there can be no implicit damage to the graph. Such dispersal for the certificate graph in Fig. 1 is shown in Table 2. Theorem 1 is modified here to take the effect of dispersal into consideration.

Theorem 6 *Let G be a certificate graph and src and dst be any two distinct nodes in G. Let D be a dispersal of G and d be a node in G whose private key $r.d$ is revealed to an adversary. The adversary can impersonate node dst to node src if and only if $src \neq d$, G has a certificate chain from src to d that does not contain any certificate issued by node dst, and one of the following two conditions holds.*

i. $dst = d$, *or*

ii. $D.src \not\ni (k, dst), k \in G$

Algorithm 2 is modified from Algorithm 1 to include the effect of dispersal in the evaluation of vulnerability. If node src has a certificate (x, dst) due to dispersal for any user x, then no adversary can impersonate dst to x with the revealed private key of any user y. Specifically, after line 4 in Algorithm 1, the following line is added: if any $D.src$ has an edge (x, dst), then remove all the edges (src, y) from C_{dst}.

The graph in Fig. 11 shows how much vulnerability is reduced by the optimal certificate dispersal of tree certificate graphs. The tree certificate graphs have 100 nodes and the degree changes from 2 to 99. The result without dispersal is the same as Fig. 10.

The cost of certificate dispersal is defined as the average number of certificates stored in each node. Note that the cost of the optimal dispersal of tree certificate graphs decreases, as shown in Fig. 12, whereas the vulnerability increases, as the degree of the tree increases. The x-axis of the graph in Fig. 12 is same as Fig. 11, and the y-axis shows the optimal dispersal cost. There is a clear trade-off between the vulnerability and the optimal dispersal cost of tree certificate graphs.

ALGORITHM 2 : Vulnerability with certificate dispersal

INPUT: a certificate graph G with n nodes and a dispersal D of G
OUTPUT: vulnerability of G

STEPS:
1: **for** $dst = 0$ **to** $n - 1$
2: $C_{dst} := G$
3: **remove** all the incoming and outgoing edges of node dst from C_{dst}
4: C_{dst} := transitive closure of C_{dst}
5: **if** any $D.src$ has an edge (x, dst), **then remove** all the edges (src, y) from C_{dst}
6: **endfor**
7: C := transitive closure of G
8: **for** $d = 0$ **to** $n - 1$
9: $V(d) := \sum_{dst \in G}$(the in-degree of node d in C_{dst})
+ the number of edges (src, d) in C for any node src in G
10: **endfor**
11: **return** $\max_{d \in G} \frac{V(d)}{(n-1)^2}$

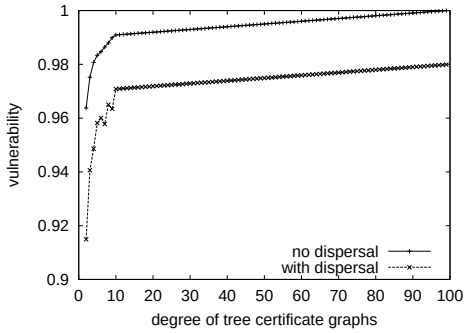

Figure 11: Vulnerability of Tree Certificate Graphs with Optimal Dispersal

The trade-off between the dispersal cost and the vulnerability in general is fairly straightforward, since a higher dispersal cost means that nodes know more correct public keys, corresponding to more nodes that the adversary will not be able to impersonate. However, the trade-off between the dispersal cost of the optimal dispersal and the vulnerability shown here suggests that the certificate graph topology must be carefully chosen to reach the right balance between the performance overhead (i.e. the size of local storage for dispersed certificates) and the resilience against attacks (i.e. the vulnerability). A graph with little vulnerability may be resource-intensive, which is suitable for high assurance networks. On the other hand, a graph with limited storage may prefer a certificate graph topology with small dispersal cost and suffer a higher exposure to impersonation attacks.

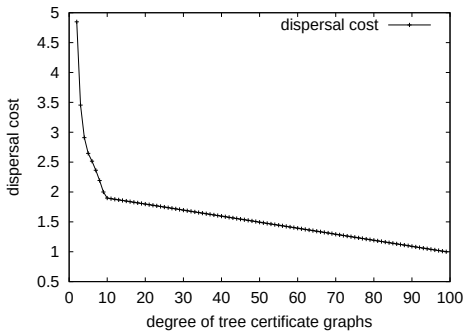

Figure 12: Optimal Dispersal Cost of Tree Certificate Graphs

7 Effect of Acceptance Criteria on Vulnerability

To reduce the implicit damage of a revealed private key of a node, many researchers proposed to use some acceptance criteria to verify the validity of the public key of the destination node of the certificate chain [2, 8, 9, 10, 11, 12, 14, 17]. Some of the results are described in Section 9. Most of these criteria can be modelled as a function that takes a set of certificate chains as an input and outputs a yes/no answer. A node u, who wants to find the public key of another node v, will find a set of certificate chains from u to v. Node u can give this set as an input to the acceptance criteria function, and if the output answer is yes, then the public key of node v in the certificate chain will be accepted by node u as valid. Theorem 1 is modified here to take the acceptance criteria into consideration.

Theorem 7 *Let G be a certificate graph and src and dst be any two distinct nodes in G. Let d be a node in G whose private key r.d is revealed to an adversary. The adversary can impersonate node dst to node src if and only if $src \neq d$ and one of the following two conditions holds.*

i. $d = dst$ and G has a set of certificate chains from src to dst that satisfies the acceptance criteria of G, or

ii. $\nexists(src, dst)$ and the set of certificate chains where each chain in the set consists of a correct certificate chain from src to d, that does not contain any certificate issued by node dst, and a forged certificate (d, dst'), satisfies the acceptance criteria of G.

A simple acceptance criteria is to limit the length of certificate chains that can be used. In fact, this acceptance criteria is implemented in the current PGP system as the parameter CERT_DEPTH. Algorithm 3 below computes vulnerability of certificate graphs in the case where this acceptance criteria is used. To explain Algorithm 3, we need to define the concept of k-closure.

A k-closure of a graph G is a directed graph that has the same number of nodes in G, and this graph has an edge (src, dst) if and only if there is a directed path of length at most k from src to dst in G. Note that 1-closure of G is G itself, and 0-closure of G is a graph with the same nodes in G but does not contain any edges.

Algorithm 3 takes a certificate graph G and the limit k on chain length as input and compute (k-1)-closures for each node dst, so that the adversary can add a forged certificate to the existing chain and the resulting chain will satisfy the limit k on chain length.

The graphs in Figs. 13-14 show how vulnerability changes as we apply different k as the limit on chain length. As k increases, the vulnerability increases. In Fig. 13, each star certificate graph has 100 nodes and 10 satellite rings, and the maximum number of nodes in a satellite ring is 10.

ALGORITHM 3 : Vulnerability with limit k on chain length

INPUT: a certificate graph G with n nodes and a limit k on chain length
OUTPUT: vulnerability of G

STEPS:
1: **for** $dst = 0$ **to** $n - 1$
2: $C_{dst} := G$
3: **remove** all the incoming and outgoing edges of node dst from C_{dst}
4: $C_{dst} := (k-1)$-closure of C_{dst}
5: **if** G has an edge (src, dst), **then remove** (src, dst) from C_{dst}
6: **endfor**
7: $C := k$-closure of G
8: **for** $d = 0$ **to** $n - 1$
9: $V(d) := \sum_{dst \in G}$(the in-degree of node d in C_{dst})
+ the in-degree of node d in C
10: **endfor**
11: **return** $\max_{d \in G} \frac{V(d)}{(n-1)^2}$

We changed the value of limit k on chain length from 1 to 11, since the longest chain that the adversary will use from the original certificate graph is 10. (The longest chain from a node in a satellite ring to the center node is 10.) After 10, the vulnerability is same as that in Fig. 9, shown as a dotted line here.

In Fig. 14, each tree certificate graph has 100 nodes and the degree is 2. Since the root node has the maximum vulnerability, the longest chain that an adversary will use from the original certificate graph is from the leaf node to the root node, which has length 6. Hence, we changed the value of limit k on chain length from 1 to 7. After 6, the vulnerability is the same as Fig. 10, shown as a dotted line here.

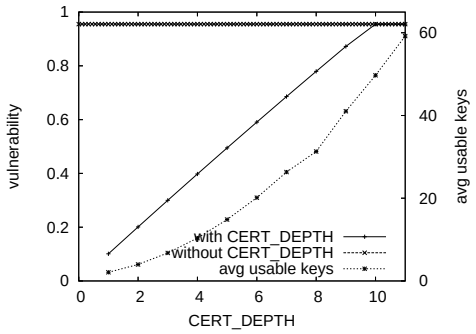

Figure 13: Effect of limit on chain length on vulnerability

As another example of acceptance criteria, we can use "path independence" proposed in [11]. This acceptance criteria requires k independent paths from src to dst for node src to be able to use the public key of dst in the certificate graph. To find independent paths, the authors propose to use the min-cut size of a certificate graph from src to dst. Since src only uses the public key of dst if the min-cut size of a certificate graph from src to dst is at least k, the adversary needs to

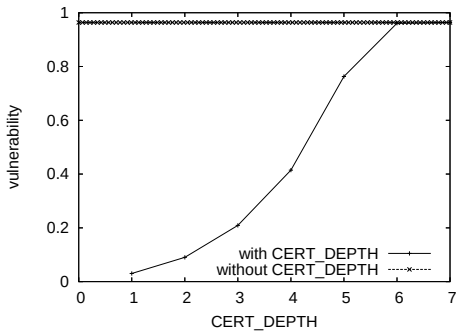

Figure 14: Effect of limit on chain length on vulnerability

know at least k private keys.

In the current Internet, SSL/TLS [4] is one of the most commonly used protocols based on certificates. In SSL/TLS, most of the websites that have certificates signed by a CA, such as VeriSign, do not have alternate certificates signed by other CA. In other words, there is only one chain from one node to another. For this type of certificate graphs, path independence cannot be used.

The other commonly used protocol based on certificates is PGP [17]. PGP certificate graphs have the properties of small world [15]. The certificate graph in Fig. 15 is an example of small world graphs. Fig. 16 shows how vulnerability changes as k changes for this example certificate graph. As k increases, the vulnerability decreases.

In both examples of acceptance criteria, the graphs in Fig. 13 and Fig. 16 show that a stricter acceptance criteria reduces the vulnerability of certificate graphs. However, it also increases the number of valid public keys that cannot satisfy the stricter acceptance criteria. For example, in Fig. 13, the average number of public keys a node can use increases as the depth limit increases. For the certificate graph in Fig. 15, the average number of public keys a node can use decreases from 5 to $\frac{14}{6} \sim 2.7$ as k increases from 2 to 3.

Also, according to the analysis in [3], the degrees of nodes in a self-organized certificate graph follow Zipf's distribution. In other words, most nodes in a self-organized certificate systems have a very small number of outgoing edges. In the Fig.9 in [3], about half of the nodes in the largest strongly connected component of the 2001 PGP graph have fewer than three outgoing edges, and about 30% of the nodes have only one outgoing edge. Therefore, when the path independence is applied as the acceptance criteria, a large k may cause many public keys to become unusable by other nodes. In the previous example of the 2001 PGP graph, $k \geq 3$ will cause half of the nodes not to be able to use any public keys in certificate chains of length at least 2.

Clearly, there is a trade-off between the vulnerability of a certificate graph and the usability of the public keys in the certificate graph. Hence, acceptance criteria needs to be chosen and configured very carefully. This metric of vulnerability can help system administrators balance the resilience against impersonation attacks and the usability of the public keys in certificate graphs.

8 Vulnerability of Many Revealed Keys

As shown in the previous section, when an acceptance criteria requires more than one certificate chain from a node *src* to a node *dst* for node *src* to accept the public key in the certificate chain as the public key of *dst*, the vulnerability of a certificate graph can change depending on how many private keys are revealed to an adversary. Theorem 7 is modified here to take the case where many

private keys are revealed to an adversary into the consideration.

Theorem 8 *Let G be a certificate graph and src and dst be any two distinct nodes in G. Let D be a set of nodes in G where the private key r.d of each node d in D is revealed to an adversary. The adversary can impersonate node dst to node src if and only if $src \neq d$ for any node d in D and one of the following two conditions holds.*

i. $d = dst$ for some node d in D and G has a set of certificate chains from src to dst that satisfies the acceptance criteria of G.

ii. $\nexists(src, dst)$ and the set of certificate chains, where each chain consists of a correct certificate chain from src to some node d in D that does not contain any certificate issued by node dst and a forged certificate (d, dst'), satisfies the acceptance criteria of G.

The vulnerability of the set D is defined as follows:

$$V(D) = \frac{|IMP(D)|}{(n - |D|) \times (n - 1)},$$

where $IMP(D) = \{(src, dst)|$ the adversary can impersonate dst to src using private keys of nodes in D and n is the number of nodes in G. Let G be a certificate graph and there can be at most x private keys revealed to an adversary, then the vulnerability of graph G with x revealed keys, denoted $V(G, x)$, is defined as follows:

$$V(G, x) = \max_{D \subseteq G, |D| \leq x} V(D)$$

Note that this definition generalizes the definition of $V(G)$, which is equal to $V(G, 1)$.

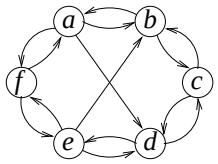

Figure 15: An example of a self-organized certificate graph

For the example certificate graph in Fig. 15, assume that the acceptance criteria of path independence with $k = 2$ is applied. Also, assume that the private keys of nodes b and d are revealed to an adversary. There is no certificate (a, c), and there are certificates (a, b) and (a, d), so the adversary can impersonate node c to node a. Also, there is no certificate (f, c), and there are certificate chains $(f, a)(a, b)$ and $(f, e)(e, d)$ that do not contain c, so the adversary can impersonate node c to node f. There are 16 node pairs (src, dst) such that the adversary can impersonate dst to src using the private keys of nodes b and d, so the vulnerability of $\{b, d\}$ is $\frac{16}{20}$. This is also the maximum vulnerability of the example certificate graph when $x = 2$, so $V(G, 2) = \frac{16}{20}$.

Fig. 16 shows how the vulnerability of the certificate graph in Fig. 15 changes as the number of revealed private keys changes. We applied the acceptance criteria of path independence with the parameter k from 1 to 3, and changed the number of revealed private keys x from 1 to 6. As the number of revealed private keys increases, the vulnerability increases. As long as the number of revealed private keys is less than k, the vulnerability is limited to explicit damage.

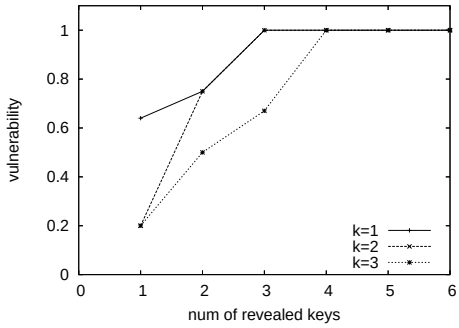

Figure 16: Vulnerability of many revealed keys

9 Related Work

This metric of vulnerability can be used in any certificate system. For example, X.509 [1], SSL/TLS [4], PGP [17], and SDSI/SPKI [5, 13]. In any of these certificate systems, when a private key of some node is revealed to an adversary, the adversary may successfully impersonate nodes to other nodes in the system. In other words, the certificate systems may be vulnerable to impersonation attacks.

There are three ways to reduce vulnerability of a certificate system: topology, dispersal, and acceptance criteria. Topology of a certificate graph is often not planned or designed. If one has the choice of certificate graph topology, the result from Section 3.

Storing a set of certificates in each node also reduces vulnerability. Hubaux, Buttyán, and Capkun [6] suggested storing $2\sqrt{n}$ certificates in each node in a mobile ad hoc network. Jung, Elmallah, and Gouda [7] proves that computing a dispersal with minimal average number of certificates is in general NP-Complete, and presented three algorithms to compute optimal dispersal for classes of certificate graphs. Zheng, Omura, Uchida, and Wada presented an optimal certificate dispersal algorithm for a strongly connected certificate graph.

Many researchers proposed mechanisms to evaluate certificate chains to mitigate this vulnerability. Tarah and Huitema [14] investigated using the path length as acceptance criteria. Reiter and Stubblebine [11] suggested "path independence" as discussed in Section 7. Beth, Borcherding, and Klein [2] and Maurer [10] proposed an acceptance criteria based on probabilities. In PGP [17], users can limit the length of acceptable certificate chains and also require certain number of certificate chains to accept the public key of destination node. Levien and Aiken [9] presented an analytical model of different types of attacks and compared the resilience of acceptance criteria in [10] and [11] based on this model. The same authors also suggested another acceptance criteria based on the max flow algorithm. In [12], Reiter and Stubblebine suggested a number of guiding principles for the design of acceptance criteria.

10 Conclusion

Impersonation attacks using forged certificates are inherent risk of certificate systems. We define a new "vulnerability" metric that evaluates the scope of explicit and implicit damage that can be caused by revealing the private key of some node to an adversary. Also, we define "vulnerability of certificate graph" as the maximum vulnerability among all nodes in the certificate graph. We give the three theorems to compute the vulnerability of three classes of certificate graphs, and also present three algorithms to compute the vulnerability of an arbitrary certificate graph. We also computed vulnerabilities when many private keys are revealed.

Using these algorithms, we investigate the effect of graph topology, certificate dispersal, and acceptance criteria on the vulnerability of certificate graphs. Graph topology shows that different topologies have different vulnerabilities even though they have Certificate dispersal in general reduces the vulnerability, but there is a trade-off between the dispersal cost of optimal dispersal and the vulnerability of tree certificate graphs. Stricter acceptance criteria reduces the vulnerability too, but also risks false negative.

These results are useful in several applications. When a system administrator designs a certificate system, he or she can choose the right combination of graph topology, certificate dispersal, and acceptance criteria to use based on system requirements. Graph topology is affected by the number of certificates a user can issue. Certificate dispersal lowers the vulnerability but requires more storage per node. Stricter acceptance criteria lowers the vulnerability but reduces the usability of public keys.

We plan to investigate further the trade-off between the optimal dispersal cost and the vulnerability in other classes of certificate graphs. We can compare certificate dispersals according to their resulting vulnerability, and also evaluate the design of certificate graphs.

References

[1] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 public key infrastructure – certificate management protocol (CMP). RFC 2510, 1999.

[2] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS '94) LNCS 875*, pages 3–18. Springer-Verlag, 1994.

[3] S. Capkun, L. Buttyán, and J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the ACM New Security Paradigms Workshop*. ACM Press, 2002.

[4] T. Dierks and E. Rescorla. The TLS protocol version 1.1. Internet Draft (draft-ietf-tls-rfc2246-bis-08.txt), 2004.

[5] C. Ellison. SPKI requirements. RFC 2692, 1999.

[6] J.-P. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking & computing*, pages 146–155. ACM Press, 2001.

[7] E. Jung, E. S. Elmallah, and M. G. Gouda. Optimal dispersal of certificate chains. In *Proceedings of the 18th International Symposium on Distributed Computing (DISC '04)*. Springer-Verlag, 2004.

[8] R. Kohlas and U. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In *Proceedings of PKC 2000, LNCS 1751*. Springer-Verlag, 2000.

[9] R. Levin and A. Aiken. Attack resistant trust metrics for public key certifications. In *Proceedings of the 7th USENIX Security Symposium*, 1998.

[10] U. Maurer. Modeling a public-key infrastructure. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS '96)*. Springer-Verlag, 1996.

[11] M. K. Reiter and S. G. Stubblebine. Resilient authentication using path independence. *IEEE Transactions on Computers*, 47(12):1351–1362, December 1998.

[12] M. K. Reiter and S. G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):138–158, 1999.

[13] R. L. Rivest and B. Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO '96 Rumpsession, 1996.

[14] A. Tarah and C. Huitema. Associating metrics to certification paths. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS '92) LNCS 648*. Springer-Verlag, 1992.

[15] D. Watts. *Small Worlds*. Princeton University Press, 1999.

[16] H. Zheng, S. Omura, J. Uchida, and K. Wada. An optimal certificate dispersal algorithm for mobile ad hoc networks. In *Proceedings of Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04)*, 2004.

[17] P. Zimmerman. *The Official PGP User's Guide*. MIT Press, 1995.