Certificate Dispersal in Ad Hoc Networks

Mohamed G. Gouda and Eunjin Jung
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{gouda, ejung}@cs.utexas.edu

***Abstract*—We investigate how to disperse the certificates, issued in an ad hoc network, among the network nodes such that the following condition holds. If any node *u* approaches any other node *v* in the network, then *u* can use the certificates stored either in *u* or in *v* to obtain the public key of *v* (so that *u* can securely send messages to *v*). We define the cost of certificate dispersal as the average number of certificates stored in one node in the network. We give upper and lower bounds on the dispersability cost of certificates, and show that both bounds are tight. We also present two certificate dispersal algorithms, and show that one of those algorithms is more efficient than the other in several important cases. Finally, we identify a rich class of "certificate graphs" for which the dispersability cost is within a constant factor from the lower bound.**

I. Introduction

We consider a network where each node u has a private key $rk.u$ and a public key $bk.u$. In this network, in order for a node u to securely send a message m to another node v, node u needs to encrypt the message m using the public key $bk.v$, before sending the encrypted message, denoted $bk.v < m >$, to node v. This necessitates that node u know the public key $bk.v$ of node v.

If a node u knows the public key $bk.v$ of another node v in this network, then node u can issue a certificate, called a certificate from u to v, that identifies the public key $bk.v$ of node v. This certificate can be used by any node in the network that knows the public key of node u to further acquire the public key of node v.

A certificate from node u to node v is of the following form:

$$rk.u < u, v, bk.v >$$

This certificate is encrypted using the private key $rk.u$ of node u, and it includes three items: the identity of the certificate issuer u, the identity of the certificate subject v, and the public key of the certificate subject $bk.v$. Any node that knows the public key $bk.u$ of node u can use $bk.u$ to decrypt the certificate from u to v and obtain the public key $bk.v$ of node v.

The certificates issued by different nodes in a network can be represented by a directed graph, called the certificate graph of the network. Each node in the certificate graph represents a node in the network. Each directed edge from node u to node v in the certificate graph represents a certificate issued by node u for node v in the network.

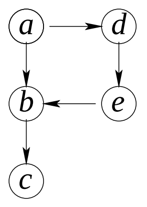


Fig. 1. A certificate graph example

Figure 1 shows a certificate graph for a network with five nodes: a, b, c, d, and e. According to this graph,

node a issued two certificates (a, b), and (a, d)
node b issued one certificate (b, c)
node c issued no certificate
node d issued one certificate (d, e)
node e issued one certificate (e, b).

Node a can use the two certificates (a, b) and (b, c) to obtain the two public keys $bk.b$ and $bk.c$, and so can securely send messages to nodes b and c. Also, node a can use the two certificates (a, d) and (d, e) to obtain the public keys $bk.d$ and $bk.e$, and can securely send messages to nodes d and e. Node d can use the three certificates (d, e), (e, b), and (b, c) to obtain the public keys of $bk.e$, $bk.b$, and $bk.c$, and can securely send messages to nodes e, b, and c.

We assume that the network is ad hoc and its nodes are mobile so move around. In this case, the issued certificates need to be dispersed among the nodes in the network such that if a node u approaches another node v and wishes to securely send messages to it, then u can obtain the public key of v from the set of certificates stored either in u or in v (provided there is a directed path from u to v in the certificate graph).

As an example, assume that each node in the certificate graph in Figure 1 stores the certificates in the maximal, shortest-path, incoming tree rooted at the node:

node a stores no certificates	
node b stores the certificates	(a, b), (d, e), (e, b)
node c stores the certificates	(a, b), (d, e), (e, b), (b, c)
node d stores the certificates	(a, d)
node e stores the certificates	(a, d), (d, e)

Thus, if node a approaches node e, then a can use the two certificates stored in e to obtain the public key of e and securely send messages to e. (Note that node e can never obtain the public key of node a because there is no directed path from node e to node a in the certificate graph in Figure 1).

As an application of this situation, consider the tanks in an armored division. Each tank has a computer and can be viewed as a node in an ad hoc network. Before the tanks are deployed into the field, a certificate graph needs to be designed to secure the future communications between the tanks in the field. Then the certificates from this certificate graph need to be dispersed amongst the tanks before they are deployed. Later, the tanks are deployed into the field and each of them has a number of certificates in its local storage. Now, if two tanks approach each other in the field, then the two tanks have enough certificates in their local storage so that each of them can compute the public key of the other and two tanks can securely exchange messages.

In this paper, we discuss three contributions to the problem of certificate dispersal in ad hoc networks. First, we present tight upper and lower bounds on the average number of certificates stored in one node. Second, we present a somewhat efficient certificate dispersal algorithm that ensures that the average number of certificates stored in a node is small (if not the smallest). Third, we identify rich classes of certificate graphs for which the average number of certificates to be stored in a node approaches the lower bound.

II. Related Work

Several papers have investigated the use of certificates to provide security in traditional and in ad hoc networks. We summarize the results of these papers in the following paragraphs.

Architectures for issuing, storing, discovery, and validating certificates in traditional networks are presented in [1], [2], [3], [4], [5], [6], [7], and [8]. There are several limitations in ad hoc networks that do not allow us to use these results without significant overhead to the networks. In traditional networks, one can assume that the communication between nodes is reliable and reasonably fast. In ad hoc networks, finding routes between nodes itself is challenging, let alone maintaining the relevant information of the found routes. Also, nodes in ad hoc networks often have very limited resources. For example, computational capability, storage, and power supply are much less than what most nodes have in traditional networks. To accommodate these limitations, different architectures for issuing, storing, discovery, and validating certificates in ad hoc networks have been developed.

In [9], Zhou and Haas have presented an architecture for issuing certificates in an ad hoc network. According to this architecture, the network has k servers. Each server has a different share of some private key rk. To generate a certificate, each server uses its own share of rk to encrypt the certificate. If no more than t servers have suffered from Byzantine failures, where $k \geq 3t+1$, then the resulting certificate is correctly encrypted using the private key rk, thanks to threshold cryptography. The resulting certificate can be decrypted using the corresponding public key which is known to every node in the ad hoc network.

In [10], Kong, Perfos, Luo, Lu and Zhang presented more distributed architecture for issuing certificates. Instead of employing k servers in the ad hoc network, each node in the network is provided with a different share of the private key rk. For a node u to issue a certificate, the node u forwards the certificate to its neighbors and each of them encrypt the certificate using its share of rk. If node u has at least $t+1$ correct neighbors (i.e. they have not suffered from any failures), the resulting certificate is correctly encrypted using the private key rk.

In our paper, we propose an architecture where every node has both a public key and a private key so it can issue certificates for any other node in the network. This architecture is very efficient in issuing and validating certificates but cannot tolerate Byzantine failures. In particular, if one node suffers from Byzantine failure, then this node can successfully impersonate any other node that is reachable from this node in the certificate graph of the network. This vulnerability to Byzantine failures is not unique to our certificate work in ad hoc networks. In fact, many proposed certificate architectures, e.g. [1], [2], [3], [7], and [8] yield similar vulnerabilities in traditional networks. Recently, we have developed a technique for augmenting the certificates with additional information to ensure that the network can tolerate some degree of Byzantine failures[11].

Perhaps the closest work to ours is [12] where the authors, Hubaux, Buttyán, and Capkun, investigated how to disperse certificates in a certificate graph among the network nodes under two conditions. First, each node stores the same number of certificates. Second, with high probability, if two nodes meet then they have enough certificates for each of them to compute the public key of the other. By contrast, our work is based on two different conditions. First, different nodes may store different number of certificates, but the average number of certificates stored in one node is minimized. Second, it is guaranteed (i.e. with probability 1) that if two nodes meet then they have enough certificates for each of them to compute the public key of the other.

Later, the same authors have showed in [13] that a lower bound on the number of certificates to be stored in a node is $\sqrt{n}-1$ where n is the number of nodes in the system. By contrast, we show below that the tight lower bound on the average number of certificates to be stored in a node is e/n, where e is the number of edges in the system.

III. Certificate Dispersal

A *certificate graph* G is a directed graph in which each directed edge, called a *certificate*, is a pair (u, v), where u and v are distinct nodes in G. For each certificate (u, v) in G, u is called the *issuer* of the certificate and v is called the *subject* of the certificate.

Note that according to this definition a certificate graph is a directed graph that does not have self-loops and does not have multiple edges from any node to any other node.

A directed path (v_0, v_1), (v_1, v_2), $\cdots$, (v_{k-1}, v_k) in a certificate graph G, where the nodes v_0, v_1, $\cdots$, v_k are distinct, is called a certificate *chain* from v_0 to v_k. The *length* of a chain is the number of certificates in the chain. A chain from u to v is *shortest* iff its length is not larger than the length of any other chain from u to v in the same certificate graph.

Let c denote the chain (v_0, v_1), $\cdots$, (v_{k-1}, v_k). Then, each of the chains (v_0, v_1), $\cdots$, (v_{j-1}, v_j), where $1 \leq j \leq k$, is called a *prefix* of c that ends at node v_j. Also, each of the chains (v_j, v_{j+1}), $\cdots$, (v_{k-1}, v_k), where $0 \leq j \leq k-1$, is called a *suffix* of c that starts at node v_j.

A certificate *dispersal algorithm* F is an algorithm that takes as input any certificate graph G and computes a subset of the certificates, denoted $F.(G,v)$, for every node v in G such that the following two conditions hold:

i) *Connectivity* :
For every distinct pair of nodes u and v in G, if there is a chain from u to v in G, then there is a chain from u to v in the set $F.(G,u) \cup F.(G,v)$.

ii) *Completeness* :
For every certificate in G, there is a node v in G such that this certificate is in $F.(G,v)$.

Let G be a certificate graph, and F be a certificate dispersal algorithm. The *cost* of using F to disperse the certificates in G among the nodes of G, denoted $c.(F,G)$, is computed as follows:

$$c.(F,G) = \frac{1}{n}(\sum_{v \text{ in } G} |F.(G,v)|)$$

where n is the number of nodes in G, and $|F.(G,v)|$ denotes the number of certificates in the set $F.(G,v)$ assigned by F to node v. Note that $c.(F,G)$ is the average number of certificates assigned by F to a node in G.

The *dispersability cost* of a certificate graph G, denoted $d.G$, is computed as follows:

$$d.G = \min_{F} c.(F,G)$$

A certificate dispersal algorithm F_e is *efficient* iff for every certificate graph G,

$$c.(F_e,G) = d.G$$

It follows from this definition that if F_e is an efficient certificate dispersal algorithm and F is a certificate dispersal algorithm then for every certificate graph G,

$$c.(F_e,G) \leq c.(F,G)$$

Lemma 1 : *(Upper Bound on Dispersability Cost)* For every certificate graph G with n nodes,

$$d.G \leq n-1$$

.

Proof :
In the next section, we present a certificate dispersal algorithm F_{full} that assigns to every node v in a certificate graph G, the certificates in a maximal outgoing tree rooted at v. Because each maximal outgoing tree in a certificate graph G has at most $n-1$ certificates, where n is the number of nodes in G, we have

$$\begin{aligned}
|F_{full}(G,v)| &\leq n-1 \\
c.(F_{full},G) &= \frac{1}{n}(\sum_{v \text{ in } G} |F.(G,v)|) \\
&\leq (n-1)
\end{aligned}$$

Let F_e be any efficient dispersal algorithm; hence

$$\begin{aligned}
d.G &= c.(F_e,G) \\
&\leq c.(F_{full},G)) \\
&\leq n-1
\end{aligned}$$

□

Lemma 2 : *(Lower Bound on Dispersability Cost)*
For every certificate graph G with n nodes and e certificates,

$$d.G \geq \frac{e}{n}$$

.

Proof :
Let F_e be any efficient certificate dispersal algorithm. From the completeness condition of a dispersal algorithm, every edge in a certificate graph G is assigned by F_e to some node in G. Thus,

$$\sum_{v \,\mathrm{in}\, G} |F_e.(G,v)| \geq e$$

where e is the number of certificates in G. We conclude

$$\begin{aligned} d.G &= c.(F_e, G) \\ &= \frac{1}{n}(\sum_{v \,\mathrm{in}\, G} |F_e.(G,v)|) \\ &\geq \frac{e}{n} \end{aligned}$$

□

Lemma 3 : *(Achieving the two Bounds on Dispersability Cost)*
There is a certificate graph G with n nodes and e certificates such that the following two conditions hold.

i) $d.G = n-1$
ii) $d.G = \frac{e}{n}$

Proof :
Let G be a fully connected certificate graph; i.e, for any two distinct nodes u and v in G, there is a certificate from u to v and a certificate from v to u. Thus, the number of certificates e in G is $n(n-1)$, and the following relations hold in G.

$$n-1 = \frac{n(n-1)}{n} = \frac{e}{n}$$

Hence the upper and lower bounds on dispersability cost meet at G, and $d.G$ is equal to each of the two bounds. Therefore, the two conditions of Lemma 3 hold for G.

□

It is also possible to construct certificate graphs that meet the upper bound on dispersability cost but not its lower bound, and to construct certificate graphs that meet the lower bound on dispersability cost but not its upper bound.

As an example, consider a *ring* certificate graph G_0 in Figure 2. This graph has n nodes and n certificates arranged in a directed ring. It is straightforward to show that any dispersal algorithm will assign to every node in the graph at least $n-1$ certificates. Thus, $d.G_0 \geq n-1$, and G_0 meets the upper bound on dispersability cost but not its lower bound.

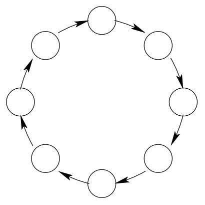


Fig. 2. A ring certificate graph

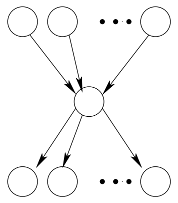


Fig. 3. An hourglass certificate graph

As a second example, consider an *hourglass* certificate graph G_1 in Figure 3. This graph has n nodes and $n-1$ certificates, where n is odd, arranged in an hourglass shape with one center node, $(n-1)/2$ input nodes, and $(n-1)/2$ output nodes. There is a certificate dispersal algorithm F that assigns certificates to every node v in G_1 as follows.

i) If v is the center node, then $F.(G_1, v) = \{\}$.
ii) If v is an input node, then $F.(G_1, v) = \{(v, center\, node)\}$.
iii) if v is an output node, then $F.(G_1, v) = \{(center\, node, v)\}$.

Thus,

$$\begin{aligned} c.(F, G_1) &= \frac{n-1}{n} \\ &= \text{the lower bound on dispersability cost} \\ &= d.G_1 \end{aligned}$$

Hence, G_1 meets the lower bound on dispersability cost but not its upper bound.

The above discussion suggests the following two problems which we explore in the rest of this paper.

Problem 1 :
Develop an efficient certificate dispersal algorithm F_e.

□

Problem 2 :
Identify rich classes of certificate graphs whose dispersability costs meet the lower bound or are within a constant factor of this lower bound.

□

Problem 1 remains open: Instead of solving Problem 1, we present two certificate dispersal algorithms F_{full} and F_{half} (in Sections 4 and 5 respectively), and show that in several important cases F_{full} is not as efficient as

F_{half}. We then present a solution of Problem 2 in Section 6.

IV. Full Tree Algorithm for Certificate Dispersal

Before we introduce our first certificate dispersal algorithm, we need to introduce the following definition of chain sets.

Let G be a certificate graph and v be a node in G. A *chain set* for v, denoted $S.v$, is a set of chains in G that satisfies the following three conditions.

i) If G has no chains that starts at v, then $S.v$ is empty.
ii) If G has a chain from v to w, then $S.v$ has exactly one shortest chain from v to w.
iii) If $S.v$ has a chain, then $S.v$ also has every nonempty prefix of this chain.

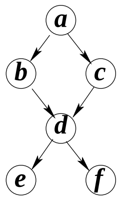


Fig. 4. The diamond certificate graph

As an example, consider the *diamond* certificate graph in Figure 4. In this graph, there are no certificate chains that start at node e or f, and the chain sets for node e and f are both empty:

$$S.e = \{\}$$
$$S.f = \{\}$$

The chain set for node d has two chains:

$$\begin{aligned} S.d = \{ &< (d,e) >, \\ &< (d,f) > \} \end{aligned}$$

Also the chain set for each of the two nodes b and c has three chains:

$$\begin{aligned} S.b = \{ &< (b,d) >, \\ &< (b,d);(d,e) >, \\ &< (b,d);(d,f) > \} \\ S.c = \{ &< (c,d) >, \\ &< (c,d);(d,e) >, \\ &< (c,d);(d,f) > \} \end{aligned}$$

The chain set for node a has five chains:

$$\begin{aligned} S.a = \{ &< (a,b) >, \\ &< (a,c) >, \\ &< (a,c);(c,d) >, \\ &< (a,c);(c,d);(d,e) >, \\ &< (a,c);(c,d);(d,f) > \} \end{aligned}$$

The following two comments are in order. First, each chain set $S.v$ for a node v defines a maximal, shortest-path, outgoing tree rooted at node v in the certificate graph. Second, it is possible to have two or more distinct chain sets for a node. For example, a second chain set for node a in the certificate graph in Figure 4 is as follows:

$$\begin{aligned} \{ &< (a,b) >, \\ &< (a,c) >, \\ &< (a,b);(b,d) >, \\ &< (a,b);(b,d);(d,e) >, \\ &< (a,b);(b,d);(d,f) > \} \end{aligned}$$

Using the above definition of a chain set, we are now ready to present our first certificate dispersal algorithm, called the *full tree algorithm* and denoted F_{full}. This algorithm assigns to every node v all the certificates in a chain set $S.v$ for v. In other words,

$F_{full}.(G,v)$ = the set of all certificates that exist in a chain set $S.v$ for v.

Lemma 4 :
F_{full} is a certificate dispersal algorithm.

Proof :
We show that F_{full} satisfies the two conditions of a certificate dispersal algorithm, connectivity and completeness. First, if there is a chain from u to v in G, then at least one of the shortest chains from u to v is in $S.u$ by condition *ii* in the definition of chain set. Second, any certificate (u, v) in G is in $S.u$ since it is the shortest chain from u to v. By the definition of F_{full}, the certificate (u, v) is in $F_{full}.(G,u)$. Therefore, F_{full} satisfies two properties of connectivity and completeness.

□

Next, we show that the dispersal algorithm F_{full} is far from being efficient. First, we show in Lemma 5 that the cost of applying F_{full} to any strongly connected certificate graph meets the upper bound on dispersability cost. Second, we show in Lemma 6 that the cost of applying F_{full} to any hourglass certificate graph is within a factor of four from the upper bound on dispersability cost (even though the dispersability cost of an hourglass graph meets the lower bound).

Lemma 5 :
For any strongly connected certificate graph G with n nodes,

$$c.(F_{full}, G) = n - 1$$

Proof :
The certificate dispersal algorithm F_{full} assigns, to every node v in a certificate graph G, the certificates in a maximal outgoing tree rooted at v. If G is strongly connected, then any maximal outgoing tree is in fact a spanning tree with $(n-1)$ certificates, where n is the number of nodes in G. Therefore, for any node v in G,

$$\begin{aligned} |F_{full}.(G,v)| &= n-1 \\ c.(F_{full}, G) &= \frac{1}{n}(\sum_{v \,\mathrm{in}\, G} |F_{full}.(G,v)|) \\ &= n-1 \end{aligned}$$

□

Lemma 6 :
For any hourglass certificate graph G with n nodes (see Figure 3),

$$c.(F_{full}, G) = \frac{n^2+2n-3}{4n} \sim \frac{n}{4}$$

Proof :
Recall that any hourglass certificate graph G has one center node, $\frac{n-1}{2}$ input nodes, and $\frac{n-1}{2}$ output nodes.

$$|F_{full}.(G, \text{center})| = \frac{n-1}{2}$$

For every input node v,

$$|F_{full}.(G,v)| = \frac{n-1}{2} + 1 = \frac{n+1}{2}$$

For every output node v,

$$|F_{full}.(G,v)| = 0$$

Thus,

$$\begin{aligned} c.(F_{full}, G) &= \frac{1}{n}(\frac{n-1}{2} + (\frac{n-1}{2})(\frac{n+1}{2})) \\ &= \frac{n^2+2n-3}{4n} \\ &\sim \frac{n}{4} \end{aligned}$$

□

V. Half Tree Algorithm for Certificate Dispersal

Before we introduce our second certificate dispersal algorithm, we need to introduce the following definition of consistent chain sets.

Let $S.u$ and $S.v$ be two chain sets for nodes u and v, respectively, in a certificate graph G. $S.u$ and $S.v$ are *consistent* iff for every two nodes x and y in G, if $S.u$ has a subchain that starts at x and ends at y and $S.v$ also has a subchain that starts at x and ends at y then these two subchains are identical.

A collection of chain sets $\{S.v | v \text{ is a node in } G\}$ is *consistent* iff every two chain sets in the collection are consistent.

We are now ready to present our second certificate dispersal algorithm, called the *half tree algorithm* and denoted F_{half}. This algorithm takes as input a consistent collection of chain sets $\{S.v | v \text{ is a node in a certificate graph } G\}$ and computes a set of certificates $F_{half}.(G,v)$ for every node v in G. Algorithm F_{half} is defined as follows.

```
1:  for every nonempty $S.v$ in the consistent collection
        of chain sets do
2:      let $c$ denote the longest chain $< (v_0,v_1); \cdots ;$
            $(v_{k-1},v_k) >$ in $S.v$: note that $v_0 = v$;

3:      let $x := \lfloor \frac{k}{2} \rfloor$;

4:      find the largest $y$, $0 \le y \le k$, such that all
            certificates in the prefix $< (v_0,v_1); \cdots ;$
            $(v_{y-1},v_y) >$ are already in $F_{half}(G,v)$;

5:      if $x \le y$
6:      then
            store the certificates in every prefix of
            the subchain $< (v_y,v_{y+1}); \cdots ; (v_{k-1},v_k) >$
            in $F_{half}.(G,w)$ where $w$ is the node at
            which the prefix ends;
7:      else
7a:         store the certificates in the prefix
            $< (v_y,v_{y+1}); \cdots ; (v_{x-1},v_x) >$ in $F_{half}(G,v)$;
7b:         store the certificates in every prefix of
            the subchain $< (v_x,v_{x+1}); \cdots ; (v_{k-1},v_k) >$
            in $F_{half}.(G,w)$ where $w$ is the node at
            which the prefix ends;
        endif;

8:      remove chain $c$ from $S.v$;
9:  enddo;
```

Lemma 7 :
F_{half} is a certificate dispersal algorithm.

Proof :
First, if there is a chain between nodes u and v, then at least one of the shortest chains from u to v is stored in $S.u$ either as it is or as part of a longer chain. All the certificates in the chain from u to v will be stored in u and v by the definition of F_{half}. Second, any certificate (u, v) in G will be stored in $S.u$ since it will be the shortest chain from u to v. By the definition of F_{half}, the certificate (u, v) is stored in u or in v. Therefore, F_{half} satisfies two properties of certificate dispersal algorithm. □

Next, we show in Lemma 8 that in the important case of strongly connected certificate graphs, F_{half} is not less efficient than F_{full}, and in some instances, F_{half} is in fact more efficient than F_{full}. Then in Lemma 9, we show that in the important case of tree certificate graphs, F_{half} is not less efficient than F_{full}, and in some instances, F_{half} is in fact more efficient than F_{full}. In Lemma 10, we show that in the case of the hourglass certificate graphs F_{half} achieves the lower bound on dispersability cost which is much less than what F_{full} achieves. (For convenience, the proof of Lemma 9 is moved to the appendix at the end of the paper.)

Lemma 8 :
For any strongly connected certificate graph G,

$$c.(F_{half}, G) \leq c.(F_{full}, G)$$

For some strongly connected certificate graph G,

$$c.(F_{half}, G) < c.(F_{full}, G)$$

Proof :
Let G be any strongly connected certificate graph, and v be any node in G. The certificates in the set $F_{half}.(G, v)$ define a graph G', which is a subgraph of the original graph G. In G', there can be at most one path from any node to node v, and at most one path from node v to any other node. Graph G' satisfies exactly one of the following two conditions.

i) G' has no cycle.
ii) G' has a cycle, but it has at most $n-1$ nodes.

In the first case, the number of certificates in G' is at most $n-1$, since there is no cycle in G'. In the second case, the number of certificates in G' is also at most $n-1$, which is the number of certificates if all the $n-1$ nodes participate in the cycle. Therefore, $|F_{half}.(G, v)| \leq n-1$.

$$\begin{aligned} c.(F_{half}, G) &= \frac{1}{n} \sum_{v \text{ in } G} |F_{half}.(G, v)| \\ &\leq \frac{n(n-1)}{n} \\ &= n-1 \end{aligned}$$

Because G is strongly connected, $c.(F_{full}, G) = n-1$ by Lemma 5. Therefore,

$$c.(F_{half}, G) \leq c.(F_{full}, G)$$

This completes our proof of the first part of the lemma.

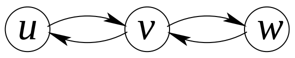


Fig. 5. The two-ring certificate graph

To prove the second part of the lemma, consider the two-ring certificate graph G'' in Figure 5. This graph is strongly connected and has three nodes. Then by Lemma 5,

$$\begin{aligned} c.(F_{full}, G'') &= n-1 \\ &= 2 \end{aligned}$$

By applying F_{half} to G'', we get

$$\begin{aligned} F_{half}.(G'', u) &= \{(u, v), (v, u)\} \\ F_{half}.(G'', v) &= \{\} \\ F_{half}.(G'', w) &= \{(v, w), (w, v)\} \end{aligned}$$

Therefore,

$$\begin{aligned} c.(F_{half}, G'') &= \frac{1}{3}(2+0+2) = \frac{4}{3} \\ &< c.(F_{full}, G'') \end{aligned}$$

□

Lemma 9 :
For every tree certificate graph T,

$$c.(F_{half}, T) \leq c.(F_{full}, T)$$

For any complete tree certificate graph G,

$$c.(F_{half}, G) < c.(F_{full}, G)$$

□

Lemma 10 :
For any hourglass certificate graph G with n nodes and e certificates (see Figure 3) where n is odd,

$$c.(F_{half}, G) = \frac{e}{n} < c.(F_{full}, G)$$

Proof :
Recall that an hourglass certificate graph G with n nodes has one center node, $\frac{n-1}{2}$ input nodes, and $\frac{n-1}{2}$ output nodes. Applying F_{half} to this certificate graph, we get

$$\text{for every input node } u, \quad F_{half}.(G,u) = \{(u,c)\}$$

$$\text{for the center node } c, \quad F_{half}.(G,c) = \{\,\}$$

$$\text{for every output node } w, \quad F_{half}.(G,w) = \{(c,w)\}$$

Therefore,

$$\begin{aligned} c.(F_{half},G) &= \frac{n-1}{n} = \frac{e}{n} < \frac{n}{4} \\ &\sim c.(F_{full},G) \end{aligned}$$

□

VI. Certificate Graphs with Small Dispersability Costs

In this section, we consider Problem 2, stated in Section 2, and identify a class of strongly connected certificate graphs that have small dispersability costs. This class is based on the star certificate graph G in Figure 6.

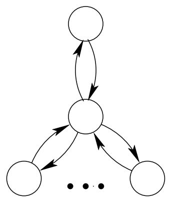


Fig. 6. A star certificate graph

The star graph consists of one center node and m satellite nodes. In this graph, the center node is connected with each of the satellite nodes by a directed ring. The best way to disperse the certificates in this graph is to assign the two certificates in each ring to the satellite node in that ring, while assigning no certificates to the center node. Thus,

$$d.G = \frac{1}{m+1}(2m)$$

Because the number of nodes in this graph is $(m+1)$ and the number of certificates in this graph is $2m$, the dispersability cost of this graph achieves its lower bound.

Unfortunately, the star certificate graph has a security problem. If the private key of the center node is compromised by an adversary, then this adversary can impersonate any node as it communicates with any other node in the system. Thus, although the dispersability cost of this graph achieves the lower bound, the security problem of this graph compels us to seek other certificate graphs. Next, we discuss how to generalize this star graph into a class of graphs whose dispersability costs are small and whose security problems are not so severe.

In our generalization of the star certificate graph, each of the m rings in the graph has k satellite nodes (beside the center node that exists in every ring). We refer to this generalized certificate graph as (m,k)-star. Figure 7 shows an $(m,2)$-star.

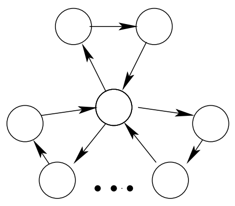


Fig. 7. An (m,2)-star certificate graph

An efficient way to disperse the certificates in the (m,k)-star G is to assign the $(k+1)$ certificates in a ring to every satellite node in that ring, while assigning no certificates to the center node. Thus,

$$d.G = \frac{1}{mk+1}(mk(k+1))$$

The number of nodes in the (m,k)-star is $mk+1$, and the number of certificates in this graph is $m(k+1)$. Thus, the lower bound on the dispersability cost for the (m,k)-star is $\frac{m(k+1)}{mk+1}$. Therefore, the dispersability cost of an (m,k)-star is within a factor of k from its lower bound.

It is straightforward to show that an (m,k)-star, where $k \geq 2$, has better security properties than the original $(m,1)$-star. In particular, if the private key of the center node is compromised by an adversary then this adversary will not be able to impersonate any satellite node in a ring while it communicates with any other satellite node in the same ring.

VII. Conclusion

In this paper, we introduce the problem of certificate dispersal. Tight lower and upper bounds on the cost of the certificate dispersability are given along with example certificate graphs that achieve both bounds. Two certificate dispersal algorithms, F_{full} and F_{half}, that can reach these bounds for certain graphs are devised. The algorithm F_{full} makes each node store a maximal shortest-path outgoing tree, whereas F_{half} makes each node store half of an outgoing tree and half of an incoming tree. We show that F_{half} performs better than F_{full} in strongly connected certificate graphs and in tree certificate graphs. We also present a class of certificate graphs whose dispersability costs is within a constant factor from the lower bound.

The dispersal algorithm discussed in this paper assume that all the directed paths between two nodes are equally good. In some applications this may not be true, as discussed in [14]. Further research is needed to adapt the dispersal algorithms in this paper to these applications.

REFERENCES

[1] Ronald L. Rivest and Butler Lampson, "SDSI – A simple distributed security infrastructure," Presented at CRYPTO'96 Rumpsession, 1996.

[2] S. Boeyen, T. Howes, and P. Richard, "Internet X.509 public key infrastructure operational protocols - LDAPv2," RFC 2559, 1999.

[3] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - OCSP," RFC 2560, 1999.

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

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The keynote trust-management system version 2," RFC 2704, 1999.

[6] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R.L. Rivest, "Certificate chain discovery in SPKI/SDSI," *Journal of Computer Security*, vol. 9, no. 4, pp. 285–322, 2001.

[7] Ninghui Li, William H. Winsborough, and John C. Mitchell, "Distributed credential chain discovery in trust management: extended abstract," in *Proceedings of the 8th ACM conference on Computer and Communications Security*. 2001, pp. 156–165, ACM Press.

[8] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti, "dRBAC: distributed role-based access control for dynamic coalition environments," in *Proceedings. 22nd International Conference on Distributed Computing Systems, 2002*, 2002, pp. 411–420.

[9] Lidong Zhou and Zygmunt J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[10] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang, "Providing robust and ubiquitous security support for wireless mobile networks," in *Ninth Internation Conference on Network Protocols (ICNP'01)*, 2001, pp. 251–260.

[11] Eunjin Jung and Mohamed G. Gouda, "Strongly chained certificates," *in preparation*, 2003.

[12] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun, "The quest for security in mobile ad hoc networks," in *Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking & computing*. 2001, pp. 146–155, ACM Press.

[13] Srdjan Capkun, Levente Buttyán, and Jean-Pierre Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[14] Michael K. Reiter and Stuart G. Stubblebine, "Authentication metric analysis and design," *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 2, pp. 138–158, 1999.

VIII. APPENDIX

Lemma 9 :
For every tree certificate graph T,

$$c.(F_{half}, T) \le c.(F_{full}, T)$$

For any complete tree certificate graph G,

$$c.(F_{half}, G) < c.(F_{full}, G)$$

Proof :
For any node u in G, the reduced chain set $S.u$ of u constructs a maximal shortest-path outgoing tree T_u. Since we may repeatedly store same incoming edges in nodes in F_{half}, $c.(F_{half}, G) \le \sum_{u\in G} c.(F_{half}, T_u)$, while $c.(F_{full}, G) = \sum_{u\in G} c.(F_{full}, T_u)$. If we can prove $c.(F_{half}, T_u) \le c.(F_{full}, T_u)$ for any tree T_u, then

$$\begin{aligned} c.(F_{half}, G) &\le \sum_{u\in G} c.(F_{half}, T_u) \\ &\le \sum_{u\in G} c.(F_{full}, T_u) = c.(F_{full}, G) \end{aligned}$$

$$c.(F_{half}, G) \le c.(F_{full}, G)$$

We can prove $c.(F_{half}, T_u) \le c.(F_{full}, T_u)$ for any tree T_u by induction. When the number of certificates is 2 in the maximal tree, there are 2 possible trees. If the tree looks like Figure 8(a), then $c.(F_{half}, T_u) = c.(F_{full}, T_u) = 2$. If the tree looks like Figure 8(b), then $c.(F_{half}, T_u) = 3$, whereas $c.(F_{full}, T_u) = 2$. Therefore $c.(F_{half}, T_u) \le c.(F_{full}, T_u)$ holds for any maximal tree T_u with 2 certificates.

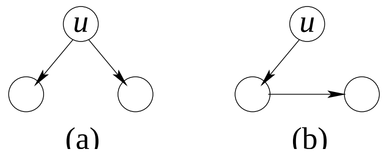
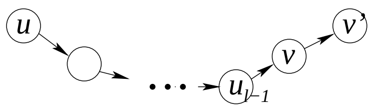


Fig. 8. Maximal trees with 2 edges

Let's assume that $c.(F_{half}, T_u) \le c.(F_{full}, T_u)$ holds for trees with up to n certificates. When $n+1^{th}$ certificate (v, v') is added at a node v, then it will increase the chain length from the root node u of the tree to v(This new certificate has to come with a new subject node v', otherwise it will break the tree property). For a chain from u to a leaf node v' in the given maximal tree T_u, we show that $\sum_{w\in u->v} |F_{half}(T_u, w)| \le \sum_{w\in u->v} |F_{full}(T_u, w)|$ for any node w on the path from u to v'. The number of certificates stored in the nodes that are not on the path from u to v' will not be affected by this new certificate.



Let l be the chain length from u to v. By the definition of F_{full} algorithm, the increment of $c.(F_{full}, T_u)$ is $l+1$ because the nodes from u to v will store the new certificate (v, v') locally.

For F_{half}, if a node w is far from v' by even length of chain, for example u_{l-1}, the node w has to store one more outgoing certificates, as the chain length from w to the leaf node v' increases. If $l = 2k$, then the number of such nodes are k. Also $c.(F_{half}, T_u)$ is increased by $F_{half}.(T_u, v')$, which is $k+1$. Therefore, the increment of $c.(F_{half}, T_u)$ is also $l+1$, which is equal to that of $c.(F_{full}, T_u)$. If $l = 2k+1$, then the nodes which stores one more outgoing certificate are k, and $F_{half}.(T_u, v')$ is $k+1$. But in this case, the certificate from kth node to $k+1$th node on the chain is not going to be stored as incoming certificate in any nodes any longer. Therefore, $k+1$ nodes can reduce their $F_{half}.(T_u, v')$ by 1. In total, the increment will be k in $l = 2k+1$ case.

Since the increment of $c.(F_{half}, T)$ is $l+1$ or $(l-1)/2$ when that of $c.(F_{full}, T)$ is fixed as $l+1$ when $n+1$th certificate is added, $c.(F_{half}, T) \leq c.(F_{full}, T)$ holds for any tree T with $n+1$ number of certificates.

By induction, it is shown that $c.(F_{half}, G) \leq c.(F_{full}, G)$ for any maximal tree T_u for any node u in G. Therefore, $c.(F_{half}, G) \leq c.(F_{full}, G)$ for any tree certificate graph G. This completes our proof of the first part of the lemma.

To prove the second part of the lemma, let h be $\lfloor \log_d n \rfloor$, which is the height of the tree, where d is the degree of the tree, $d \geq 2$.

$$\begin{aligned} c.(F_{full}, G) &= \sum_{v \text{ in } G} \text{the number of certificates that appear} \\ &\quad \text{in } S.v \\ &= \sum_{1 \leq i \leq h} i * d^i \end{aligned}$$

$$\begin{aligned} c.(F_{half}, G) &= \sum_{v \text{ in } G} \text{the number of certificates that appear} \\ &\quad \text{in } S.v \\ &= \sum_{1 \leq i \leq \lfloor \frac{h}{2} \rfloor} i * d^i + \sum_{\lfloor \frac{h}{2} \rfloor + 1 \leq i \leq h} d^i * (h-i) \\ &\quad + \sum_{\lfloor \frac{h}{2} \rfloor + 1 \leq i \leq h} d^i * (h-i+1) \\ &= \sum_{1 \leq i \leq \lfloor \frac{h}{2} \rfloor} i * d^i + \sum_{\lfloor \frac{h}{2} \rfloor + 1 \leq i \leq h} d^i * (2h-2i+1) \end{aligned}$$

Since

$$\sum_{\lfloor \frac{h}{2} \rfloor + 1 \leq i \leq h} d^i * (2h-2i+1) < \sum_{\lfloor \frac{h}{2} \rfloor + 1 \leq i \leq h} i * d^i$$

holds when $d \geq 2$ and $h \geq 1$,

$$c.(F_{half}, G) < c.(F_{full}, G)$$

□