Optimal dispersal of special certificate graphs

Eunjin Jung
Department of Computer Sciences
The University of Texas at Austin
Email:ejung@cs.utexas.edu

Ehab S. Elmallah
Department of Computing Science
University of Alberta
Email:ehab@cs.ualberta.ca

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Email:gouda@cs.utexas.edu

***Abstract*— We consider a network where nodes can issue certificates that identify the public keys of other nodes in the network. The issued certificates in a network constitute a directed graph, called the certificate graph of the network. The issued certificates are dispersed among the network nodes such that the following condition holds. If any node u needs to send messages to any other node v in the network, then u can use the certificates stored in both u and v to obtain the public key of v (then u can securely send messages to v). The cost of a dispersal which assigns certificates to the nodes of a network is measured by the average number of certificates that need to be stored in one node. A dispersal is optimal if its cost is minimum. In this paper, we present three algorithms and show that each algorithm computes optimal dispersals for a rich class of certificate graphs. The time complexity of each of these algorithms, when one of the algorithms is used to disperse the certificates from a given certificate graph, is $O(n^2)$ where n is the number of nodes in the input certificate graph.**

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. (Or v can send to u a shared key encrypted with $bk.v$ for further secure communication.) 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. Note that when a user acquires the public key bk_v of user v from the certificate, the user not only finds out what bk_v is, but also acquires the proof of the association that bk_v is indeed the public key of user v.

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

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

This certificate is signed 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 check the validity of 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.

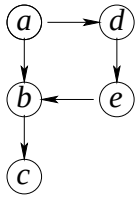

Fig. 1. A certificate graph example

Fig. 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 $bk.e$, $bk.b$, and $bk.c$, and can securely send messages to nodes e, b, and c.

The issued certificates need to be dispersed among the nodes in the network such that if a node u wishes to securely send messages to another node v, then u can obtain the public key of v from the set of certificates stored both in u and v (provided there is a directed path from u to v in the certificate graph of the network).

As an example, assume that each node in the certificate graph in Fig. 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 wishes to securely send messages to 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 Fig. 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.

The definition of certificate dispersal and its cost are given in Section II. In short, certificate dispersal is an assignment of certificates in the network to each node so that any two nodes can use the certificates stored in them to compute the public key of each other. The cost of certificate dispersal depends on the number of certificates nodes need to store. The cost of dispersal is optimal if the average number of certificates stored in the nodes is minimal.

In a previous paper [1], we introduced the concept of certificate dispersal and its cost and gave tight upper and lower bounds of the dispersal cost. Also, we identified two problems in certificate dispersal: develop efficient algorithms that compute optimal dispersals of certificate graphs and identify rich classes of certificate graphs whose dispersal costs meet the lower bound or are within a constant factor of the lower bound. We explored the first problem and presented two suboptimal dispersal algorithms: a full tree algorithm and a half tree algorithm. We also explored the second problem and identified a rich class of certificate graphs, called hierarchical star graphs. We showed that the optimal dispersal cost of each graph in this class is within a constant factor of the lower bound.

In another paper [2], we explored a variation of the first problem, namely finding an optimal dispersal of certificate chains in a certificate graph, and showed that this problem is NP-Complete. We also presented in [2] three polynomial-time algorithms that compute optimal dispersals for three special classes of chain sets.

In the current paper, we present three efficient algorithms that compute optimal dispersals for rich classes of certificate graphs.

II. 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 simple directed path of certificates (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 all distinct, is called a certificate *chain* from v_0 to v_k in G.

A *dispersal* D of a certificate graph G assigns a set of certificates in G to each node in G such that the following condition holds. The certificates in each chain from a node u to a node v in G are in the set $D.u \cup D.v$, where $D.u$ and $D.v$ are the two sets of certificates assigned by dispersal D to nodes u and v, respectively.

Let D be a dispersal of a certificate graph G. The *cost* of dispersal D, denoted $cost.D$, is the average number of certificates assigned by dispersal D to each node in G:

$$cost.D = \frac{1}{n}\Big(\sum_{v \text{ in } G} |D.v|\Big),$$

where n is the number of nodes in G.

A dispersal D of a certificate graph G is *optimal* if and only if for any other dispersal D' of the same certificate graph G, $cost.D \leq cost.D'$.

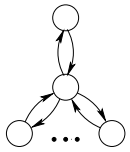

Fig. 2. A star certificate graph

For example, consider the star certificate graph in Fig. 2. This graph can be dispersed as follows. If v is the center node, then $D.v = \{\}$. Otherwise, $D.v = \{(v, \text{center node}), (\text{center node}, v)\}$. The cost of this certificate dispersal is $\frac{2(n-1)}{n}$, where n is the number of nodes in this graph. This dispersal is optimal since it meets the lower bound of certificate dispersal cost, discussed in [1].

In the following three sections, we present three special classes of certificate graphs and discuss three algorithms that compute optimal dispersals for each of these classes.

III. Optimal Dispersal of Reflexive Graphs

In this section we identify a class of certificate graphs called reflexive graphs, and give an algorithm that computes an optimal dispersal of these graphs.

A certificate graph G is called *reflexive* iff the following two conditions hold.

1) *Short Cycles* : Every simple directed cycle in G is of length 2.
2) *Reflexivity* : If there is a certificate from a node u to a node v in G, then G also has a certificate from v to u.

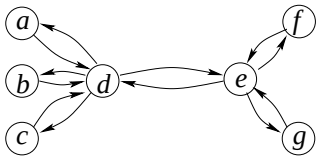

Fig. 3. An example of a reflexive certificate graph

Fig. 3 shows an example of a reflexive graph that has 7 nodes and 12 certificates. Note that there are two opposite direction certificates between the two nodes a and d, and there are no certificates between the two nodes a and b.

A nice feature of reflexive graphs is that there is a certificate chain from any node to any other node in the graph. Thus any node can get the public key of any other node in the graph and can securely send messages to it.

Let G be a reflexive graph. An *undirected version* of G is obtained from G by replacing each pair of opposite direction certificates between two nodes by an undirected edge. For example, an undirected version of the reflexive graph in Fig.3 is shown in Fig. 4.

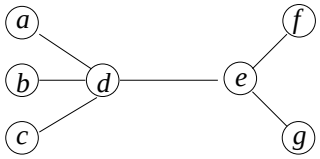

Fig. 4. An undirected version of the reflexive certificate graph in Fig. 3

Next we describe an algorithm for optimal dispersal of any reflexive graph G. Note that this algorithm operates on an undirected version G' of G.

Algorithm 1 can be applied to the reflexive certificate graph in Fig. 3 as follows. First, the undirected version of the certificate graph is constructed as shown in Fig. 4. For the edge $\{a,d\}$, the two sets $R.a$ and $R.d$ are computed as follows:

$$R.a = \{a\}, R.d = \{b,c,d,e,f,g\}$$

Since $|R.a| = 1 < 6 = |R.d|$, the two certificates (a, d) and (d,a) are stored in $D.a$. Similarly, the two certificates (b, d) and (d, b) are stored in $D.b$ and the two certificates (c, d) and (d, c) are stored in $D.c$.

For the edge $\{e,f\}$, the two sets $R.e$ and $R.f$ are computed as follows:

$$R.e = \{a,b,c,d,e,g\}, R.f = \{f\}$$

Since $|R.e| = 6 > 1 = |R.f|$, the two certificates (e, f) and (f, e) are stored in $D.f$. Similarly, the two certificates (e, g) and (g,e) are stored in $D.g$.

ALGORITHM 1

INPUT: a reflexive certificate graph G
OUTPUT: an optimal dispersal D of G

STEPS:
1: **construct** an undirected version G' of G.
2: **for** each node u in G', $D.u := \{\}$
3: **for** each undirected edge $\{u,v\}$ in G' **do**
4: **compute** the set $R.u$ that contains u and every node x where there is a simple path between x and u in G' and this path does not contain the edge $\{u,v\}$
5: **compute** the set $R.v$ that contains v and every node x where there is a simple path between x and v in G' and this path does not contain the edge $\{u,v\}$
6: **if** $|R.u| \leq |R.v|$
7: **then** for every node x in $R.u$, $D.x := D.x \cup \{(u,v),(v,u)\}$
8: **else** for every node x in $R.v$, $D.x := D.x \cup \{(u,v),(v,u)\}$

For the edge $\{d,e\}$, the two sets $R.d$ and $R.e$ are computed as follows:

$$R.d = \{a,b,c,d\}, R.e = \{e,f,g\}$$

Since $|R.d| = 4 > 3 = |R.e|$, the two certificates (d, e) and (e,d) are stored in $D.e$, $D.f$, and $D.g$.

The resulting certificate dispersal of the graph is as follows:

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

The cost of this dispersal is $(2+2+2+0+2+4+4)/7 = 16/7 \sim 2.3$ certificates per node.

***Theorem** 1:* Given a reflexive certificate graph G, the dispersal D of G computed by Algorithm 1 is optimal.

Proof: We divide the proof into two parts. First, we show that Algorithm 1 computes a dispersal. Second, we show that D is optimal.

Proof of First Part: By the definition of dispersal in Section II, if all the certificates in each chain from a node u to a node v in G are in set $D.u \cup D.v$, then D is a dispersal of G.

Consider a pair of nodes v_0 and v_k, where there is a certificate chain (v_0, v_1), (v_1, v_2), $\cdots$, (v_{k-1}, v_k) from v_0 to v_k in G. For each certificate (v_i, v_{i+1}) in this chain, the two sets $R.v_i$ and $R.v_{i+1}$ are computed by Algorithm 1 for the undirected edge $\{v_i, v_{i+1}\}$. Since there is a chain from v_0 to v_i in G, there is a simple path between v_0 and v_i in G'. Thus, $R.v_i$ contains v_0. Similarly, since there is a simple directed chain from v_{i+1} to v_k in G, there is a simple path between v_{i+1} and v_k in G'. Thus, $R.v_{i+1}$ contains v_k. By steps in line 6-8 in Algorithm 1, (v_i, v_{i+1}) is stored either in all nodes in $R.v_i$ or

in all nodes in $R.v_{i+1}$. Because $R.v_i$ contains v_0 and $R.v_{i+1}$ contains v_k, certificate (v_i, v_{i+1}) is stored either in $D.v_0$ or in $D.v_k$. Thus, every certificate (v_i, v_{i+1}) in the chain, is stored in $D.v_0 \cup D.v_k$. Therefore, the chain from v_0 to v_k is stored in the set $D.v_0 \cup D.v_k$. D is a dispersal of G.

For every pair of certificates (u,v) and (v,u) in G, an undirected edge $\{u,v\}$ is constructed in G'. The two certificates (u,v) and (v,u) are stored either in all nodes in $R.u$ or in all nodes in $R.v$, where $R.u$ and $R.v$ are the two sets computed by Algorithm 1 for the undirected edge $\{u,v\}$. By the definition of $R.u$ and $R.v$, $R.u$ contains u and $R.v$ contains v. Thus, by step iii in Algorithm 1, the two certificates (u,v) and (v,u) are either stored in $D.u$ or in $D.v$. Therefore, for every certificate in G, there is a node x in G such that this certificate is in $D.x$. The completeness condition holds.

Proof of Second Part:

Let D' be any other dispersal of a reflexive certificate graph G and let (u,v) be any directed certificate in G. The certificate (u,v) is on every directed chain from a node in $R.u$ to a node in $R.v$, where $R.u$ and $R.v$ are the two sets computed by Algorithm 1 for the undirected edge $\{u,v\}$. Therefore, D' needs to assign certificate (u,v) to every node in $R.u$ or to every node in $R.v$. In either case, D' yields a dispersal cost that is no less than the dispersal cost of D computed by Algorithm 1. ■

The complexity of Algorithm 1 is $O(en)$, where e is the number of edges in the undirected version of the input reflexive graph and n is the number of nodes in the reflexive graph. Since $e = n-1$, the complexity of this algorithm is $O(n^2)$.

Note that the star certificate graph in Fig. 2 is reflexive and so Algorithm 1 can be used to compute an optimal dispersal of this graph. Using Algorithm 1, we obtain the following certificate dispersal for this graph:

$D.v = \{\}$ if v is the center node
$D.v = \{(v,$ center node$),($center node$, v)\}$ otherwise

The cost of this certificate dispersal = $(0 + 2(n-1))/n$. From Theorem 1, we conclude that this cost is the smallest possible cost of certificate dispersal for the star certificate graph. (Thanks to Theorem 1, we no longer need to appeal to the fact that this cost meets the lower bound of certificate dispersal cost in [1] in order to reach this conclusion.)

IV. Optimal Dispersal of Biased Graphs

In this section, we present an algorithm that computes an optimal dispersal for another class of certificate graphs, called biased graphs. As discussed below, the class of biased graphs is for all practical purposes mutually exclusive from the class of reflexive graphs discussed in the previous section.

A certificate graph G is called *biased* iff it satisfies the following two conditions.

1) *Acyclicity* : G has no directed cycles.
2) *Nonredundancy* : G has at most one certificate chain from any node to any other node.

From the definitions of reflexive and biased graphs, it follows that every reflexive graph that has one or more certificates is not biased and every biased graph that has one or more certificates is not reflexive. Biased certificate graphs represent many useful certificate systems. For example, a hierarchical certificate system would typically generate a tree-shaped certificate graph. Any directed tree-shaped certificate graph is a biased certificate graph.

Note that a reflexive graph supports secure two-way communication between every two nodes in the graph, whereas a biased graph supports secure one-way communication between some two nodes in the graph. For example, consider the biased graph in Fig. 5. This graph supports secure one-way communication from node a to node b and from node a to node c, but it does not support any secure communication between the two nodes b and c.

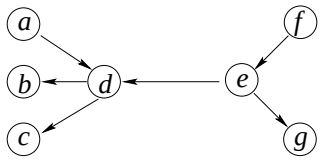

Fig. 5. A biased certificate graph

Next, we present an algorithm which computes optimal dispersals for the class of biased graphs.

ALGORITHM 2

INPUT: a biased certificate graph G
OUTPUT: an optimal dispersal D of G

STEPS:
1: **for** each node u in G, $D.u := \{\}$
2: **for** each certificate (u,v) in G **do**
3: **compute** the set $R.u$ that contains u and every node x where there is a chain from x to u in G
4: **compute** the set $R.v$ that contains v and every node x where there is a chain from v to x in G
5: **if** $|R.u| \leq |R.v|$
6: **then** for every node x in $R.u$, $D.x := D.x \cup \{(u,v)\}$
7: **else** for every node x in $R.v$, $D.x := D.x \cup \{(u,v)\}$

As an example, let us consider the application of the steps in lines 5–7 in Algorithm 2 on the certificate (a,d) in the biased graph in Fig. 5. In this case, the two sets $R.a$ and $R.d$ are computed as follows:

$$R.a = \{a\}, R.d = \{d,b,c\}$$

Thus, $|R.a| = 1 < 3 = |R.d|$ and so certificate (a,d) is added only to $D.a$.

As a second example, consider the application of the steps in lines 5–7 in Algorithm 2 on the certificate (e,g) in the biased graph in Fig. 5. In this case, the two sets $R.e$ and $R.g$ are computed as follows:

$$R.e = \{f,e\}, R.g = \{g\}$$

Thus, $|R.e| = 2 > 1 = |R.g|$ and so certificate (e,g) is added only to $D.g$.

***Theorem** 2:* Given a biased certificate graph G, the dispersal D of G computed by Algorithm 2 is optimal.

Proof: The proof is similar to that of Theorem 1. ■

V. Optimal Dispersal of Mixed Graphs

In this section, we present an algorithm which computes optimal dispersals for a third class of certificate graphs called mixed graphs. As discussed below, the class of mixed graphs contains, as proper subsets, the class of reflexive graphs discussed in Section III and the class of biased graphs discussed in Section IV.

A certificate graph G is called *mixed* iff it satisfies the following two conditions.

1) *Short Cycles* : Every simple directed cycle in G is of length 2.
2) *Nonredundancy* : G has at most one certificate chain from any node to any other node[1].

From this definition and the definitions of reflexive and biased graphs, it follows that every reflexive graph is a mixed graph and every biased graph is a mixed graph.

Fig. 6 shows an example of a mixed certificate graph. Note that in a mixed graph there can be two opposite direction certificates between two adjacent nodes. We refer to any such pair of certificates as *twins*, and we refer to each one of those certificates as the *twin certificate* of the other. Referring to the mixed graph in Fig. 6 the two certificates (a,d) and (d,a) are twins. This concept of twin certificates is utilized in the next algorithm that computes optimal dispersals for the class of mixed graphs.

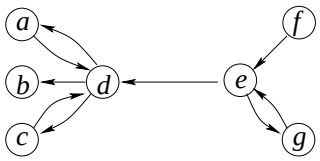

Fig. 6. A mixed certificate graph

***Theorem** 3:* Given a mixed certificate graph G, the dispersal D of G computed by Algorithm 3 is optimal.

Proof: The proof is similar to that of Theorem 1. ■

VI. Related Work

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

Architectures for issuing, storing, discovery, and validating certificates in traditional networks are presented in [3], [4], [5], [6], [7], [8], [9], and [10]. 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 [11], 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 signed 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 [12], 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 signed using the private key rk.

In [1], we proposed 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

ALGORITHM 3

INPUT: a mixed certificate graph G
OUTPUT: an optimal dispersal D of G

STEPS:
1: **for** each node u in G, $D.u := \{\}$
2: **for** each certificate (u,v) in G **do**
3: **compute** the set $R.u$ that contains u and every node x from which there is a chain to u in G and this chain does not contain the twin certificate (v,u)
4: **compute** the set $R.v$ that contains v and every node x to which there is a chain from v in G and this chain does not contain the twin certificate (v,u)
5: **if** $|R.u| \leq |R.v|$
6: **then** for every node x in $R.u$, $D.x := D.x \cup \{(u,v)\}$
7: **else** for every node x in $R.v$, $D.x := D.x \cup \{(u,v)\}$

[1]A certificate chain is a simple directed path in a certificate graph, so even when there is a cycle in a mixed certificate graph, there can be at most one certificate chain for each pair of nodes.

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. [3], [4], [5], [9], and [10] yield similar vulnerabilities in traditional networks. Recently, we have identified a metric to evaluate the damage from this type of faults. We call it "vulnerability" of the certificate graph and discuss it in more details in [13].

Perhaps the closest work to ours is [14] 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 in [1] and here are 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 [15] 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 showed in [1] 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.

VII. Conclusion

We have discussed three algorithms, each of which computes optimal dispersals for a rich class of certificate graphs. This result can be used in any network setting. However, these algorithms are particularly useful when the network is ad-hoc or when nodes are mobile. In an ad-hoc network, one cannot expect to have a central authority for storing and distributing certificates among nodes in the network. Instead, each node needs to carry a subset of the certificates in the network so that any two nodes can compute the public key of each other and securely send messages to each other (if there was a certificate chain in the original certificate graph). This result can be also used as a metric to evaluate certificate graphs, since the optimal dispersal cost is a unique property of a certificate graph.

For our future work, we would like to devise an optimal dispersal algorithm for dynamic certificate graphs.

References

[1] M. G. Gouda and E. Jung, "Certificate dispersal in ad-hoc networks," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS '04)*. IEEE, 2004.

[2] 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.

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

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

[5] 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.

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

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

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

[9] N. Li, W. H. Winsborough, and J. C. Mitchell, "Distributed credential chain discovery in trust management," *Journal of Computer Security*, vol. 11, no. 1, pp. 35–86, 2003.

[10] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti, "dRBAC: distributed role-based access control for dynamic coalition environments," in *Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS '02)*, 2002, pp. 411–420.

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

[12] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for wireless mobile networks," in *Proceedings of Ninth International Conference on Network Protocols (ICNP '01)*, 2001, pp. 251–260.

[13] M. G. Gouda and E. Jung, "Vulnerability analysis of certificate chains," *in preparation*, 2004.

[14] 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*. ACM Press, 2001, pp. 146–155.

[15] S. Capkun, L. Buttyán, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.