Optimal Dispersal of Certificate Chains

Eunjin Jung[1], Ehab S. Elmallah[2], and Mohamed G. Gouda[1]

[1] Department of Computer Sciences
The University of Texas at Austin
Austin, TX USA
[2] Department of Computing Science
University of Alberta
Edmonton, Alberta Canada

Abstract. We consider a network where users can issue certificates that identify the public keys of other users in the network. The issued certificates in a network constitute a set of certificate chains between users. A user u can obtain the public key of other user v from a certificate chain from u to v in the network. For the certificate chain from u to v, u is called the source of the chain and v is called the destination of the chain. Certificates in each chain are dispersed between the source and destination of the chain such that the following condition holds. If any user u needs to securely send messages to any other user v in the network, then u can use the certificates stored in u and v to obtain the public key of v (then u can use the public key of v to set up a shared key with v to securely send messages to v). The cost of dispersing certificates in a set of chains among the source and destination users in a network is measured by the total number of certificates that need to be stored in all users. A dispersal of a set of certificate chains in network is optimal if no other dispersal of the same chain set has a strictly lower cost. In this paper, we show that the problem of computing optimal dispersal of a given chain set is NP-Complete. We also present three polynomial-time algorithms that compute optimal dispersals for three special classes of chain sets.

1 Introduction

We consider a network where users would like to send messages securely to other users. A user who would like to send a secure message is called a *source* and a user who is intended to receive such a message is called a *destination*.

In the Internet, it is common that one source may wish to send messages to many destinations. For example, a source Alice may wish to send her credit card number securely to several destination shopping sites, say Amazon.com, eBay.com, and priceline.com. The secure communication between a source and a destination is protected by encrypting each exchanged message with a shared key only known to the source and destination.

In this network, each user u, whether source or destination, has a private key rk_u and a public key bk_u. In order for a source u to share a key sk with a destination v, u encrypts key sk using the public key bk_v of v and send the result, denoted $bk_v < u, v, sk >$, to v. Only v can decrypt this message and obtain key sk shared with u. This scenario

necessitates that u knows the public key bk_v of v. In the above example, Alice needs to know the public keys of Amazon, eBay, and priceline.

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

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

$$rk_u < u, v, bk_v >$$

This certificate is signed using the private key rk_u of 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 user that knows the public key bk_u of u can use bk_u to obtain the public key bk_v of v from the certificate from u to v. Note that when a user obtains 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.

The certificates issued by different users 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 user in the network. Each directed edge from node u to node v in the certificate graph represents a certificate from u to v in the network.

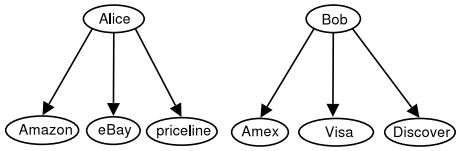


Fig. 1. A certificate graph of Alice and Bob

Fig. 1 shows a certificate graph for a network with two sources, Alice and Bob, and six destinations, Amazon, eBay, priceline, Amex, Visa, and Discover. According to this graph,

Alice issues three certificates
(*Alice*, *Amazon*), (*Alice*, *eBay*), and (*Alice*, *priceline*), and
Bob issues three certificates
(*Bob*, *Amex*),(*Bob*, *Visa*), and (*Bob*, *Discover*)

A more efficient way to support secure communication between the sources and the destinations is to introduce some intermediaries between the sources and the destinations. The number of introduced intermediaries is much smaller than the number of sources and the number of destinations. Each intermediary has its own public and private key pair. The sources know the public keys of intermediaries and the intermediaries issue certificates of the public keys of the destinations. For example, two intermediaries,

namely VeriSign and CertPlus, can be introduced between the two sources and the six destinations in Fig. 1. The result is the certificate graph in Fig. 2.

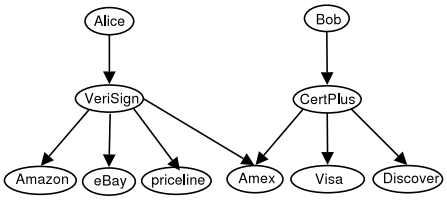


Fig. 2. A certificate graph with intermediaries

According to the certificate graph in Fig. 2, Alice needs to issue only one certificate to VeriSign and Bob needs to issue only one certificate to CertPlus. Alice can then use the two certificates (*Alice*, *VeriSign*) and $(VeriSign, Amazon)$ to obtain the public key bk_{Amazon}, and so can securely send messages to Amazon. Also, Bob can use the two certificates $(Bob, CertPlus)$ and $(CertPlus, Visa)$ to obtain the public key bk_{Visa}, and then can securely send messages to Visa.

Note that there is a certificate (*VeriSign*, *Amex*) in the certificate graph in Fig. 2 that is not needed to support secure communication between any source and any destination in Fig. 1. This redundancy is removed by specifying which "certificate chains" are being used by the sources and destinations. Certificate chains are defined as follows:

A simple path from a source *u* to a destination *v* in a certificate graph *G* is called a *chain* from *u* to *v* in *G*. *u* is the *source* of the chain and *v* is the *destination* of the chain. For users *u* and *v* in a certificate graph *G*, if *u* wishes to securely send messages to *v*, then there must be a chain from *u* to *v* in *G*. On the other hand, if there is a chain from *u* to *v*, then *u* does not necessarily wish to securely send messages to *v*. Fig. 3 shows six chains that are needed to support the secure communications between the two sources and the six destinations in Fig. 1. Since Alice does not need to securely communicate with Amex, the certificate chain (*Alice*, *VeriSign*),(*VeriSign*, *Amex*) in the certificate graph in Fig. 2 is not included in Fig. 3.

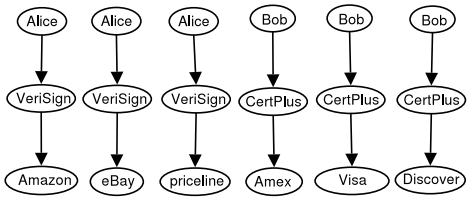


Fig. 3. Certificate chains from Fig. 2

The certificates in each chain need to be dispersed between the source and destination of the chain such that if a source *u* wishes to securely send a message to a

destination v then u can obtain the public key of v from the set of certificates stored in u and v. (Note that to "store a certificate in a user" does not necessarily mean that the user has a local copy of the certificate. Rather, it means that the user only needs to know where to find the certificate, if a need for that certificate arises, either in its local storage or in a remote location.)

For example, assume that each source in Fig. 3 stores its certificate to the corresponding intermediary, and that each destination in Fig. 3 stores the certificate from its corresponding intermediary to itself. Thus,

Alice	stores the certificate (*Alice*, *VeriSign*),
Bob	stores the certificate (*Bob*, *CertPlus*),
Amazon	stores the certificate (*VeriSign*, *Amazon*),
eBay	stores the certificate (*VeriSign*, *eBay*),
priceline	stores the certificate (*VeriSign*, *priceline*),
Amex	stores the certificate (*CertPlus*, *Amex*),
Visa	stores the certificate (*CertPlus*, *Visa*), and
Discover	stores the certificate (*CertPlus*, *Discover*)

In this case, if Alice wishes to securely send messages to priceline, then Alice can use the two certificates stored in Alice's computer and priceline website to obtain the public key of priceline and securely send the messages to priceline. Certificates that are not part of any chain are not stored because they are not needed. This is illustrated by the certificate (*VeriSign*, *Amex*), which appears in Fig. 2 but is not stored in Amex.

Dispersal of certificate chains and its cost are defined in Section 2. In Section 3, we show that finding an optimal dispersal of any set of chains is NP-Complete. Then we present three polynomial-time algorithms which compute optimal dispersal of three rich classes of chain sets.

2 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 no certificate has the same node as both its issuer and subject.

A sequence 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 *chain* from v_0 to v_k in G. Node v_0 is called the *source* of the chain and node v_k is called the *destination* of the chain. A set of chains in a certificate graph G is called a *chain set* of G.

A *dispersal* D of a chain set CS assigns a set of certificates in CS to each source node and each destination node in CS such that the following condition holds. The certificates in each chain from a source node u to a destination node v in CS 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 chain set CS. The *cost* of dispersal D, denoted $cost.D$, is the sum of cardinalities of the sets assigned by dispersal D to every source or destination node in CS.

$$cost.D = \sum_{v \text{ is a source or destination node in } CS} |D.v|$$

A dispersal D of a chain set CS is *optimal* if and only if for any other dispersal D' of the same chain set CS,

$$cost.D \leq cost.D'$$

Let c be a certificate that appears in one or more chains in a chain set CS, and let D be a dispersal of CS. The *location* of certificate c assigned by D, denoted $D.c$, is defined as a set of all nodes v such that c is in the set of certificates $D.v$.

The location $D.c$ of a certificate c assigned by a dispersal D of a chain set CS is *optimal* if and only if for any other dispersal D' of CS, $|D.c| \leq |D'.c|$.

Theorem 1. *Let D be a dispersal of a chain set CS. If D is optimal, then for every certificate c in CS the location $D.c$ is optimal.*

Proof. The proof is by contradiction. Assume that D is optimal, and there exists another dispersal D' of CS and at least one certificate c in CS such that $|D.c| > |D'.c|$.

Let c be a certificate in CS such that $|D.c| > |D'.c|$. Now define a set of certificates $D''.v$ for every node v in CS as follows.

$$D''.x := \begin{cases} D'.x & \text{if } x = c, \\ D.x & \text{if } x \neq c \end{cases}$$

The sets $D''.v$ for every node v in CS constitute a dispersal, D'', because each certificate c' other than c is assigned to the same nodes to which c' is also assigned by D and c is assigned to the same nodes to which c is assigned by D'. The cost of dispersal D'' is computed as follows.

$$cost.D'' = \sum_{v \in CS} |D''.v| = \sum_{c' \in CS, c' \neq c} |D.c'| + |D'.c|$$

By the assumption $|D.c| > |D'.c|$,

$$cost.D'' = \sum_{c' \in CS, c' \neq c} |D.c'| + |D'.c| < \sum_{c' \in CS, c' \neq c} |D.c'| + |D.c| = cost.D$$

Thus, the cost of dispersal D'' is less than the cost of dispersal D contradicting the assumption that D is an optimal dispersal.

Therefore, the location $D.c$ of c is optimal for every certificate c in CS. □

Theorem 2. *Let D be a dispersal of a chain set CS. If for every certificate c in CS the location $D.c$ is optimal, then D is an optimal dispersal of CS.*

Proof. The proof is by contradiction. Let D be a dispersal for a chain set CS and for every certificate c in CS the location $D.c$ is optimal. Also, let D' be another dispersal of CS where $cost.D' < cost.D$. By the definition of the cost of dispersal,

$$cost.D' = \sum_{c \in CS} |D'.c| < \sum_{c \in CS} |D.c| = cost.D$$

Thus, there must be at least one certificate c in CS such that $|D'.c| < |D.c|$. This contradicts the definition of an optimal location of c.

Therefore, D is an optimal dispersal of the chain set CS. □

3 NP-Completeness of optimal dispersal of chain sets

The problem of optimal dispersal of chain sets is to compute an optimal dispersal of any given chain set.

Theorem 3. *The problem of optimal dispersal of chain sets is NP-Complete.*

Proof. The proof of NP-Completeness of an optimal dispersal of a given chain set consists of two parts. First, we prove that there is a polynomial time algorithm which verifies that an assignment of certificates to nodes is a dispersal. Second, we prove that a well-known NP-Complete problem, the vertex cover problem, can be reduced in polynomial time to an optimal dispersal of a chain set.

Proof of First Part:

Given a chain set CS and a set $D.u$ for each node u in CS, we can verify whether D is a dispersal in polynomial time. For each chain from a node u to a node v in CS, reconstruct the chain from the certificates in $D.u$ and $D.v$. If all the chains in the chain set can be reconstructed, then the given set of $D.u$'s is a dispersal. The time complexity of this verification algorithm is $O(p \times n)$, where p is the number of chains in the chain set and n is the length of the longest chain in CS.

Proof of Second Part:

Consider a vertex cover of a directed graph G=(V,E). A vertex cover of G is a subset $VC \subset V$ such that if $(u,v) \in E$, then $u \in VC$ or $v \in VC$ (or both). We show that an algorithm for optimal dispersal can be used to compute a vertex cover of minimum size of any given graph G=(V,E). We consider the set of nodes $V' = V \cup \{x,y\}$, and build, for every edge (u,v) in E, a chain $(u,x);(x,y);(y,v)$. This constitutes our chain set CS.

Let D be an optimal dispersal of CS. By theorem 1, for every certificate c in CS, $D.c$ is optimal, including $c = (x,y)$. For every chain from u to v in CS, $D.u$ or $D.v$ contains (x,y) from the definition of dispersal. Therefore, u or v is in $D.(x,y)$. For every edge (u,v) in G, $D.(x,y)$ contains u or v. Therefore, $D.(x,y)$ is a vertex cover of G.

We show that $D.(x,y)$ is a vertex cover of minimum size by contradiction. Let S be a vertex cover of G where $|S| < |D.(x,y)|$. Since S is a vertex cover of G, for every edge (u,v) in G, node u or node v is in S. Let D' be a dispersal where all certificates other than (x,y) in CS remain in the same node as in D, and (x,y) stored in all the nodes in S. (D' is a dispersal since for every chain from a node u to a node v in CS, all the certificates

in the chain are in $D.u \cup D.v$.) Since we constructed D' so that all other certificates than (x,y) in the same nodes as D and (x,y) is stored in fewer nodes by D' than by D,

$$cost.D' = \sum_{c' \in CS, c' \neq c} |D.c'| + |S| < \sum_{c' \in CS, c' \neq c} |D.c'| + |D.c| = \sum_{c \in CS} |D.c| = cost.D$$

This contradicts that D is an optimal dispersal of CS. Hence, $D.(x,y)$ is a vertex cover of G of minimum size.

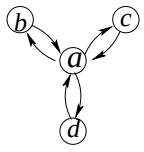
Therefore, any vertex cover problem can be reduced to an optimal dispersal of an edge in polynomial time and the optimal dispersal of the resulting chain set is equivalent to a vertex cover of minimum size in the original vertex cover problem.

An optimal dispersal problem is verifiable in polynomial time and any vertex cover problem can be reduced to an optimal dispersal problem in polynomial time. If we can find an optimal dispersal of an edge then we can find a vertex cover for any undirected graph. Therefore, an optimal dispersal problem is NP-hard. Furthermore, The vertex cover problem is a well known NP-Complete problem, so the optimal dispersal problem is NP-Complete. □

4 Optimal Dispersal of Short Chain Sets

In the previous section, we proved that computing an optimal dispersal of any chain set, which includes chains whose length is 3 or more, is NP-Complete. In this section, we show that there is a polynomial-time algorithm that computes an optimal dispersal of any chain set whose chains are all of length 2 or less.

A chain set CS is *short* if and only if the length of the longest chain in CS is at most 2. For example, consider the star certificate graph in Fig. 4(a). In this certificate graph, assume that each satellite node, b, c, or d, wishes to securely communicate with every other satellite node. Fig. 4(b) shows the resulting short chain set.



(a)

$\{\ (b,a)(a,c),$
$(c,a)(a,b),$
$(b,a)(a,d),$
$(d,a)(a,b),$
$(c,a)(a,d),$
$(d,a)(a,c)\ \}$

(b)

Fig. 4. An Example of Short Chain Set

Consider a certificate (b,a) in the example short chain set. Chains that have (b,a) are $(b,a)(a,c)$ and $(b,a)(a,d)$. So b is the source of every chain that has (b,a). Therefore, (b,a) is stored in $D.b$. After considering all the certificates in the short chain set, the certificates are dispersed by Algorithm 1 as follows:

ALGORITHM 1 : optimal dispersal of short chain sets

INPUT: a short chain set CS
OUTPUT: a dispersal D of CS

STEPS:
1: **for** each node u in CS, $D.u := \{\}$
2: **for** each certificate (u,v) in CS **do**
3: **if** there is a node x such that
the source or destination of every chain that has (u,v) is x
4: **then** $D.x := D.x \cup \{(u,v)\}$
5: **else** $D.u := D.u \cup \{(u,v)\}$, $D.v := D.v \cup \{(u,v)\}$

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

Theorem 4. *Given a short chain set CS, the dispersal D of CS computed by Algorithm 1 is optimal.*

Proof. The proof consists of two parts. First, we show that Algorithm 1 computes a dispersal D. Second, we show that D is optimal.

Proof of First Part:

By the definition of dispersal in Section 2, if all the certificates in each chain from a source node u to a destination node v in CS are in set $D.u \cup D.v$, then D is a dispersal of CS. In other words, if a certificate (u,v) is stored in the source or destination nodes of every chain that contains (u,v), then D is a dispersal.

By Algorithm 1, every certificate (u,v) is stored either in $D.x$ of some node x, or both $D.u$ and $D.v$. Since the maximum length of a chain in CS is 2, every chain that contains (u,v) starts at u or ends at v. Hence if (u,v) is stored in both $D.u$ and $D.v$ then certificate (u,v) is stored in the source or destination node of every chain that contains (u,v). If (u,v) is stored in node x, by Algorithm 1 x is either the source node or the destination node of every chain that contains (u,v). Therefore, (u,v) is stored in the source or the destination node of every chain that contains (u,v).

Proof of Second Part:

The proof is by contradiction. Let D be the dispersal of a short chain set CS computed by Algorithm 1 and D' be another dispersal of CS. Assume that $cost.D' < cost.D$. There must be at least one certificate c such that $|D'.c| < |D.c|$.

Let (u,v) be such a certificate, $|D'.(u,v)| < |D.(u,v)|$. By Algorithm 1, $|D.(u,v)|$ is either 1 (if there exists some node x that is the source or destination node of every chain that has (u,v)) or 2 (otherwise). Therefore, $|D'.(u,v)| = 1$ and $|D.(u,v)| = 2$, and there exists no node x in CS that is the source or destination node of every chain that has (u,v). By the definition of dispersal, the node w in $D'.(u,v)$ should be the source or a destination of every chain that contains (u,v) in CS. This contradicts that there exists no node x in CS such that x is the source or destination node of every chain that has (u,v).

Therefore, $cost.D \leq cost.D'$ for any dispersal D' of CS. Algorithm 1 computes an optimal dispersal of a short chain set CS. □

The time complexity of Algorithm 1 is $O(ep)$, where e is the number of certificates in the input short chain set and p is the number of chains in the chain set.

5 Optimal Dispersal of Disconnected Chain Sets

In this section, we present an algorithm which computes optimal dispersal for a class of chain sets called disconnected chain sets. A chain set CS is disconnected if and only if for every certificate c in CS, the set of source nodes of the chains that contain c and the set of destination nodes of the chains that contain c are disjoint. Fig. 5 shows an example of a disconnected chain set.

$$\{\ (d,a),\\ (a,b)(b,c),\\ (a,c)(c,d),\\ (a,b)(b,c)(c,d)(d,e)\}$$

Fig. 5. An Example of Disconnected Chain Set

(d,a) has the set of source nodes $\{d\}$ and the set of destination nodes $\{e\}$, which are disjoint. (a,b) has the set of source nodes $\{a\}$ and the set of destination nodes $\{c,e\}$, which are disjoint. Every certificate in this chain set has disjoint sets of source and destination nodes.

Disconnected chain sets represent many useful certificate systems. No strongly-connected certificate graph can produce a disconnected chain set if all possible chains are used. For example, PGP's web of trust[1] commonly results in a certificate graph with a large strongly-connected component. If all the chains are used, it is NP-Complete to compute an optimal dispersal for this strongly-connected component. In fact, not all chains have to be used. As long as the subset of chains in use forms a disconnected chain set, we can find an optimal dispersal in polynomial time.

Consider certificate (a,b) in the example disconnected chain set. G' for (a,b) is V'=$\{a,c,e\}$ and E'=$\{(a,c),(a,e)\}$. Therefore, the vertex cover of minimum size of G' is $\{a\}$. So (a,b) is stored in $D.a$. After considering all certificates in the chain set, the example disconnected chain set is dispersed by Algorithm 2 as follows:

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

Theorem 5. *Given a disconnected chain set CS, the dispersal D of CS computed by Algorithm 2 is optimal.*

Proof. The proof consists of two parts. First, we show that Algorithm 2 produces a dispersal. Second, we show that the resulting dispersal is optimal.

ALGORITHM 2 : optimal dispersal of disconnected chain sets

INPUT: a disconnected chain set CS
OUTPUT: a dispersal D of CS

STEPS:
1: **for** each node u in G, $D.u := \{\}$
2: **for** each certificate (u,v) in G **do**
3: G'=(V',E') where $V' = \{\}$ and $E' = \{\}$
4: **for** each chain from node x to node y that contains (u,v) **do**
5: V':=$V' \cup \{x,y\}$
6: E':=$E' \cup \{(x,y)\}$
7: **compute** a minimal vertex cover of the bipartite graph G'
8: **add** (u,v) to each node in the vertex cover

Proof of First Part:

Let $D.u$ be the set of certificates assigned to a node u in CS by Algorithm 2. Consider any certificate (u,v) in a chain from a source node x to a destination node y in CS. By Algorithm 2, since there is a chain from x to y that goes through (u,v), there is an edge (x,y) in G' for (u,v). By the definition of vertex cover, for edge (x,y) in G', node x or node y is in the vertex cover. Therefore, for the chain from x to y, (u,v) is stored in $D.x$ or $D.y$. This is true for all the certificates in the chain from x to y, for any chain in CS. Hence, D satisfies the dispersal condition in Section 2, so D is a dispersal of CS.

Proof of Second Part:

By Theorem 2, if we can find a dispersal D where $D.c$ of every certificate c in CS is optimal, then D is an optimal dispersal of CS. So we only need to prove that a dispersal computed by Algorithm 2 produces an optimal location of each certificate in CS. The proof is by contradiction. Assume there is another dispersal D' of CS, where $cost.D' < cost.D$. There must be at least one certificate c where $|D'.c| < |D.c|$. For every chain from a node x to a node y that contains c, $D'.c$ should contain x or y. Therefore, $D'.c$ is a vertex cover of the bipartite graph G' constructed for c, where $|D'.c| < |D.c|$. This contradicts that $D.c$ is the vertex cover of minimum size of G' by line 7 in Algorithm 2. Therefore, $D.c$ is an optimal location of c for every certificate c in CS. By Theorem 2, D is optimal. □

For each certificate (u,v), the graph G' constructed for (u,v) is a bipartite graph. It is because the set of source nodes of the chains that contain (u,v) and the set of the destination nodes of the chains that contain (u,v) are disjoint by the definition of disconnected chain set. Finding a vertex cover in a bipartite graph is a well known problem in graph theory, which takes $O(n'e')$ steps where n' is the number on nodes in G' and e' is the number of edges in G'. In the worst case $n' = n$ and $e' = p$, where n is the number of nodes in CS, and p is the number of chains in CS. Therefore, the time complexity of Algorithm 2 is $O(e \times np)$=$O(enp)$, where e is the number of certificates in CS.

6 Optimal Dispersal of Concise Graphs

In this section, we present an algorithm which computes optimal dispersal for full chain sets in concise certificate graphs. A chain set is *full* if and only if it contains all chains in a certificate graph. A certificate graph G is called *concise* if and only if it satisfies the following two conditions.

i. *Short Cycles* : Every simple directed cycle in G is of length 2.
ii. *Nonredundancy* : G has at most one chain from any node to any other node.

Fig. 6 shows an example of a concise certificate graph. Note that in a concise 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 concise graph in Fig. 6 the two certificates (a,c) and (c,a) are twins. This concept of twin certificates is utilized in the following algorithm which computes optimal dispersal of full chain set of concise certificate graphs.

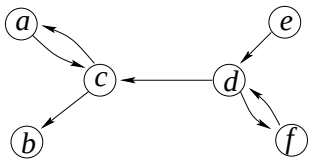


Fig. 6. A concise certificate graph

Concise certificate graphs represent many useful certificate systems. For example, a hierarchical certificate system would typically generate a tree-shaped certificate graph. Any tree-shaped certificate graph is a concise certificate graph.

Consider certificate (a,c) in the example concise certificate graph in Fig. 6. $R.a = \{a\}$ and $R.c = \{b.c\}$ so (a,c) is stored in a. After considering all the certificates in the graph, the example concise certificate graph is dispersed by Algorithm 3 as follows:

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

Theorem 6. *Given a concise certificate graph G, the dispersal D of the full chain set CS of G computed by Algorithm 3 is optimal.*

Proof. We divide the proof into two parts. First, we show that Algorithm 3 computes a dispersal D. Second, we show that D is optimal.

Proof of First Part:

We show that the certificate subsets $D.x$, computed by Algorithm 3 for every node x in G, satisfy the condition of dispersal in Section 2.

Consider a pair of nodes v_0 and v_k, where there is a chain $(v_0, v_1), (v_1, v_2), \cdots, (v_{k-1}, v_k)$ from v_0 to v_k in G. By the definition of full chain set, chain from v_0 to v_k is in CS.

ALGORITHM 3 : optimal dispersal of concise certificate graphs

INPUT: a concise certificate graph G
OUTPUT: a dispersal D of the full chain set CS 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 $R.v$, $D.x := D.x \cup \{(u,v)\}$

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 3. Since there is a chain from v_0 to v_i in G, $R.v_i$ contains v_0. Similarly, since there is a simple directed chain from v_{i+1} to v_k in G, $R.v_{i+1}$ contains v_k. By line 5-7 in Algorithm 3, (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 from v_0 to v_k is stored in $D.v_0 \cup D.v_k$. Hence, D is a dispersal of the full chain set CS of G.

Proof of Second Part:

Let D' be another dispersal of CS and (u,v) be any certificate in CS. By the definition of full chain set, if Algorithm 3 is applied to G, then certificate (u,v) is on every directed chain from a node in $R.u$ to a node in $R.v$ in CS, where $R.u$ and $R.v$ are the two sets computed by Algorithm 3 for certificate (u,v). Therefore, $D'.(u,v)$ is a superset of $R.u$ and $R.v$, so $|D'.(u,v)| \geq |R.u \cup R.v| \geq \min(|R.u|, |R.v|) = |D.(u,v)|$. This is true for any certificate (u,v) in CS, thus $cost.D'$ is no less than $cost.D$. Therefore, D computed by Algorithm 3 is optimal. □

The complexity of Algorithm 3 is $O(en)$, where e is the number of certificates in the input concise certificate graph and n is the number of nodes in the concise certificate graph.

7 Related Work

Several papers have investigated the use of certificates for confidentiality, authentication, and authorization. We summarize the results of these papers in the following paragraphs.

Architectures for issuing, storing, discovery, and validating certificates in networks are presented in [2], [3], [4], [5], [6], [7], [8], [9], and [10]. In a large scale network such as today's Internet, one cannot expect to have a central authority to issue, store, and validate all the certificates. A distributed system, where each user participates in issuing, storing, and validating certificates is desirable in such a network.

In [11] and [12], distributed architectures for issuing certificates, particularly in mobile networks, are presented.

In [11], Zhou and Haas present 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 another 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.

Both work assume that a certificate will be signed by a special private key of an authority, and distribute the private key among many servers or users. By contrast, in [13] and this paper, we propose a distributed 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 fact, many proposed certificate architectures, e.g. [2], [3], while [12] , [4], [10], and [9] yield similar vulnerabilities. Recently, we have identified a metric to evaluate the damage from this type of attacks. We call it "vulnerability" of the certificate system and discuss it in more details in [14].

In [10], Li, Winsborough, and Mitchell presented a role-based trust management language RT_0 and suggested the use of strongly typed distributed certificate storage to solve the problem of certificate chain discovery in distributed storage. However, they do not discuss how to efficiently assign certificates among the distributed storages. By contrast, our work focuses on minimizing storage overhead in certificate dispersal among the users while they have enough certificates so that there is no need for certificate chain discovery.

In [15], Ajmani, Clarke, Moh, and Richman presented a distributed certificate storage using peer-to-peer distributed hash table. This work assumes dedicated servers host a SDSI certificate directory and focuses on fast look-up service and load balancing among the servers. By contrast, our work assigns certificates to users such that there is no need for look-up and there are no dedicated certificate storage servers. Our work also focuses on efficient use of storages in all users in network.

In [16], Reiter and Stubblebine investigate how to increase assurance on authentication with multiple independent certificate chains. They introduce two types of independent chains, disjoint paths (no edge is shared by any two chains) and k-connective

paths (k certificates need to be compromised to disconnect all these paths). This paper shows that there are no polynomial-time algorithms for locating maximum sets of paths with these properties and presents approximation algorithms.

Perhaps the closest work to ours is [17] 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 obtain the public key of the other. By contrast, our work in [13] and here are based on two different conditions. First, different nodes may store different number of certificates, but the number of certificates stored in nodes 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 obtain the public key of the other (if there exists a chain between them in the chain set).

Later, the same authors have showed in [18] 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 [13] 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 certificates in the system. Our work here shows that finding an optimal dispersal of a given chain set is NP-Complete, and presents three polynomial-time algorithms which compute optimal dispersal of three classes of chain sets.

8 Conclusion

We have shown that, in general, finding an optimal dispersal of a given chain set is NP-Complete. We have also discussed three polynomial-time algorithms, each of which computes an optimal dispersal for a rich class of chain sets. In [19], we have presented more polynomial-time algorithms which compute an optimal dispersal for more classes of chain sets. This result can be used in any network setting. However, these algorithms are particularly useful when the network is large. In a large scale network such as today's Internet, one cannot expect to have a central authority for storing and distributing certificates among all users in the network. Instead, users can store a subset of certificates in the network so that any user can obtain the public key of the other whom the user wants to securely communicate with (if there was a chain in the chain set). Moreover, in a large scale network, not all certificate chains in a certificate graph are in use. Computing an optimal dispersal of a chain set instead of the full chain set of a certificate graph reduces the cost of dispersal.

This result can be also used as a metric to evaluate certificate graphs. The optimal dispersal cost is an important property of a certificate graph, since it affects the storage requirement of each node in the network. This is especially important in ad-hoc networks, where mobile nodes may be more restricted in terms of storage than stable nodes can be.

9 Acknowledgement

The authors would like to thank Jean-Philippe Martin for interesting discussions on the NP-Completeness proof.

References

1. McBurnett, N.: PGP web of trust statistics. http://bcn.boulder.co.us/ neal/pgpstat/ (1996)
2. Rivest, R.L., Lampson, B.: SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession (1996)
3. Boeyen, S., Howes, T., Richard, P.: Internet X.509 public key infrastructure operational protocols - LDAPv2. RFC 2559 (1999)
4. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet public key infrastructure online certificate status protocol - OCSP. RFC 2560 (1999)
5. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI certificate theory. RFC 2693 (1999)
6. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The keynote trust-management system version 2. RFC 2704 (1999)
7. Clarke, D., Elien, J.E., Ellison, C., Fredette, M., Morcos, A., Rivest, R.: Certificate chain discovery in SPKI/SDSI. Journal of Computer Security **9** (2001) 285–322
8. Elley, Y., Anderson, A., Hanna, S., Mullan, S., Perlman, R., Proctor, S.: Building certificate paths: Forward vs. reverse. In: Proceedings of the 2001 Network and Distributed System Security Symposium (NDSS '01). (2001) 153–160
9. Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V.: dRBAC: distributed role-based access control for dynamic coalition environments. In: Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS 02). (2002) 411–420
10. Li, N., Winsborough, W.H., Mitchell, J.C.: Distributed credential chain discovery in trust management. Journal of Computer Security **11** (2003) 35–86
11. Zhou, L., Haas, Z.J.: Securing ad hoc networks. IEEE Network **13** (1999) 24–30
12. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing robust and ubiquitous security support for wireless mobile networks. In: Proceedings of Ninth Internation Conference on Network Protocols (ICNP'01). (2001) 251–260
13. Gouda, M.G., Jung, E.: Certificate dispersal in ad-hoc networks. In: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 04), IEEE (2004)
14. Gouda, M.G., Jung, E.: Vulnerability analysis of certificate chains. *in preparation* (2004)
15. Ajmani, S., Clarke, D.E., Moh, C.H., Richman, S.: ConChord: Cooperative SDSI certificate storage and name resolution. In: LNCS 2429 Peer-to-Peer Systems: First International Workshop, IPTPS 2002. (2002) 141–154
16. Reiter, M.K., Stubblebine, S.G.: Resilient authentication using path independence. IEEE Transactions on Computers **47** (1998) 1351–1362
17. Hubaux, J.P., Buttyán, L., Capkun, S.: 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) 146–155
18. Capkun, S., Buttyán, L., Hubaux, J.P.: Self-organized public-key management for mobile ad hoc networks. IEEE Transactions on Mobile Computing **2** (2003) 52–64
19. Jung, E., Elmallah, E.S., Gouda, M.G.: Optimal dispersal of certificate chains. In: to appear in the Proceedings of the 18th International Symposium on Distributed Computing (DISC 04), Springer-Verlag (2004)