Optimal Dispersal of Certificate Chains

Eunjin (EJ) 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 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 a 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. Thus, minimizing the average number of certificates stored in any node is NP-Complete. We identify three special classes of chain sets that are of practical interests and devise three polynomial-time algorithms that compute optimal dispersals for each class. We also present two polynomial-time extensions of these algorithms for more general classes of chain sets.

Index Terms

D.4.6 Security and Privacy Protection, D.4.6.b Authentication, K.6.5 Security and Protection, K.6.5.a Authentication, certificate graph, certificate dispersal, public-key management

I. 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 $bk_v\{sk\}$, 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.

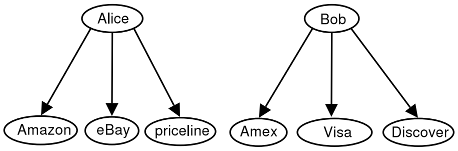


Fig. 1. A certificate graph of Alice and Bob

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.

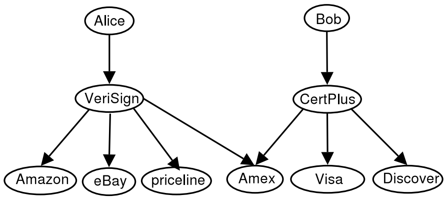


Fig. 2. A certificate graph with intermediaries

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 the 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.

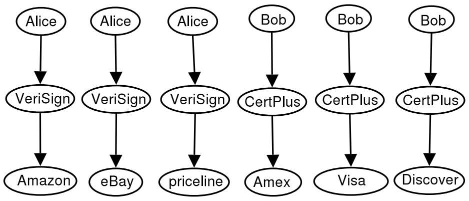


Fig. 3. Certificate chains from Fig. 2

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 II. In Section III, we show that finding an optimal dispersal of any set of chains is NP-complete. Thus it becomes of interest to characterize the special cases of practical interest where the problem can be solved efficiently, as well as effective heuristic algorithms to solve general instances of problems. Subsequently, we identify three special classes of chain sets that are of practical interests and devise three polynomial-time algorithms that compute optimal dispersals for each class. For instance, the example dispersal above reflects the certificate dispersal in Secure Socket Layer (SSL). Such chain sets are defined as "short" chain sets in Section IV, and we present an algorithm that computes an optimal dispersal of any given short chain set. We also present two extensions of these algorithms for more general classes of chain sets.

II. Certificate Dispersal

In this section, we introduce definitions and notations to describe the optimal dispersal and prove two theorems of the properties of an optimal 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. Thus, given a chain in CS, the source node u and the destination node v of the chain can find all the certificates in the chain in the set $D.u\cup D.v$. When the source node u and the destination node v need to search for a chain from u to v, then they can simply merge $D.u$ and $D.v$ to construct a certificate graph $G_{u,v}$, and search for a simple path from u to v in $G_{u,v}$. If there is a simple path from u to v in $G_{u,v}$, then this path is a certificate chain from u to v. On the other hand, if there is no path from u to v in $G_{u,v}$, then nodes u and v recognize that there was no certificate chain in the given CS.

Let D be a dispersal of a chain set CS. The *cost* of dispersal D, denoted $cost.D$, is the sum of the number of certificates in 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,

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

In other words, an optimal dispersal D of a chain set CS minimizes the average number of certificates stored in each node.

Dispersal of a chain set is useful for many types of systems. We discuss three example types of systems here.

i) Deployed systems: In a deployed system, all the certificates are dispersed among the nodes in the system before the nodes start on a particular mission. For example, consider mobile units participating in a military operation. Chains that can be used for authentication are carefully chosen and dispersed. Each unit stores the assigned set of certificates by a dispersal of chosen chains. The units are deployed in mission and when a unit needs to authenticate another unit, they do not have guarantee that any other unit will be available. Thanks to dispersal, these two nodes can use the certificates stored in each unit to find a certificate chain from one to the other. Many military applications fit in this type of systems.

ii) Client-Server systems: In a client-server system, there are only a limited number of certificate authorities that issue certificates. In such systems, it is not necessary to collect all the certificates to optimally disperse them. For example, in Secure Socket Layer (SSL) systems, VeriSign is one of the few certificate authorities. A server, for example Amazon.com, does not need to know all the certificates in the system but only stores the certificate $(Amazon.com, VeriSign)$. This is an optimal dispersal (more details are in Section IV) of this SSL system.

iii) Evolving systems: In an evolving system where certificates may be issued and revoked during the execution of the system, the system can start with an optimal dispersal of such system and gradually diverge from the dispersal. Even when the system diverges from its dispersal, it is still beneficial to start with an optimal dispersal as long as the changes in certificates are not a major portion of certificates in the system. Moreover, the dynamic dispersal protocol in [1] disperses newly issued certificates and revocation certificates so that the system stabilizes back to dispersal.

Let (u, v) 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 set* of certificate (u, v) assigned by D, denoted $D(u, v)$, is defined as a set of all nodes x such that (u, v) is in the set of certificates $D.x$. It is straightforward to show that the cost of dispersal D equals $\sum_{(u,v)\in CS} |D(u, v)|$.

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

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

Proof: The proof is by contradiction. Assume that D is optimal, and there exists another dispersal D' of CS where for some certificate (u, v) in CS, $|D(u, v)| > |D'(u, v)|$.

Now consider the following assignment of certificates to each node in CS.

$$D''(x, y) := \begin{cases} D'(x, y) & \text{if } (x, y) = (u, v), \\ D(x, y) & \text{if } (x, y) \neq (u, v) \end{cases}$$

Note that D'' is a dispersal of CS. This is true because for any chain from a node i to another node j in CS, all the certificates in the chain are in $D''.i \cup D''.j$. Consider a certificate (x, y) in the chain from i to j in CS, where $(x, y) \neq (u, v)$. $D(x, y)$ contains node i or node j by the definition of dispersal, so $D''(x, y)$ contains node i or node j. In other words, any certificate (x, y) in a chain from node i to node j in CS, where $(x, y) \neq (u, v)$, is in $D''.i \cup D''.j$. Similarly, for certificate (u, v), if (u, v) is in a chain from i to j in CS, $D'(u, v)$ contains node i or node j by the definition of dispersal, so $D''(u, v)$ contains node i or node j. In other words, if certificate (u, v) is in a chain from node i to j in CS, then (u, v) is in $D''.i \cup D''.j$. Therefore, for any given chain from a node i to another node j in CS, all the certificates in the chain are in $D''.i \cup D''.j$. Thus, D'' is a dispersal of CS.

The cost of dispersal D'' is computed as follows.

$$cost.D'' = \sum_{v\in CS} |D''.v| = \left(\sum_{(x,y)\in CS, (x,y)\neq(u,v)} |D(x, y)| \right) + |D'(u, v)|$$

By the assumption $|D'(u,v)| < |D(u,v)|$,

$$\begin{aligned} cost.D'' &= \Bigg(\sum_{(x,y)\in CS,(x,y)\neq(u,v)} |D(x,y)| \Bigg) + |D'(u,v)| \\ &< \Bigg(\sum_{(x,y)\in CS,(x,y)\neq(u,v)} |D(x,y)| \Bigg) + |D(u,v)| = cost.D \end{aligned}$$

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 set $D(u,v)$ assigned by an optimal dispersal D is optimal for every certificate (u,v) in CS.

***Theorem** 2:* Let D be a dispersal of a chain set CS. If for every certificate (u,v) in CS the location set $D(u,v)$ 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 (u,v) in CS the location set $D(u,v)$ is optimal. Also, let D' be another dispersal of CS where $cost.D' < cost.D$. By the definition of the cost of dispersal,

$$\sum_{(u,v)\in CS} |D'(u,v)| = cost.D' < cost.D = \sum_{(u,v)\in CS} |D(u,v)|$$

Thus, there must be at least one certificate (u,v) in CS such that $|D'(u,v)| < |D(u,v)|$. This contradicts the definition of an optimal location set of (u,v). ■

Therefore, if $D(u,v)$ is optimal for every certificate (u,v) in a chain set CS, then D is an optimal dispersal of CS.

III. NP-COMPLETENESS OF OPTIMAL DISPERSAL OF CHAIN SETS

In this section, we show that the chain dispersal problem is NP-Complete by a reduction from the vertex cover problem. For convenience, these two problems are described below.

- The Vertex Cover (VC) Problem: Given a connected graph G and a positive integer k, we ask if there exists a vertex cover of size $\leq k$. Any instance of this problem can be represented by the pair (G,k). For directed graphs, the VC problem can be defined similarly by ignoring the directions associated with the arcs; the resulting problem on directed graphs remains NP-complete.

- The Certificate Dispersal (CD) Problem: Given a chain set CS and a positive integer m, we ask if there exists a dispersal D of CS such that $cost.D \leq m$. Any instance of this problem can be represented by the pair (CS, m).

***Theorem** 3:* CD is NP-Complete.

Proof: First, we show that CD is in NP. Given an instance (CS, m) of CD, and a dispersal D of CS with $cost.D \leq m$, one can verify in polynomial-time that indeed D is a dispersal of CS and $cost.D \leq m$. To verify that D is a dispersal of CS, one checks that all the certificates in each chain from a node u to another node v in CS are in $D.u \cup D.v$. Once D is verified as dispersal, $cost.D$ is computed as the sum of $|D.u|$ for each node u in CS and can be compared to m. The time complexity of this verification step 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.

Second, we show that VC reduces to CD in polynomial-time. Given an instance (G, k) of VC, we construct an instance (CS, m) of CD such that the CD instance has a yes answer if and only if the given VC has a yes answer. The construction is as follows:

i) For each edge (u, v) in G, CS has a chain $(u, x)(x, y)(y, v)$ of length 3.

ii) Let n^+ be the number of nodes that have outgoing edges in G, and n^- be the number of nodes that have incoming edges in G. Set $m = n^+ + n^- + k$.

(CD $\Leftarrow$ VC) We now show that if the instance (G, k) of VC has a yes answer, then the corresponding instance (CS, m) of CD has a yes answer. Let X be a vertex cover of G, where $|X| \leq k$. For each node u in the cover X, assign certificate (x, y) in CS to $D.u$. For each node u in G, if there exists (u, x) in CS, then assign certificate (u, x) to $D.u$. For each node v in G, if there exists (y, v) in CS, then assign certificate (y, v) to $D.v$. In the following two steps, we prove that D is a dispersal of CS whose cost is at most m.

i) D is a dispersal of CS: For any chain in CS from a node u to a node v, the chain consists of three certificates (u, x), (x, y), and (y, v). Certificate (u, x) is stored in $D.u$ and certificate (y, v) is stored in $D.v$. For certificate (x, y), (x, y) is stored in every node in the vertex cover of G. By the definition of the vertex cover, for each edge

(u, v) in G, the vertex cover contains node u or node v. Certificate (x, y) is assigned to every node in the vertex cover of G, so (x, y) is stored in $D.u$ or $D.v$. Thus, every certificate in the chain from u to v is stored in $D.u \cup D.v$, as required by the definition of dispersal.

ii) $cost.D \leq m$: For each node u in G that has any outgoing edges, there is certificate (u, x) in CS that is assigned only to node u by D. Similarly, for each node v in G that has any incoming edges, there is certificate (y, v) in CS that is assigned only to node v by D. For certificate (x, y), (x, y) is assigned to all the nodes in the vertex cover, so (x, y) is assigned to at most k nodes. In total, $cost.D$ is at most $m = (k + n^+ + n^-)$.

The above argument shows that D is a dispersal of constructed CS and $cost.D \leq m$. This proves that if an instance of VC (G, k) has a yes answer, then the corresponding instance of CD (CS, m) has a yes answer.

(CD $\Rightarrow$ VC) We now show that if the constructed instance (CS, m) of CD has a yes answer, then the given instance (G, k) of VC has a yes answer. Let D be a dispersal of CS, where $cost.D \leq m$. For every edge (u, v) in G, there is chain $(u, x)(x, y)(y, v)$ in CS. For certificates (u, x) and (y, v), they will be assigned to at least one node, so $|D(u, x)| \geq 1$ and $|D(y, v)| \geq 1$. The number of such (u, x) certificates is n^+ and the number of such (y, v) certificates is n^-. So certificate (x, y) is assigned to at most k nodes, where k is $m - n^+ - n^-$. In other words, $|D(x, y)| \leq k$.

Now, for each edge (u, v) in G, there is chain $(u, x)(x, y)(y, v)$ in CS, and (x, y) is stored in $D.u \cup D.v$. In other words, for each edge (u, v) in G, the location set of $D(x, y)$ contains node u or node v. Therefore, the location set of $D(x, y)$ is a vertex cover of G. The size of the location set $D(x, y)$ is at most k, so the size of the vertex cover is at most k, and the instance (G, k) of VC has a yes answer.

In conclusion, the above proof shows that CD is in NP and VC reduces to CD in polynomial-time. Therefore, CD is NP-Complete. ■

In the light of the above complexity result, it becomes of importance to identify special classes of chain sets of practical interest for which the problem can be solved efficiently.

This direction is pursued in the following cases.

i) *Short chain sets*: In Section IV, we start by investigating the class of chain sets, where each chain is of length at most 2. This class of chain sets is the one currently being used in the Secure Socket Layer (SSL) protocol. Recall that the chain set in the example in Fig. 3 in Section I falls into this class.

ii) *Disconnected chain sets*: In Section V, we investigate the class of chain sets where for a given certificate, no node can be both the source and the destination of any chain that contains this certificate. This reflects a system where the authentication is needed in an asymmetric manner. For example, when there are clients and servers in the system, one can imagine that clients would use certificates to authenticate servers, while servers would use passwords to authenticate clients. Such asymmetric systems can be represented as this class of chain sets.

iii) *Concise graphs*: In Section VI, we investigate the class of chain sets where the chains are derived from acyclic certificate graphs. This class reflects systems where the need for authentication is uni-directional. For example, any hierarchical system where a lower level user is authenticated by a higher level user, but not the other way around, would be represented by an acyclic certificate graph.

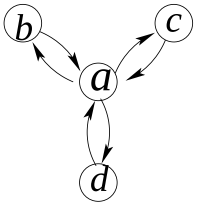
For all these three classes of chain sets, we present polynomial-time algorithms that compute optimal dispersals of chain sets in each class and prove their optimality.

Also below, we identify two classes of parameterized chain sets that are defined using an integer parameter k. In the first class, each chain set has at most k chains with 3 or more certificates. In the second class, each chain set has at most k nodes that may act both as sources and destinations. For both classes, we obtain polynomial-time algorithms that compute optimal dispersals when k is fixed.

IV. 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 at most 2. This class of chain sets is currently in use in the Internet in Secure Socket Layer (SSL).

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)

$$\begin{aligned}\{\quad &(b,a)(a,c), \quad (d,a)(a,b),\\ &(c,a)(a,b), \quad (c,a)(a,d),\\ &(b,a)(a,d), \quad (d,a)(a,c)\}\end{aligned}$$

(b)

Fig. 4. An Example of Short Chain Set

ALGORITHM 1 : optimal dispersal of short chain sets

INPUT: a short chain set CS

OUTPUT: a dispersal D of CS

STEPS:

```
for each node $u$ in $CS$, $D.u := \{\}$
for each certificate $(u, v)$ in $CS$ do
    if there is a node $x$ such that
    the source or destination of every chain that has $(u, v)$ is $x$
        then add $(u, v)$ to $D.x$
        else add $(u, v)$ to both $D.u$ and $D.v$
```

Algorithm 1 computes an optimal dispersal of a short chain set. Consider the certificate (b, a) in the example short chain set in Fig. 4. 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, Algorithm 1 assigns

(b, a) to $D.b$. After considering all the certificates in the short chain set, the optimal dispersal computed by Algorithm 1 as follows:

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

***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 II, 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, then 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 (u, v) such that $|D'(u, v)| < |D(u, v)|$.

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.

V. Optimal Dispersal of Disconnected Chain Sets

In this section, we identify a special class of chain sets and present an algorithm that computes an optimal dispersal for this class of chain sets in polynomial-time. A chain set CS is *disconnected* if and only if for every certificate (u, v) in CS, the set of source nodes of the chains that contain (u, v) and the set of destination nodes of the chains that contain (u, v) are disjoint. This reflects a system where the authentication is performed in an asymmetric manner. For example, when there are clients and servers in the system, one can imagine that clients would use certificates to authenticate servers, while servers would use passwords to authenticate clients. Such asymmetric systems can be represented as disconnected chain sets. Fig. 5 shows an example of a disconnected chain set.

$$\begin{array}{ll} \{ & (d, a), \\ & (a, b)(b, c), \\ & (a, c)(c, d), \\ & (a, b)(b, c)(c, d)(d, e)\} \end{array}$$

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.

ALGORITHM 2 : optimal dispersal of disconnected chain sets

```
INPUT: a disconnected chain set $CS$
OUTPUT: a dispersal $D$ of $CS$
STEPS:
**for** each node $u$ in $G$, $D.u := \{\}$
**for** each certificate $(u, v)$ in $G$ **do**
    $G'$=$(V', E')$ where $V' = \{\}$ and $E' = \{\}$
    **for** each chain from node $x$ to node $y$ that contains $(u, v)$ **do**
        add nodes $x$ and $y$ to $V'$
        add $(x, y)$ to $E'$
    **compute** a minimal vertex cover of the bipartite graph $G'$
    **add** $(u, v)$ to each node in the vertex cover
```

Algorithm 2 computes an optimal dispersal of a disconnected chain set. Consider certificate (a, b) in the example disconnected chain set in Fig. 5. Algorithm 2 constructs a bipartite graph G' for certificate (a, b), where $G' = (V', E')$, V'=$\{a, c, e\}$, and E'=$\{(a, c), (a, e)\}$. The vertex cover of minimum size of G' is $\{a\}$. Thus, (a, b) is stored in $D.a$. After considering all certificates in the chain set, the example disconnected chain set is optimally dispersed by Algorithm 2 as follows:

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

***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.

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 II, so D is a dispersal of CS.

Proof of Second Part:

By Theorem 2, if we can find a dispersal D where $D(u, v)$ of every certificate (u, v) 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 set 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 (u, v) where $|D'(u, v)| < |D(u, v)|$. For every chain from a node x to a node y that contains (u, v), $D'(u, v)$ should contain x or y. Therefore, $D'(u, v)$ is a vertex cover of the bipartite graph G' constructed for (u, v), where $|D'(u, v)| < |D(u, v)|$. This contradicts that $D(u, v)$ is the vertex cover of minimum size of G' by line 7 in Algorithm 2. Therefore, $D(u, v)$ is an optimal location set of (u, v) for every certificate (u, v) 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.

VI. Optimal Dispersal of Concise Graphs

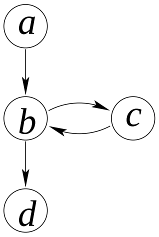
In this section, we present an algorithm that computes optimal dispersal for chain sets "derivable" from a class of certificate graphs called concise certificate graphs. A certificate graph G is called *concise* if and only if it satisfies the following two conditions.

i) *Short Cycles* : Every simple directed cycle in G is of length 2.

ii) *Non-redundancy* : G has at most one chain from any node to any other node.

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.

Fig. 6(a) 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. In the concise graph in Fig. 6(a), the two certificates (b, c) and (c, b) are twins.



$$\begin{aligned} \{ \quad & (a, b),\ (b, c),\ (c, b),\ (b, d), \\ & (a, b)(b, c), \\ & (a, b)(b, d), \\ & (c, b)(b, d)\ \} \end{aligned}$$

(a) (b)

Fig. 6. An Example of Concise Certificate Graph and Derivable Chain Set

A chain set is *derivable* from some certificate graph G if and only if the chain set consists of all the certificate chains in G. For example, the chain set in Fig. 6(b) is derivable from the certificate graph in Fig. 6(a).

Algorithm 3 computes an optimal dispersal of a concise certificate graph. Consider certificate (b, c) in the example concise certificate graph in Fig. 6(a). Algorithm 3 computes the set of nodes from which there is a chain to b, denoted $R.b$, as $\{a, b\}$. Also, Algorithm 3

ALGORITHM 3 : optimal dispersal of concise certificate graphs

```
INPUT: a concise certificate graph $G$
OUTPUT: a dispersal $D$ of the chain set $CS$ derivable from $G$
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 $R.u$, add $(u, v)$ to $D.x$
7:         else for every node $y$ in $R.v$, add $(u, v)$ to $D.y$
```

computes the set of nodes to which there is a chain from c, denoted $R.c$ as $\{c\}$. $|R.b| > |R.c|$, so (b, c) is stored in c. After considering all the certificates in the graph, the example concise certificate graph is optimally dispersed by Algorithm 3 as follows:

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

***Theorem** 6:* Given a concise certificate graph G, the dispersal D of the chain set CS derivable from 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 II.

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 the derivable chain set, the chain from v_0 to v_k is in CS. 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 chain set CS derivable from G.

Proof of Second Part: The proof is by contradiction. Let D' be another dispersal of CS where $cost.D' < cost.D$. Then there must be such a certificate (u, v) that $|D'(u, v)| < |D(u, v)|$. By the definition of dispersal, (u, v) needs to be stored in $D'.x \cup D'.y$ for every chain from x to y that contains (u, v). By the definition of derivable chain set, certificate (u, v) is used in every directed chain from any node x in $R.u$ to any node y in $R.v$, where $R.u$ and $R.v$ are the two sets computed by Algorithm 3 for certificate (u, v). In other words, $|D'(u, v)| \geq min(|R.u|, |R.v|)$. Since $|D(u, v)| = min(|R.u|, |R.v|)$, $|D'(u, v)| \geq |D(u, v)|$. This contradicts the assumption of $|D'(u, v)| < |D(u, v)|$.

Therefore, D computed by Algorithm 3 is optimal. ∎

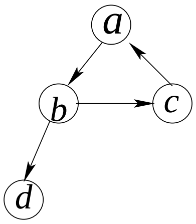
The complexity of Algorithm 3 is $O(en)$, where e is the number of certificates in the input concise certificate graph and n is the number of nodes in the concise certificate graph.

VII. Optimal Dispersal of *k*-long Chain Sets

In Section III, we showed that computing an optimal dispersal of any chain set, which includes chains of length 3 or more, is NP-complete. If all the chains in a chain set are of length at most 2, i.e. if the chain set is short, then we can use Algorithm 1 in Section IV to compute an optimal dispersal of the short chain set. In this section, we consider a more general class of chain sets where there are a fixed number k, $k \geq 1$, of chains of length greater than 2. Consideration of such chain sets is motivated, for instance, by the following example.

Consider a hierarchical network made of a number of autonomous systems. Certificate chains within any single autonomous system are expected to be short, whereas certificate chains that span multiple autonomous systems are expected to be long. The chain set of these autonomous systems contain mostly short *intra*-chains, but may contain a fixed number of long *inter*-chains. Our main result here is a polynomial-time algorithm that computes an optimal dispersal for such chain set for fixed k.

In this section, we present Algorithm 4 that computes an optimal dispersal of a chain set where there are k chains of length greater than 2 for some constant k. We call such sets k*-long* chain sets. Roughly speaking, our general strategy is to consider all possible ways of assigning certificates that appear in long chains to the relevant source and destination nodes, and then handling the remaining short chains with the aid of Algorithm 1. To develop some initial intuition, first we show how to compute an optimal dispersal of an example 1-long chain set in Fig. 7(b), and then we show how to generalize for k-long chain sets.



(a)

$$\begin{aligned} \{ \quad & (a,b)(b,c), \\ & (b,c)(c,a), \\ & (c,a)(a,b), \\ & (c,a)(a,b)(b,d) \ \} \end{aligned}$$

(b)

Fig. 7. An Example of 1-Long Chain Set

Let CS be the 1-long chain set in Fig. 7(b), which is a chain set of the certificate graph in Fig. 7(a). There is one long chain $(c,a)(a,b)(b,d)$ and three other short chains. There are three types of certificates in this chain set.

i) Certificates used only in long chains: for example, (b,d).

A certificate of this type can be dispersed either to the source or to the destination of each long chain that contains this certificate. For example, certificate (b,d) in CS is used only in the long chain and needs to be dispersed either to c or to d. This certificate is not used in any other chains, so it does not change the cost of dispersal whether it is dispersed to c or d.

ii) Certificates used only in short chains: for example, (b, c).

For certificates of the second type, we can use Algorithm 1 in Section IV to disperse such certificates. For example, certificate (b, c) is dispersed to node a by Algorithm 1.

iii) Certificates used in both long and short chains: for example, (a, b), (c, a).

Dispersing a certificate of the third type needs to consider every possible assignment of this certificate among sources and destinations of long chains. For example, certificate (a, b) is used in three chains, $(a, b)(b, c)$, $(c, a)(a, b)$ and $(c, a)(a, b)(b, d)$. If we choose to disperse (a, b) to the source c of long chain, then we do not need to disperse (a, b) to any other node in CS, since c happens to be source or destination of all the short chains that contain (a, b). By contrast, if we choose to disperse (a, b) to the destination d of long chain, then we need to disperse (a, b) to other nodes than d since d is neither source nor destination of two short chains $(a, b)(b, c)$ and $(c, a)(a, b)$. In other words, $D(a, b)$ could be either $\{c\}$ or $\{a, b, d\}$, depending on whether (a, b) is assigned to the source or the destination of the long chain. This shows that for each certificate of the third type that is used in both long and short chains, in each assignment of this certificate in sources and destinations of long chains, we need to check which short chains still needs dispersal of this certificate.

After considering all three types of certificates in CS, the resulting optimal dispersal of CS in Fig. 7(b) becomes as follows:

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

To extend this solution for 1-long chain set to k-long chain set, we need to define a terminal set of a chain set. A *terminal* set of a chain set CS is a subset of nodes in CS that consists of the source or destination of each chain in CS. For example, the four nodes a, b, c, c are the sources of all four chains in the chain set in Fig. 7(b), so $\{a, b, c\}$ is a terminal set of this chain set. Algorithm 4 computes an optimal dispersal of k-long chain sets using this terminal set.

ALGORITHM 4 : optimal dispersal of k-long chain sets

```
INPUT: a $k$-long chain set $CS$
OUTPUT: a dispersal $D$ of $CS$
STEPS:
for each node $u$ in $CS$, $D.u := \{\}$
for each certificate $(u, v)$ in $CS$ do
    compute the chain set $LS$ of all long chains that contain $(u, v)$ in $CS$
    for each possible terminal set $X$ of $LS$
        for each node $w$ in $CS$, if $w \in X$ then $D_X.w := \{(u, v)\}$ else $D_X.w := \{\}$
        compute the chain set $S$ of all the chains that contain $(u, v)$
        and their sources and destinations are not in $X$
        run Algorithm 1 on $S$ and add the resulting location set of $(u, v)$ to $D_X$
    find $D_X$ with the minimal cost
    for each node $u$ in $CS$, add $D_X.u$ to $D.u$
```

Consider (c, a) in the example chain set in Fig. 7(b). The set of all long chains that contain (c, a), denoted LS in Algorithm 4, is $\{(c, a)(a, b)(b, d)\}$. For a terminal set $\{c\}$, (c, a) is dispersed to node c and the set of remaining short chains, denoted S in Algorithm 4, becomes $\{(b, c)(c, a)\}$. There is node b that is the source of every chain in S, so (c, a) is dispersed to node b. The resulting dispersal of (c, a), $\{b, c\}$, is an optimal location set of (c, a). After considering every certificate, the dispersal of the example chain set in Fig. 7(b) computed by Algorithm 4 becomes the same with the dispersal above, and this dispersal is optimal. Theorem 7 shows that Algorithm 4 computes an optimal dispersal of a given k-long chain sets.

***Theorem** 7:* Given a k-long chain set CS, the dispersal D of the chain set CS computed by Algorithm 4 is optimal.

Proof: We divide the proof into two parts. First, we show that Algorithm 4 computes a dispersal D. Second, we show that D is optimal.

Proof of First Part:

We show that the certificate subsets $D.u$, computed by Algorithm 4 for every node u in CS, satisfy the condition of dispersal in Section II.

Consider a certificate (u, v). Algorithm 4 computes the chain set LS of for all the long chains that contain (u, v). Algorithm 4 stores (u, v) in every node in a terminal set of LS. By the definition of a terminal set, (u, v) is stored in either source or destination of each long chain in LS. For all the remaining short chains that contain (u, v) in CS, by line 7-9 in Algorithm 4 (same as line 3-5 in Algorithm 1), (u, v) is stored either in $D.w$ for some node w or in $D.u$ and $D.v$. (The rest of proof is same with the optimality proof of Algorithm 1.) For each remaining short chain, the 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 remaining chain that contains (u, v). If (u, v) is stored in node w, then by Algorithm 4, then w is either the source node or the destination node of every remaining chain. Therefore, (u, v) is stored in the source or the destination node of every chain that contains (u, v). This is true for any certificate (u, v) in CS. Hence, D is a dispersal of the chain set CS.

Proof of Second Part:

The proof is by contradiction. Let D be the dispersal of a k-long chain set CS computed by Algorithm 4 and D' be another dispersal of CS. Assume that $cost.D' < cost.D$. There must be at least one certificate (u, v) such that $|D'(u, v)| < |D(u, v)|$.

There are three cases of (u, v):

i) (u, v) is a certificate used only in long chains.

ii) (u, v) is a certificate used only in short chains.

iii) (u, v) is a certificate used in both long and short chains.

For case i), Algorithm 4 considers every possible terminal set X of the long chains that contain (u,v). Therefore, the resulting $|D(u,v)| = \min_X |D_X(u,v)|$. By the definition of the terminal set, $D'(u,v)$ has to be a terminal set of the long chains that contain (u,v). In other words, $|D'(u,v)| \geq \min_X |D_X(u,v)| = |D(u,v)|$. Therefore, $|D(u,v)| \leq |D'(u,v)|$

For case ii), Algorithm 4 computes an optimal dispersal of the short chains containing (u,v). The proof is same as the optimality proof of Algorithm 1 for short chain sets. Therefore, $|D(u,v)| \leq |D'(u,v)|$.

For case iii), find a terminal set X of the long chains that contain (u,v), such that $X \subset D'(u,v)$. Since Algorithm 4 considers every possible terminal set of the long chains that contain (u,v), it also computes $D_X(u,v)$ for the found terminal set X, where $X \subset D_X(u,v)$. For the remaining short chains in S, since the sources and destinations of the short chains in S are not in X, so $D'(u,v) \setminus X$ should contain source or destination of each chain in S. Also, Algorithm 1 computes an optimal location set of (u,v) in S. Therefore, $|D_X(u,v) \setminus X| \leq |D'(u,v) \setminus X|$. Since $X \subset D'(u,v)$ and $X \subset D_X(u,v)$, $|D_X(u,v)| \leq |D'(u,v)|$. $|D(u,v)| = \min_X |D_X(u,v)|$, so $|D(u,v)| \leq |D'(u,v)|$.

In all three cases, $|D(u,v)| \leq |D'(u,v)|$, which contradicts the assumption of $|D(u,v)| > |D'(u,v)|$. Therefore, dispersal D computed by Algorithm 4 is optimal. ■

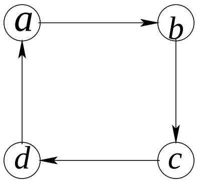
The time complexity of this algorithm is $O(2^k \times ep)$, where k is the number of long chains in CS, e is the number of certificates in CS, and p is the number of chains in CS. This complexity is computed as follows: the number of terminal sets for k long chains is $O(2^k)$, and for each terminal set, the number of short chains to consider is $O(p)$. We repeat this procedure for e certificates. Since k is a constant, the time complexity becomes $O(ep)$.

VIII. Optimal Dispersal of k-Connected Chain Sets

In Section V, we presented Algorithm 2 that computes an optimal dispersal of a disconnected chain set. In this section, we investigate more general class of chain sets where there are at most k nodes in the intersection of the source set and the destination set of each certificate in a chain set. We call such chain sets *k-connected* chain sets. This class of chain sets models a client-server system that uses two different authentication methods.

As discussed in Section V, in some client-server systems, clients authenticate servers via certificates, whereas servers authenticate clients via other means, e.g. passwords. However, there may be a few mutual authentications via certificates between servers. These certificates used by servers may have non-empty intersection of the source and destination sets. Such client-server systems can be represented as k-connected chain sets.

Fig. 8(b) shows an example of 1-connected chain set, which is a chain set of the certificate graph in Fig. 8(a). For certificate (a,b), the sources of the chains that contain (a,b) are $\{a,c\}$ and the destinations of such chains are $\{b,c,d\}$. The intersection of two sets is $\{c\}$. Similarly, the cardinality of the intersection set is at most 1 for every certificate in this chain set, so the chain set in Fig. 8(b) is 1-connected.



$$\begin{aligned}\{\quad & (a,b)(b,c),\\ & (a,b)(b,c)(c,d),\\ & (b,c)(c,d)(d,a),\\ & (c,d)(d,a)(a,b)\ \}\end{aligned}$$

(a) (b)

Fig. 8. An Example of 1-Connected Chain Set

Assume that (a,b) is stored in $D.c$ in some dispersal D of this chain set. The remaining chain to be dispersed is $(a,b)(b,c)(c,d)$. Certificate (a,b) can be stored either in $D.a$ or in $D.d$, either of which makes no difference in the dispersal cost. Or, assume that (a,b) is not stored in $D'.c$ in some dispersal D' of this chain set. Certificate (a,b) needs to be stored in $D'.a$ and $D'.b$. We can repeat this process for each certificate to find the dispersal as follows:

$$\begin{aligned}\{D.a = \ & \{(a,b),(b,c)\}, \quad D.b = \{(c,d),(d,a)\},\\ D.c = \ & \{(a,b)\}, \qquad\qquad D.d = \{(c,d)\} \qquad\qquad \}\end{aligned}$$

This is also an optimal dispersal of this 1-connected chain set.

To extend this solution for 1-connected chain set to k-connected chain set, we need to define an *intersection* set of a certificate. An intersection set of a certificate (u, v) in a chain set CS is a set of nodes that appear both in the set of sources and the set of destinations of the chains that contain (u, v). For certificate (a, b) in Fig. 8(b), the sources of the chains that contain (a, b) are $\{a, c\}$ and the destinations of such chains are $\{b, c, d\}$. The intersection of two sets is $\{c\}$, so $\{c\}$ is the intersection set of (a, b). Algorithm 5 computes an optimal dispersal of k-connected chain sets using this intersection set.

ALGORITHM 5 : optimal dispersal of k-connected chain sets

```
INPUT: a $k$-connected chain set $CS$
OUTPUT: a dispersal $D$ of $CS$
STEPS:
for each node $u$ in $CS$, $D.u := \{\}$
for each certificate $(u, v)$ in $CS$ do
    compute the intersection set $IS$ of $(u, v)$
    for each subset $X$ of $IS$
        for each node $w$ in $CS$, if $w \in X$ then $D_X.w := \{(u, v)\}$ else $D_X.w := \{\}$
        compute the chain set $S$ of all the chains that contain $(u, v)$
        and their sources and destinations are not in $X$
        for each chain from $y$ to $z$ in $S$
            if $y \in IS \setminus X$ then add $(u, v)$ to $D_X.z$ and remove the chain from $S$
            if $z \in IS \setminus X$ then add $(u, v)$ to $D_X.y$ and remove the chain from $S$
        run Algorithm 2 on $S$ and add the resulting location set of $(u, v)$ to $D_X$
    find $D_X$ with the minimal cost
    for each node $u$ in $CS$, add $D_X.u$ to $D.u$
```

The proof of the optimality of this algorithm is straightforward. Since this algorithm considers every possible subset of the intersection set, it is guaranteed to find the optimal location set of each certificate. By Theorem 2, the dispersal computed by this algorithm is optimal.

The time complexity of this algorithm is $O(2^k \times enp)$, where k is the tight upper bound of the number of nodes in intersection sets of all the certificates in CS, n is the number of nodes in CS, e is the number of certificates in CS, and p is the number of chains in CS. Since there are at most k nodes in the intersection set of each certificate, there are at most 2^k subsets of the intersection set. For each subset, we run Algorithm 2, whose complexity is $O(enp)$. Therefore, the total time complexity becomes $O(2^k enp)$. Since k is a constant, the time complexity becomes $O(enp)$.

IX. 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], [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 sign 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 verified 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, no special nodes such as servers are in the network and every 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 sign 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), then the resulting certificate is correctly signed using the private key rk.

Both work assume that a certificate is signed by a special private key of an authority, and distribute the private key among many servers or nodes. 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], [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 investigated 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 average 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 chain sets in three classes of practical interests and two extensions of these algorithms for more general classes of chain sets.

Zheng, Omura, Uchida, and Wada presented algorithms that compute optimal dispersals for strongly-connected graphs and directed graphs in [19]. The same authors also showed the tight upper bounds in these two classes of certificate graphs.

X. 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 and two extensions of these algorithms for two more general classes of chain sets. In [20], 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 of the network). 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 chain set derivable from the certificate graph of the network 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.

Acknowledgment

The authors would like to thank Dr. Mustaque Ahamad, Dr. Xiaodong Zhang and anonymous reviewers for their valuable comments and Jean-Philippe Martin for interesting discussions on the NP-completeness proof.

References

[1] M. G. Gouda and E. E. Jung, "Stabilizaing certificate dispersal," in *2005 Symposium on Self Stabilizing Systems - LNCS 3764*. Springer-Verlag, 2005.

[2] R. L. Rivest and B. Lampson, “SDSI – A simple distributed security infrastructure,” Presented at CRYPTO ‘96 Rumpsession, 1996.

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

[4] 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.

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

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

[7] 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.

[8] Y. Elley, A. Anderson, S. Hanna, S. Mullan, R. Perlman, and S. Proctor, “Building certificate paths: Forward vs. reverse,” in *Proceedings of the 2001 Network and Distributed System Security Symposium (NDSS ‘01)*, February 2001, pp. 153–160.

[9] 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.

[10] 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.

[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, “Certificate dispersal in ad-hoc networks,” in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS ‘04)*. IEEE, 2004.

[14] ——, “Vulnerability analysis of certificate graphs,” *International Journal of Security and Networks*, vol. 1, no. 1, 2006.

[15] S. Ajmani, D. E. Clarke, C.-H. Moh, and S. Richman, “ConChord: Cooperative SDSI certificate storage and name resolution,” in *LNCS 2429 Peer-to-Peer Systems: First International Workshop, IPTPS 2002*, March 2002, pp. 141–154.

[16] M. K. Reiter and S. G. Stubblebine, “Resilient authentication using path independence,” *IEEE Transactions on Computers*, vol. 47, no. 12, pp. 1351–1362, December 1998.

[17] 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.

[18] 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.

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

[20] E. Jung, E. S. Elmallah, and M. G. Gouda, “Optimal dispersal of special certificate graphs,” in *Proceedings of IEEE Global Telecommunications Conference (Globecom ‘04)*. IEEE, 2004.