Theoretical Computer Science 622 (2016) 66–78

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Parameterized certificate dispersal and its variants

Valentin Garnero [*], Mathias Weller [*,1]

AlGCo project-team, LIRMM, Université de Montpellier 2, Montpellier, France

ARTICLE INFO

Article history:
Received 1 August 2014
Received in revised form 1 February 2016
Accepted 2 February 2016
Available online 3 February 2016
Communicated by V.Th. Paschos

Keywords:
Asymmetric cryptography
Algorithmic graph theory
Parameterized complexity
Turing kernelization

ABSTRACT

Given a directed graph G and a set R of vertex pairs, the Minimum Certificate Dispersal problem asks for an assignment of arcs to vertices ("terminals") such that, for each $(u, v) \in R$, a u-v-path can be constructed using only arcs assigned to u or v. Herein, the total number k of assignments should be minimal. The problem is well motivated in key-exchange protocols for asymmetric cryptography. We provide a first parameterized complexity analysis of this NP-hard problem and its variant Chained Minimum Certificate Dispersal, where, instead of pairs of terminals, a set of paths ("chains") that should be constructed, is prescribed.

Although polynomial-time solvable for constant values of k, the former variant seems much harder, surfacing in the proof that it is W[1]-hard with respect to k while Chained Minimum Certificate Dispersal yields a polynomial-size problem kernel. We even show fixed-parameter tractability of the latter with respect to the stronger parameter "number t of terminals". In particular, while there is no polynomial-size kernel with respect to t, the problem admits a polynomial-size Turing kernel. Towards answering the question whether Minimum Certificate Dispersal can be solved in polynomial time when t is constant, we show polynomial-time solvability for at most two requests (comprising all instances with $t \leq 2$) using an algorithm for the Strong Distance problem, which asks for a minimum-size subdigraph in which two given vertices are strongly connected. Finally, we emphasize the hardness of Minimum Certificate Dispersal by proving it NP-hard for very restricted sets of instances, thereby excluding many parameters and combinations from consideration for efficient multivariate algorithms.

1. Introduction

Modern-day security heavily relies on asymmetric cryptography: Online banking, secure payment, VPN services, and email encryption are just a few examples. At the heart of asymmetric cryptography is the idea that each participant has a pair of keys: a "public key" that everyone can know and a "private key" that is to be kept secret. When some participant (Alice) wants to send a secret message to another (Bob), then this message is encrypted using the public key of Bob. A thusly encrypted message can only be decrypted with the private key of Bob. Note that this technique requires Alice to know the public key of Bob in advance. Often though, a participant cannot store the public keys of all communication partners locally, so Alice asks Bob for his public key before encrypting the message. This opens a possible avenue of attack for a malicious player (Chuck): pretend to be Bob and send Alice a false public key, preferably Chuck's own so he can decrypt the message

[*] Corresponding authors.
E-mail addresses: garnero@lirmm.fr (V. Garnero), weller@lirmm.fr (M. Weller).
[1] Supported by Centre National de la Recherche Scientifique (CNRS), project KERNEL.

http://dx.doi.org/10.1016/j.tcs.2016.02.001
0304-3975/© 2016 Elsevier B.V. All rights reserved.

that is subsequently sent by Alice. Chuck can then even relay the message (or a modified version of it) to Bob. This is called "man-in-the-middle attack". In order to prevent such an attack, Bob can ask someone (Dave) to "sign" his public key. The public key of Bob signed by Dave is called a *certificate* of Bob from Dave. Everyone who has the public key of Dave and the certificate of Bob from Dave can reconstruct the public key of Bob. Now, however, Chuck could impersonate Dave the same way as above, but, given that there are few "certificate authorities" like Dave, Alice can be assumed to know their public keys before attempting to communicate with Bob.

Communication between peers that, for whatever reason (financial, political, ideological, ...) cannot or won't submit to central certificate authorities is not uncommon when exchanging encrypted email, operating non-corporate VPN, or remotely authenticating oneself [15,34]. Here, users issue certificates of other users. Then, certificates can be chained in order to reconstruct the desired public key, that is, having a certificate of Bob from Dave and a certificate of Dave from Alice allows anyone who knows the public key of Alice to reconstruct the public key of Bob. This is called a "certificate chain". We consider a directed graph in which vertices are participants and arcs are certificates issued between participants. Since issuing a certificate of someone symbolizes trust in his genuineness, this directed graph is sometimes referred to as "network of trust". In our example above, Alice trusts Dave to be genuine and Dave trusts Bob to be genuine. In order to forge a certificate chain, the attacker Chuck has to know the private key of anyone whose genuineness can be verified by a certificate chain starting at Alice. Assuming that trusted participants do not let their private keys get compromised, Alice can accept certificates from any source (even Chuck) without risk of compromising the encrypted message to Bob. Therefore, it is reasonable that Alice and her communication partner (be it Bob or Chuck) pool their resources in order to construct a certificate chain.

In our toy example, the network of trust contains the arcs/certificates Alice $\to$ Dave and Dave $\to$ Bob and, to allow Alice and Bob to communicate securely, Alice could store the arc/certificate Alice $\to$ Dave and Bob could store Dave $\to$ Bob. Thus, pooling their resources, Alice can reconstruct the public key of Bob by first decrypting the public key of Dave using her own public key and the certificate Alice $\to$ Dave and then, using this public key, decrypt the public key of Bob from the certificate Dave $\to$ Bob which was provided by Bob. For further reading and advanced topics, refer to Ferguson and Schneier [15].

In the following, we consider the problem of storing certificates in vertices of a given certificate graph such that it is possible to construct certificate chains between a given set of vertex pairs without exceeding a given limit on the total storage. Herein, each certificate chain can be constructed from all certificates stored in any of its endpoints.

Minimum Certificate Dispersal (MCD)
Input: A digraph $G = (V, A)$, a request set $R \subseteq V \times V$ and an integer k.
Question: Is there some[2] $D : V \to 2^A$ such that $\sum_{v \in V} |D(v)| \leq k$ and, for each $(u, v) \in R$, there is a u-v-path in the graph $(V, D(u) \cup D(v))$?

A slight variant of MCD provides the certificate chain for each communication request in the input.

Chained Minimum Certificate Dispersal (cMCD)
Input: A set of vertices V, a set of chains $\mathfrak{C}$, and an integer k.
Question: Is there an assignment $D : V \to 2^{V \times V}$ such that $\sum_{v \in V} |D(v)| \leq k$ and, for each chain $C \in \mathfrak{C}$ from u to v, it holds that $C \subseteq D(u) \cup D(v)$?

Both problems can be considered in variants where, instead of the sum of $|D(v)|$ over all $v \in V$, the *maximum* of $|D(v)|$ over all $v \in V$ is minimized. We are convinced that some practical application scenarios are better modeled by these MinMax versions since it is more likely that the resources of a participant of a network of trust are limited than that the overall resources are limited.

Previous work cMCD was first considered by Jung et al. [23], who proved it NP-complete by reducing the well-known Vertex Cover problem to it. In the instances their reduction constructs, the maximum length of any chain is three, excluding this parameter from consideration for parameterized algorithms. Jung et al. [23] go on to show that cMCD is polynomial-time solvable if any of the following holds: (a) the maximum chain-length is two, (b) the "request graph" induced by the edge set $\{\{u, v\} \mid (u, \ldots, v) \in \mathfrak{C}\}$ is bipartite, or (c) the chains can be embedded in a directed graph of circumference two. Finally, Jung et al. [23] show that cMCD is fixed-parameter tractable with respect to the number of chains of length more than two. In fact, their algorithm takes at most $2^t \cdot \text{poly}(\sum_{C \in \mathfrak{C}} |C|)$ time, where t denotes the number of endpoints of chains ("terminals") in the input. Note that t can be upper-bounded by $2|\mathfrak{C}|$ but, since chains are allowed to be "redundant", that is, one pair of terminals can have an arbitrary number of chains between them, the number of chains cannot be bounded in the number of terminals. If we disallow redundant chains, then $|\mathfrak{C}| \leq \binom{t}{2}$. However, since Jung et al. [23] only count chains whose length exceeds two, their parameter is incomparable to t.

MCD was introduced and proved NP-complete by Zheng et al. [32,33]. Since then, it has mostly been considered under the paradigm of approximation. In particular, Izumi et al. [19] showed a lower and upper bound of $\Theta(\log n)$ for approximation factors, which can be improved to 2 for special cases of requests. If G is an undirected graph, the problem is APX-hard, even if R forms a star, but can be solved in $n^{O(\Delta)}$ time if R forms a tree of maximum degree Δ and in polynomial time

[2] We denote the power set of any set X by 2^X.