

DISSERTATION

NETWORKS AND TRUST: SYSTEMS FOR UNDERSTANDING AND SUPPORTING INTERNET SECURITY

Submitted by

Bryan Charles Boots

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2022

Doctoral Committee:

Advisor: Steven J. Simske

Ramadan Abdunabi  
Anura Jayasumana  
Leo Vijayasarathy

Copyright by Bryan Charles Boots, 2022

All Rights Reserved

## ABSTRACT

### NETWORKS AND TRUST: SYSTEMS FOR UNDERSTANDING AND SUPPORTING INTERNET SECURITY

This dissertation takes a systems-level view of the multitude of existing trust management systems to make sense of when, where and how (or, in some cases, if) each is best utilized. Trust is a belief by one person that by transacting with another person (or organization) within a specific context, a positive outcome will result. Trust serves as a heuristic that enables us to simplify the dozens decisions we make each day about whom we will transact with. In today's hyperconnected world, in which for many people a bulk of their daily transactions related to business, entertainment, news, and even critical services like healthcare take place online, we tend to rely even more on heuristics like trust to help us simplify complex decisions. Thus, trust plays a critical role in online transactions.

For this reason, over the past several decades researchers have developed a plethora of trust metrics and trust management systems for use in online systems. These systems have been most frequently applied to improve recommender systems and reputation systems. They have been designed for and applied to varied online systems including peer-to-peer (P2P) filesharing networks, e-commerce platforms, online social networks, messaging and communication networks, sensor networks, distributed computing networks, and others. However, comparatively little research has examined the effects on individuals, organizations or society of the presence or absence of trust in online

sociotechnical systems. Using these existing trust metrics and trust management systems, we design a set of experiments to benchmark the performance of these existing systems, which rely heavily on network analysis methods. Drawing on the experiments' results, we propose a heuristic decision-making framework for selecting a trust management system for use in online systems.

In this dissertation we also investigate several related but distinct aspects of trust in online sociotechnical systems. Using network/graph analysis methods, we examine how trust (or lack of trust) affects the performance of online networks in terms of security and quality of service. We explore the structure and behavior of online networks including Twitter, GitHub, and Reddit through the lens of trust. We find that higher levels of trust within a network are associated with more spread of misinformation (a form of cybersecurity threat, according to the US CISA) on Twitter. We also find that higher levels of trust in open source developer networks on GitHub are associated with more frequent incidences of cybersecurity vulnerabilities.

Using our experimental and empirical findings, we apply the Systems Engineering Process to design and prototype a trust management tool for Reddit, which we dub Coni the Trust Moderating Bot. Coni is, to the best of our knowledge, the first trust management tool designed specifically for use on the Reddit platform. We develop and present a blueprint for constructing a Reddit trust tool which not only measures trust levels, but can use these trust levels to take actions on Reddit to improve the quality of submissions within the community (a subreddit).

## ACKNOWLEDGEMENTS

I want to recognize and thank my incredible advisor, Professor Steve Simske. Your outstanding recommendations, feedback, and ideas were a central part of helping me to complete this dissertation. Your intellect, breadth and depth of expertise in distinct areas are singular – you are a true renaissance man! In addition to the content of this dissertation, you helped me learn a great deal about our field, and you have been a true mentor to me. I could not have completed this journey without your help and support. I want to recognize and thank my distinguished doctoral committee, who have provided guidance, recommendations, oversight, and encouragement to me – thank you, Professor Jayasumana, Professor Vijayasathy, and Professor Abdunabi. I am grateful for the opportunity to have learned from each of your experiences and expertise. Thank you to the innumerable other mentors and experts who helped me along different points in my doctoral journey with their expertise, ideas, advice and support. While many contributed to my journey and I can't thank them all here, I would be remiss if I didn't specifically thank Jeff Hornsby, Felipe Tapia, Sergio Canavati, Henrique Houayek, Lyle Paczkowski, Kevin Rooney, Nathan Smith, Brian Anderson, Jim Williams, Scott Savage, Steve Wildman, David Reed, Dan Larremore, Brian Keegan, and David Eargle. Thank you to all of the Reddit users who shared their time and experience with me in our interviews for Chapter VII. Thank you to the amazing communities on Stack Overflow, GitHub, and other helpful Internet strangers. Finally, thank you to the smart and talented contributors to all of the open source Python libraries I make use of in this dissertation.

## DEDICATION

I could not have completed this dissertation without the unwavering and unconditional support and love of my family. This dissertation is dedicated to Katherine, Betty, Charles, Ashley, Justin, Stephanie, Alicia, Holly, Leah, Alice, and Lucas. Thank you all.

## TABLE OF CONTENTS

ABSTRACT.....	ii
ACKNOWLEDGEMENTS .....	iv
DEDICATION.....	v
LIST OF TABLES .....	x
LIST OF FIGURES .....	xii
LIST OF EQUATIONS .....	xviii
I. Introduction .....	1
Contributions of this dissertation.....	3
Dissertation organization.....	6
II. Background and Overview .....	8
Problem statement.....	12
Research questions investigated in this dissertation.....	13
Scope.....	14
III. Systematic analysis of trust management systems for online applications.....	16
Types of and properties of trust.....	19
Quantitative scale of trust .....	20
Transitive trust (property).....	21
Direct and indirect trust.....	24
Distrust.....	25
Global trust and specific (or local) trust .....	26
Swift trust.....	27
Hard trust and soft trust.....	27
Realms of trust.....	28
Key objectives of trust models.....	29
Application context.....	31
Systems or algorithms for measuring and managing trust online.....	32
Approaches to computing trust.....	35

Graph-based approaches .....	38
Content-based and behavior-based approaches .....	64
Probabilistic approaches .....	67
Fuzzy Logic approaches .....	73
Game Theoretic approaches .....	75
Hybrid approaches (using two or more of the previously-identified approaches together) .....	78
Other perspectives of trust online .....	89
Social and psychological perspectives on trust .....	95
Open challenges in the design and deployment of trust metrics and trust management systems .....	97
IV. Benchmarking performance of trust metrics, and proposed framework for matching a trust metric with an application .....	100
Introduction.....	100
Research questions.....	101
Related work.....	102
Benchmarking performance of trust metrics.....	103
Design of benchmarking experiments .....	104
Analysis .....	115
Results and Discussion .....	123
Framework for selecting a trust model.....	156
Limitations .....	163
Future Work.....	164
V. Characterizing the nature of trust and misinformation on Twitter .....	166
Introduction.....	167
Background .....	168
Related Work.....	171
Research Questions and Datasets.....	172
Methods.....	176
The trust overlay network .....	178
Defining metrics used: hubs, authority, PageRank, eigenvector centrality.....	181



Generating random networks.....	185
Results and Discussion.....	185
Key findings .....	185
Summary statistics.....	188
Visualizations.....	191
Who is in both networks? .....	196
Components.....	204
Degree distribution .....	205
Reciprocity .....	217
Mean shortest paths.....	217
Centrality measures.....	219
Small worldness .....	227
Conclusions and Future Work.....	227
VI. The relationship between trust and security in open source development .....	231
Introduction.....	231
Background .....	232
Related Work.....	234
Research Questions.....	237
Methods and Data .....	238
Dataset: rationale, data collection and dataset construction.....	239
Analysis measures.....	252
Results and Discussion.....	255
Summary statistics.....	257
Degree assortativity.....	258
Density .....	258
Reciprocity .....	259
Geodesic paths.....	259
Network visualizations.....	260
Node degree distribution .....	267
Direct trust and transitive trust analysis results.....	271

Community detection .....	293
Conclusions and Future Work.....	297
Limitations .....	299
Future work.....	300
VII. Applying the Systems Engineering Process to designing a trust management tool for Reddit .....	303
Introduction.....	303
Overview of Reddit.....	305
Related Work .....	310
Methods.....	311
Overview of the Systems Engineering Process.....	312
Application of the Systems Engineering Process to Our Design Challenge .....	315
Discussion of Results.....	346
Limitations.....	347
Future Work.....	347
VIII. Discussion, Limitations, and Future Work.....	350
Discussion of this dissertation’s contributions.....	350
Limitations.....	352
Future Work.....	354
References.....	358
Chapter I .....	358
Chapter II .....	358
Chapter III .....	358
Chapter IV .....	363
Chapter V .....	364
Chapter VI.....	367
Chapter VII.....	369

## LIST OF TABLES

<b>Table 1:</b> Research questions investigated in this dissertation .....	14
<b>Table 2:</b> Selected trust management systems and trust metrics since 1999 .....	33
<b>Table 3:</b> Features review of selected trust metrics and trust management systems, in descending order of influence (as measured by number of citations).....	35
<b>Table 4:</b> Experiment set combinations. Features highlighted in green signify the feature that is variable for that experiment set. ....	106
<b>Table 5:</b> Structural characteristics of randomly small world and $G_{n,p}$ networks used in Experiment Set 1.1 and 1.2.....	122
<b>Table 6:</b> Structural characteristics for randomly-generated small networks used in Experiment Set 2. ....	122
<b>Table 7:</b> Structural characteristics for randomly-generated small networks used in Experiment Set 3. ....	123
<b>Table 8:</b> MAE, code execution time, and Balanced Performance for different trust measures in Experiment Set 1.2. ....	132
<b>Table 9:</b> MAE, code execution time, and combined performance (MAE x execution time) for Experiment Set 2. For all experiments, $n = 400$ and $K = 150$ , while $p$ varies .....	138
<b>Table 10:</b> MAE, code execution time, and combined performance (MAE x execution time) for Experiment Set 3. For all experiments, $n = 400$ and $p = 0.02$ , while $K$ varies in increments of 50, from 50 to 250.....	142
<b>Table 11:</b> Correlations between graph structural characteristics and performance of trust metrics.....	147
<b>Table 12:</b> Relationship between graph structural characteristics in small world networks and MAE for different categories of trust model.....	162
<b>Table 13:</b> Relationship between graph structural characteristics and code execution times for different categories of trust model.....	162
<b>Table 14:</b> Number of nodes and directed edges for empirical networks analyzed .....	178
<b>Table 15:</b> Trust overlay network summary .....	180
<b>Table 16:</b> Summary Statistics for Directed Network Models.....	188
<b>Table 17:</b> Trust network with brokers .....	198
<b>Table 18:</b> Correlations of network structural measures in conspiracies and non-conspiracies network .....	211
<b>Table 19:</b> Reciprocity in the 5G COVID conspiracies graph and the non-conspiracies graph .....	217
<b>Table 20:</b> Mean shortest path lengths for empirical and random graphs of the 5G COVID conspiracies network and the non-conspiracies network, respectively .....	218

<b>Table 21:</b> Summary of graphs generated from the OpenSSL and LE projects for analysis in this chapter .....	252
<b>Table 22:</b> Summary Statistics for OpenSSL and LE trust networks.....	257
<b>Table 23:</b> Key structural measures for empirical and random versions of OpenSSL and LE trust networks.....	257
<b>Table 24:</b> Mean shortest path lengths for the empirical and random versions of the OpenSSL trust network and the LE trust network .....	260
<b>Table 25:</b> Correlations of trust measures in OpenSSL and LE empirical networks .....	293
<b>Table 26:</b> Greedy modularity maximization (GMM) community detection statistics for OpenSSL and LE trust networks .....	297
<b>Table 27:</b> Operational requirements for our trust tool .....	320
<b>Table 28:</b> Candidate PRAW features for use in Reddit trust system.....	324
<b>Table 29:</b> Functional Requirements and Performance Requirements for Coni the Trust Moderating Bot.....	326
<b>Table 30:</b> Selected features for computing trust in our system.....	330
<b>Table 31:</b> Sample computation of raw trust values for each object in the example.....	340
<b>Table 32:</b> Trust adjacency matrix for sample trust calculation .....	341

## LIST OF FIGURES

<b>Figure 1:</b> Visual overview of the universe of trust; green objects indicate those that are within the scope of this work .....	15
<b>Figure 2:</b> Examples of trust in different contexts.....	19
<b>Figure 3:</b> Illustration of transitive trust concept in a small trust network. The solid black lines represent a direct trust relationship, while the dashed lines indicate transitive trust. 21	
<b>Figure 4:</b> Illustration of transitive trust concept in a small trust network. The values placed on the edges signify the trust that exists between the corresponding nodes.....	24
<b>Figure 5:</b> Categories of features considered in design of benchmarking experiments.....	105
<b>Figure 6:</b> Visualizations of small world graphs of different sizes used for Experiment Set 1.1; panel (1) with $n = 400$ , panel (2) with $n = 800$ , panel (3) with $n = 1600$ , panel (4) with $n = 3200$ , and panel (5) with $n = 6400$ . $p=0.02$ and $K=150$ for all. Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1. ....	117
<b>Figure 7:</b> Visualizations of $G_{n,p}$ random graphs of different sizes used for Experiment Set 1.2; panel (1) with $n = 400$ , panel (2) with $n = 800$ , panel (3) with $n = 1600$ , panel (4) with $n = 3200$ , and panel (5) with $n = 6400$ . $p=0.02$ for all. Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1. ....	118
<b>Figure 8:</b> Visualization of random graphs used in Experiment Set 2. In Experiment Set 2, we hold constant $n = 400$ and $K = 150$ while varying $p$ from 0.02 to (Panel 1), to 0.05 (Panel 2), 0.50 (Panel 3), 0.95 (Panel 4), and 0.98 (Panel 5). Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1. ....	119
<b>Figure 9:</b> Visualizations of random small world graphs used in Experiment Set 3. In Experiment Set 3, we hold constant $n = 400$ and $p = 0.02$ while varying $k$ from 50 to (Panel 1), to 100 (Panel 2), 150 (Panel 3), 200 (Panel 4), and 250 (Panel 5). Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1.....	120
<b>Figure 10:</b> Performance of trust metrics as measured by Mean Absolute Error (MAE) for varying sizes of graphs in Experiment Set 1.2.....	128
<b>Figure 11:</b> Code execution time (in seconds) for different trust measures in Experiment Set 1.1, for varying sizes of Watts-Strogatz small-world networks. Semilog scale. ....	129
<b>Figure 12:</b> Ratio of execution time to MAE of trust metrics for varying sizes of graphs in Experiment Set 1.1. Semilog scale. ....	130
<b>Figure 13:</b> Performance of trust metrics as measured by Mean Absolute Error (MAE) for varying sizes of graphs in Experiment Set 1.2.....	133
<b>Figure 14:</b> Code execution time (in seconds) for different trust measures in Experiment Set 1.2, for varying sizes of Watts-Strogatz small-world networks. Semilog scale. ....	134

<b>Figure 15:</b> Ratio of execution time to MAE of trust metrics for varying sizes of graphs in Experiment Set 1.2. Semilog scale. ....	135
<b>Figure 16:</b> MAE of each trust metric when applied to the random graphs of Experiment Set 2. ....	139
<b>Figure 17:</b> Code execution time (in seconds) for different trust measures in Experiment Set 2. Semilog scale. ....	140
<b>Figure 18:</b> Balanced performance (MAE x code execution time) for different trust measures in Experiment Set 2. ....	141
<b>Figure 19:</b> MAE of each trust metric when applied to the random graphs of Experiment Set 3. ....	144
<b>Figure 20:</b> Code execution time (in seconds) for different trust measures in Experiment Set 3. ....	145
<b>Figure 21:</b> Correlation between graph structural characteristics and eigenvector centrality's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).....	149
<b>Figure 22:</b> Correlation between graph structural characteristics and PageRank's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).....	150
<b>Figure 23:</b> Correlation between graph structural characteristics and GTT's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds). ....	151
<b>Figure 24:</b> Correlation between graph structural characteristics and EigenTrust's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).....	152
<b>Figure 25:</b> Correlation between graph structural characteristics and TrustRank's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).....	153
<b>Figure 26:</b> Correlation between MAE and graph structural characteristics for trust metrics. ....	158
<b>Figure 27:</b> Correlation between code execution times and graph structural characteristics for trust metrics. ....	159
<b>Figure 28:</b> Correlation between Balanced Performance and graph structural characteristics for trust metrics. ....	160
<b>Figure 29:</b> Conceptual illustration of the construction of a tweet subgraph. Authors' own work.....	175
<b>Figure 30:</b> Conceptual illustration of how tweet subgraphs are interconnected, forming the overall tweet graph.....	176
<b>Figure 31:</b> Graphical illustration of reciprocity among pairs of nodes.....	184

**Figure 32:** Weakly connected components of empirical networks analyzed in this chapter, visualized using circular layout; panel (A) depicts the empirical non-conspiracies network, and panel (B) depicts the empirical 5G COVID conspiracies network..... 193

**Figure 33:** 5G COVID conspiracies network, visualized using NetworkX and matplotlib in Python language, with spring layout. In this figure, nodes' sizes are adjusted based on their degree, with higher degree being illustrated by a larger node size..... 194

**Figure 34:** 5G COVID conspiracies networks weakly connected components, visualized using NetworkX and matplotlib in Python language, with spring layout. Panel (A) depicts the empirical network with nodes in gold, and panel (B) depicts the random version of the same network in olive. .... 195

**Figure 35:** Possible scenarios for explaining presence of nodes in both the conspiracies and the non-conspiracies networks. In panel (A), C is a member of the conspiracies network, and visits the non-conspiracy network to attempt to convert non-conspiracy nodes. In panel (B), C was originally a non-conspiracy member, but is converted and becomes a member of the conspiracy network..... 197

**Figure 36:** Ranking by in-degree of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale..... 200

**Figure 37:** Ranking by out-degree of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale..... 201

**Figure 38:** Ranking by eigenvector centrality of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale..... 203

**Figure 39:** Illustration of a bridge in an example network ..... 204

**Figure 40:** Top 100 components by size for the empirical networks; the non-conspiracies network is depicted in green, and the 5G COVID conspiracies network is depicted in gold ..... 208

**Figure 41:** Normalized plots of out-degree v. (clockwise from top left) hub, authority, eigenvector centrality, and PageRank for the 5G COVID conspiracies network ..... 209

**Figure 42:** Normalized plots of out-degree v. (clockwise from top left) hub, authority, eigenvector centrality, and PageRank for the non-conspiracies network ..... 210

**Figure 43:** In-degree v. out-degree for the non-conspiracies network; panel (A) focuses on degrees below 100 where data is richer, and (B) illustrates the entire degree distribution ..... 212

**Figure 44:** In-degree v. out-degree for the 5G COVID conspiracies network; panel (A) focuses on degrees below 100 where data is richer, and (B) illustrates the entire degree distribution..... 212

**Figure 45:** Out-degree ranking plot for 5G COVID conspiracies network (left, in gold) and non-conspiracies network (right, in green) ..... 213

<b>Figure 46:</b> Degree distributions plotted with log-log scale; on the left (in gold) the empirical 5G conspiracies network is depicted, and on the right (in olive) the random 5G conspiracies network is depicted .....	213
<b>Figure 47:</b> Degree distributions plotted with log-log scale; on the left (in gold) the 5G conspiracies network is depicted, and on the right (in green) the non-conspiracies network is depicted .....	214
<b>Figure 48:</b> Ranking of indegree of top 5000 nodes for empirical networks.....	215
<b>Figure 49:</b> Ranking of outdegree of top 5000 nodes for empirical networks.....	216
<b>Figure 50:</b> Ranking of PageRank for empirical and random versions of 5G COVID conspiracies network .....	221
<b>Figure 51:</b> Rankings of PageRank for empirical and random versions of non conspiracies network. Semilog scale. ....	222
<b>Figure 52:</b> Rankings of PageRank for empirical 5G COVID conspiracies and non conspiracies networks. Semilog scale.....	223
<b>Figure 53:</b> Rankings of eigenvector centralities in empirical and random versions of the 5G COVID conspiracies network. Semilog scale. ....	224
<b>Figure 54:</b> Rankings of eigenvector centralities in empirical and random versions of the non conspiracies network. Semilog scale. ....	225
<b>Figure 55:</b> Rankings of eigenvector centralities for empirical 5G COVID conspiracies and non-conspiracies networks. Semilog scale.....	226
<b>Figure 56:</b> An example of a pull request on GitHub. This pull request is the first public pull request for the OpenSSL project, from 2013. ....	240
<b>Figure 57:</b> Sample linked list constructed from a thread of comments in one pull request .....	243
<b>Figure 58:</b> “For” loop written to transform linked lists of pull request interactions into edge lists useful for network analysis.....	245
<b>Figure 59:</b> Sample edge list, transformed from the linked list in Figure 57 .....	246
<b>Figure 60:</b> Conceptual illustration of the construction of directed versions of the graphs	247
<b>Figure 61:</b> Example graph with self-loops. Generated in the Python language using the NetworkX and matplotlib libraries .....	248
<b>Figure 62:</b> Conceptual illustration of the construction of directed versions of the graphs	249
<b>Figure 63:</b> Illustration of representation of reciprocity and weighted edges in our dataset construction .....	250
<b>Figure 64:</b> Illustration of bipartite network representation; round nodes are developers, square nodes are documents.....	251
<b>Figure 65:</b> Empirical version of OpenSSL trust network, generated using spring layout in NetworkX and Matplotlib in the Python language.....	262
<b>Figure 66:</b> Random version of OpenSSL trust network, generated using spring layout in NetworkX and Matplotlib in the Python language.....	263



<b>Figure 67:</b> Empirical version of LE trust network, generated using spring layout in NetworkX and Matplotlib in the Python language .....	264
<b>Figure 68:</b> Random version of LE trust network, generated using spring layout in NetworkX and Matplotlib in the Python language .....	265
<b>Figure 69:</b> Empirical OpenSSL trust network core .....	266
<b>Figure 70:</b> Empirical LE trust network core .....	266
<b>Figure 71:</b> Degree distribution in OpenSSL trust network. Log-log scale.....	268
<b>Figure 72:</b> Degree distribution in LE trust network. Log-log scale.....	269
<b>Figure 73:</b> Degree rank plot (left) and degree histogram (right) of OpenSSL trust network. ....	270
<b>Figure 74:</b> Degree rank plot (left) and degree histogram (right) of OpenSSL trust network. ....	270
<b>Figure 75:</b> Direct trust measures for OpenSSL empirical trust network. Normalized degree $v$ . (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank. ....	273
<b>Figure 76:</b> Direct trust measures for LE empirical trust network. Normalized degree $v$ . (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank. ....	274
<b>Figure 77:</b> Transitive trust measures for OpenSSL empirical trust network. Normalized transitive trust $v$ . (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank. ....	275
<b>Figure 78:</b> Transitive trust measures for LE empirical trust network. Normalized transitive trust $v$ . (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank. ....	276
<b>Figure 79:</b> Comparing eigenvector centrality rankings for OpenSSL and LE empirical networks.....	279
<b>Figure 80:</b> Comparing PageRank rankings for OpenSSL and Let's Encrypt empirical trust networks.....	280
<b>Figure 81:</b> Ranking of eigenvector centralities for OpenSSL trust networks (empirical and random).....	281
<b>Figure 82:</b> Ranking of eigenvector centralities for Let's Encrypt trust networks (empirical and random) .....	282
<b>Figure 83:</b> Ranking of PageRank for OpenSSL empirical and random networks. Semilog scale.....	283
<b>Figure 84:</b> Ranking of PageRank for Let's Encrypt empirical and random networks. Semilog scale.....	284
<b>Figure 85:</b> Transitive trust rankings for empirical OpenSSL and LE networks. Semilog scale. ....	286
<b>Figure 86:</b> Transitive trust rankings for empirical and random OpenSSL network. Semilog scale .....	287

<b>Figure 87:</b> Transitive trust rankings for empirical and random LE network. Semilog scale. .....	288
<b>Figure 88:</b> Normalized degree v. normalized transitive trust for OpenSSL empirical network .....	290
<b>Figure 89:</b> Normalized degree v. normalized transitive trust for LE empirical network.....	291
<b>Figure 90:</b> Greedy modularity maximization community detection in the OpenSSL empirical trust network. Green colored edges signify edges across communities, and black edges signify edges within communities.....	295
<b>Figure 91:</b> Greedy modularity maximization community detection in the LE empirical trust network. Gold colored edges signify edges across communities, and black edges signify edges within communities.....	296
<b>Figure 92:</b> Top 10 subreddits by number of subscribers, Sept. 2022. Source: <a href="http://redditlist.com/all">http://redditlist.com/all</a> .....	306
<b>Figure 93:</b> Example of a post and comments from Reddit.....	308
<b>Figure 94:</b> Examples of awards given to a post on Reddit .....	309
<b>Figure 95:</b> The Systems Engineering Process (or Systems Engineering V). Author’s own work, adapted from [7.7] .....	312
<b>Figure 96:</b> System Architecture of Coni the Trust Moderating Bot.....	333
<b>Figure 97:</b> Functional Flow Block Diagram for Coni the Trust Moderating Bot .....	334
<b>Figure 98:</b> Pseudocode for the data retrieval module of Coni the Trust Moderating Bot..	336
<b>Figure 99:</b> Conceptual structure of a subreddit graph .....	338
<b>Figure 100:</b> Bipartite subreddit graph.....	338
<b>Figure 101:</b> Network projection of user graph within a subreddit .....	339
<b>Figure 102:</b> Conceptual trust overlay network .....	340
<b>Figure 103:</b> Resulting trust overlay network for this example .....	341
<b>Figure 104:</b> Trust computation module pseudocode .....	342

## LIST OF EQUATIONS

<b>Equation 1:</b> Eigenvector centrality of a given node, $i$ .....	108
<b>Equation 2:</b> Mean Absolute Error (MAE), from [4.21] .....	114
<b>Equation 3:</b> Eigenvector centrality of a given node, $i$ .....	182
<b>Equation 4:</b> Geodesic distance/shortest distance.....	183
<b>Equation 5:</b> Equation for calculating reciprocity in a directed network.....	184
<b>Equation 6:</b> Calculating assortativity in a network. Adapted from [5.21] Mixing patterns in networks, Newman. ....	184
<b>Equation 7:</b> Equation for calculating network density.....	190
<b>Equation 8,</b> from [6.18], where $Q$ is modularity, $m$ is the number of edges in the network being analyzed, $A$ is an element of the adjacency matrix representing the network in question, $k$ is the degree of a node $v$ or $w$ , and $c$ are the communities detected in the network .....	253
<b>Equation 9,</b> from [6.18] .....	253

## I. Introduction

To function effectively in a modern economy and society – one which is underpinned and driven by information and communication technologies – we have to make some simplifying assumptions about the people, organizations, machines, and other agents we engage with. One way we do so is through *trust*. Do I trust this person to give me a good recommendation on Yelp? Do I trust this person’s opinions about news or politics on Twitter? Do I trust this business to properly safeguard my payment information? Do I trust this developer’s pull request to an open source software project?

Trust, while an imperfect, rough, and subjective measure, nonetheless serves us quite well to be able to engage with people and companies online or in-person with acceptably low transaction costs [1.1]. Trust is subjective, and is different for every person. It also changes depending on the context, and it changes over time as agents have new interactions (or stop interacting). Trust is like a shortcut, which helps us to quickly make decisions using heuristics [1.2]. This doesn’t mean that trust does or should replace the deep, critical analysis that’s called for in complex decisions, but in a world where we have to make hundreds of decisions each day (most of them minor and mundane) trust enables us to dedicate our time and attention to the areas of our lives that require deeper thinking. In subsequent chapters we provide a deeper discussion of the definition and boundaries of trust as used in this dissertation, but to orient the reader we provide the following

definition of trust for our purposes, from [1.3]: *"Alice trusts Bob if she commits to an action based on the belief that Bob's future actions will lead to a good outcome."*

In 2022, there were approximately 13 billion devices connected to the Internet in – that's more than one device for every single living person on the planet [1.4]. And, according to the same estimate, by 2030 that figure will more than double to more than 29 billion connected devices. According to [1.5], the average American spent more than 1300 hours on social media of all types in 2021 – a mean of more than 3.5 hours per day, every day of the year. According to [1.6] more than half of Americans shop online at least once per year. And, according to [1.7], GitHub – the wildly popular online software code repository – counted more than 73 million users (software developers) worldwide in 2021.

The preceding are all illustrations of the reality of being a participant and a citizen in today's hyperconnected world. As this connectivity not only grows but accelerates, it becomes increasingly difficult for individual users, companies, and other organizations to accurately discern who they should deal with, how much to trust the people or organizations they do deal with, and who they should lock out of their sphere.

Since at least the 1990s, researchers and practitioners in computer science and adjacent fields have sought to develop computational methods to improve our understanding – even if in only a coarse-grained manner – of whom we should trust online, whom we should distrust online, how much, and in what contexts. Dozens of measures – varyingly referred to as trust metrics, trust systems, trust management systems, trust

values, and other related terms – have been proposed in an attempt to give a quantified measure of trust in online systems.

Given the nature of online connected systems, it can be useful to model these systems as networks or graphs, allowing us to extract useful insights based on these graphs' structure and dynamics. How can we utilize network-centric information about online connected systems to better understand the nature of trust in these systems? And, if we can better understand trust in these systems – what encourages it, what discourages it, what its effects are (or what the effects are when it is absent) – can we use these understandings to design higher-performing, safer online networks? This dissertation explores these questions, using a variety of methods including empirical analysis of real online networks, analytical tools from graph theory and network science, experimental simulations, and formal design processes.

### **Contributions of this dissertation**

While it would be satisfying to design or discover a universal trust management system that can be effectively utilized in any network (at least, any online network), all of the evidence thus far from other scholars and practitioners indicates that each application, platform or context requires its own unique version of a trust management system.

This stems from the fact that, among other reasons, agents have different expectations for trust or distrust depending on the system they interact with; the types of data available to serve as inputs into a trust management system can vary dramatically depending on the system in question; the actions being carried out are generally different

on each platform; and/or the definition of success for quality of service, security, or performance (which can all be affected by trust) vary for each system with some systems prioritizing speed, others prioritizing engagement, and so on.

This dissertation takes a systems-level view of the multitude of existing trust management systems in an attempt to make sense of when, where and how (or, in some cases, if) each is best utilized.

With this in mind, this dissertation makes the following contributions to the domain of online trust metrics and online trust management systems. With respect to decision making related to trust management system selection, the following contributions are highlighted:

1. Systems-level analysis of different trust models (*Ch. III*),
2. Quantitative benchmarking and detailed analysis of trust models' performance (*Ch. IV*),
3. To the best of our knowledge, the most extensive (and one of only a few) examination of how different graph structural characteristics influence the performance of trust models for online systems (*Ch. IV*), and
4. A proposed heuristic framework for selecting which trust model to use, given specific network characteristics (*Ch. IV*).

With respect to characterizing empirical trust networks to provide better understanding of them and the effects of trust (or lack thereof) in real networks, we note the following contributions:

5. An extensive investigation of how online trust affects and is affected by (cyber)security. This is collectively supported by the following contributions:
  - a. Discovery of small worldness in an online misinformation trust network (*Ch. V*),
  - b. Characterization of the topology of an online misinformation trust network on Twitter (*Ch. V*),
  - c. Exploration of the relationship between trust and the spread of misinformation in an online social network, Twitter (*Ch. V*),
  - d. Construction of a novel dataset for exploring the relationship between trust and cybersecurity vulnerabilities in GitHub projects (*Ch. VI*),
  - e. Discovery of small worldness in online developer networks on GitHub (*Ch. VI*),
  - f. Exploration of the relationship between trust and cybersecurity vulnerabilities in an online developer network on GitHub (*Ch. VI*),
6. Construction of a novel dataset of GitHub data (*Ch. VI*), which can be used by other researchers investigating different questions related to trust and networks, or researchers pursuing general (not related to trust) networks questions,
7. To the best of our knowledge, the first Systems Engineering Process-driven design of a new trust model (*Ch. VII*), and
8. To the best of our knowledge, design of the first open source prototype trust model specifically for the Reddit online social network, which we dub Coni the Trust Moderating Bot (*Ch. VII*).



## **Dissertation organization**

The remainder of the dissertation is organized as follows. In Chapter II we provide a more detailed overview of the research challenges to be addressed in this dissertation, discussion of the scope of this dissertation, and basic yet critical definitions. In Chapter III, we provide a systematic review of foundational research and more recent research related to online trust metrics. In Chapter IV, using the research landscape understanding developed from Chapter III, we perform benchmarking of different trust metrics to understand their accuracy, their code execution speeds, and how these two measures are affected by different graph structural characteristics. In Chapter V, we apply our insights from the two preceding chapters to examine what the relationship is (if any) between trust levels and the spread of misinformation in an online social media platform (Twitter). In Chapter VI, we apply our insights from preceding chapters to examine what the relationship is (if any) between trust levels and cybersecurity incidents (measured using CVEs, or Common Vulnerabilities and Exposures) in GitHub projects. In Chapter VII, we take our learning and insights from the preceding dissertation chapters and apply them to the design of a trust tool for the online social network and news aggregator platform, Reddit. Finally, in Chapter VIII we summarize our key findings, provide further interpretation and discussion of them from an integrative perspective, and summarize future research directions resulting from this dissertation.

While each of the realms which we research are different (Twitter, GitHub, and Reddit), the common thread across all of this dissertation's work is trust in networks

(graphs), and how differing trust levels affect and are affected by network structure and performance (which includes security). In this regard, the methods and research questions used in this dissertation can be extended with minor adjustments to also examine similar questions in other technological, social, or sociotechnical networks.

## II. Background and Overview

In this dissertation, we explore several related but distinct aspects of trust in online sociotechnical systems. While much of the research related to online trust management systems has focused on developing more effective or faster algorithms for estimating trust in networks and between individuals, comparatively little research has explored what the organizational or societal *effects* are of high or low levels of trust in online networks.

A trust metric is a method for measuring (or estimating) trust in an online network; a trust management system, in addition to measuring trust, includes a module for making use of estimated trust values to perform actions (such as providing a recommendation to a user) in the online network. [2.1] provides a thorough review of trust management systems designed for application in online social networks. Critically, this work provides an excellent and thorough treatment of trust systems available at the time, breaks the key functions of a trust management system down into three key dimensions: trust information collection, trust evaluation, and trust dissemination. Trust information collection is the critical step of gathering and organizing into a useful format the data that will be needed to compute trust values; many proposed trust management systems take this step as a given, failing to fully consider the data input needs of the proposed system and how the system would obtain data from a real network. These data are often related to historical transactions and ratings

between agents in an online system. Trust evaluation is the function of using the data to compute or estimate trust values between pairs of nodes, and/or for a network as a whole. And, trust dissemination is the function which makes use of the computed trust values with a specific goal in mind – for example, providing a movie recommendation to a user, based on the user’s trust in another similar user.

The uses of a trust management system vary depending on who is applying it. When applied by the owner or manager of a (typically proprietary) platform or network, global trust values can be computed and managed for the entire network (or any subset of interest) to improve aspects of interest such as quality of engagements, amount of engagement, increasing revenue by providing better recommendations, etc. When applied by individuals within a network – which typically means that global network information is unavailable – a trust management system can, among other functions, help to serve as a decision-making heuristic about with whom a user should interact, and at what level (for example, as in the Pretty Good Privacy or PGP encryption schema). Most trust management systems also include a mechanism for attack resistance (this is discussed in greater detail Chapter III).

A rich body of scholarly work on trust metrics by computer scientists, mathematicians, network theorists, and related disciplines has been developed in the past approximately three decades [2.1]. Similarly, much scholarly work related to trust, mistrust, and distrust has been developed over several decades in the fields of psychology, social psychology, sociology, management and organizational behavior, and related areas. While

these works are critical for understanding the nuances and subtleties of how trust forms and evolves in real human interactions, the state of trust metric research isn't (yet) sufficiently advanced to capture the richness and subtleties of these interactions.

Instead, trust metrics model trust relationships in a simplistic way (compared to the messy and possibly unquantifiable nature of real-world human-to-human trust relationships) that allows for computer and information systems to calculate and utilize them to achieve a goal. Why are researchers and practitioners interested in having a way to measure trust in the first place? Understanding trust can help to improve network performance by rewarding trusted and trustworthy nodes, and punishing (or removing) untrusted or untrustworthy nodes.

Understanding trust can also enable *prediction* of trust among network nodes that haven't previously interacted and aren't otherwise directly connected [2.2]. Through so doing, the performance of the network may be improved: users can receive recommendations for items they may enjoy based on trust predictions between them and other users, or they can *avoid* recommendations from users that would cause them to waste time or money.

To be useful, the trust metric or trust management system must be capable of operating efficiently at scale in large networks, it needs to be capable of making sense of historical transactions among agents in the network, and needs to provide results to users and/or managers of networks that are straightforward in their interpretation.

Two of the most common applications of online trust metrics have been seen in supporting and improving recommender systems and reputation systems [2.1].

Recommender systems provide suggestions to users for other items they may enjoy or be interested in. Recommender systems take many different approaches, and trust-based recommender systems are one type. Reputation systems measure and manage online reputations, serving as a guide to users of the network as to whether and how much to trust another heretofore unknown user in a transaction. A common example is eBay's buyer and seller rating mechanism, which gives a prospective buyer a rough guide for whether they can trust a seller of an item. Other less common but no less important applications of online trust metrics have been seen in data communications (PGP/Pretty Good Privacy), email communications, messaging applications (like WhatsApp, which is also an online social network), and others.

More recently, beginning around 2015 computational trust metrics have also seen a spike in interest from the blockchain and cryptocurrency communities. Some blockchain-based projects are applying decentralized computational trust metrics in a variety of ways today. One example comes from work.nation, which is an open source collaboration among Cisco Systems, Comakery, ConsenSys, and the Institute for the Future [2.3].

Work.nation enables users to build a professional work portfolio (for example, JavaScript coding examples) that can be validated by a trusted network of peers. Work.nation uses the Ethereum blockchain, Interplanetary File System (IPFS), and uPort for the infrastructure of its system. To map and manage the trust network, work.nation utilizes Trust Graph,

which is also an open source project that according to their GitHub page, is *“An open protocol; A toolkit for building and reading distributed trust graphs; An ambitious plan to create interoperability between existing and future trust networks; Compatible with existing rating schemes (scores, percentages, star ratings, etc); Open Source (Apache 2 licensed)”* [2.4].

## **Problem statement**

Although we approach the specific research questions of this dissertation using different methods and data, all of them can be distilled down to one simple question: *“how does trust affect the performance of online networks?”*

While much research has been completed to propose and develop new trust management systems for use in different contexts and applications, there is comparatively little neutral, third-party evaluation of these systems' performance. Moreover, there is comparatively little prior research that examines how trust management systems' performance is affected by various network structural characteristics (for example, the presence or absence of small worldness) or network behavior. To address this gap (in part), in Chapter IV we perform benchmarking of several well-known trust metrics, and examine how their performance varies with changes in network structure.

Computational social scientists have completed a great deal of research into understanding the structure and dynamics of massive social or sociotechnical networks. Computational trust researchers have developed numerous methods and techniques for measuring trust in different systems. There has been comparatively little work done at the *intersection* of these two domains to draw insights from social or sociotechnical networks

regarding if and how varying levels of trust in these networks affects their performance and security. We explore these questions as part of Chapter V and Chapter VI.

There has been much attention paid to developing trust management systems for various peer-to-peer (P2P) online applications (including filesharing and e-commerce), for online social networks including Twitter, WhatsApp, and Facebook, as well as for technological networks like wireless or mobile ad hoc networks (WANETS and MANETS), distributed computing networks, and even blockchain networks. Surprisingly, there has been comparatively little trust metric research performed with respect to Reddit, which is one of the Internet's ten most frequently-visited sites, putting it in the company of other such well-known sites as YouTube, Wikipedia, Facebook, Google, Amazon, and Twitter. To address this gap, in Chapter VII we apply the Systems Engineering Process to the design, prototype, and testing of a trust management tool specifically designed for the Reddit platform.

#### *Research questions investigated in this dissertation*

Through the course of this dissertation, we propose and explore several research questions, which all relate to the core question of this dissertation described in the previous section. In Table 1, we present each of the research questions investigated in greater detail in their respective chapters of this dissertation.



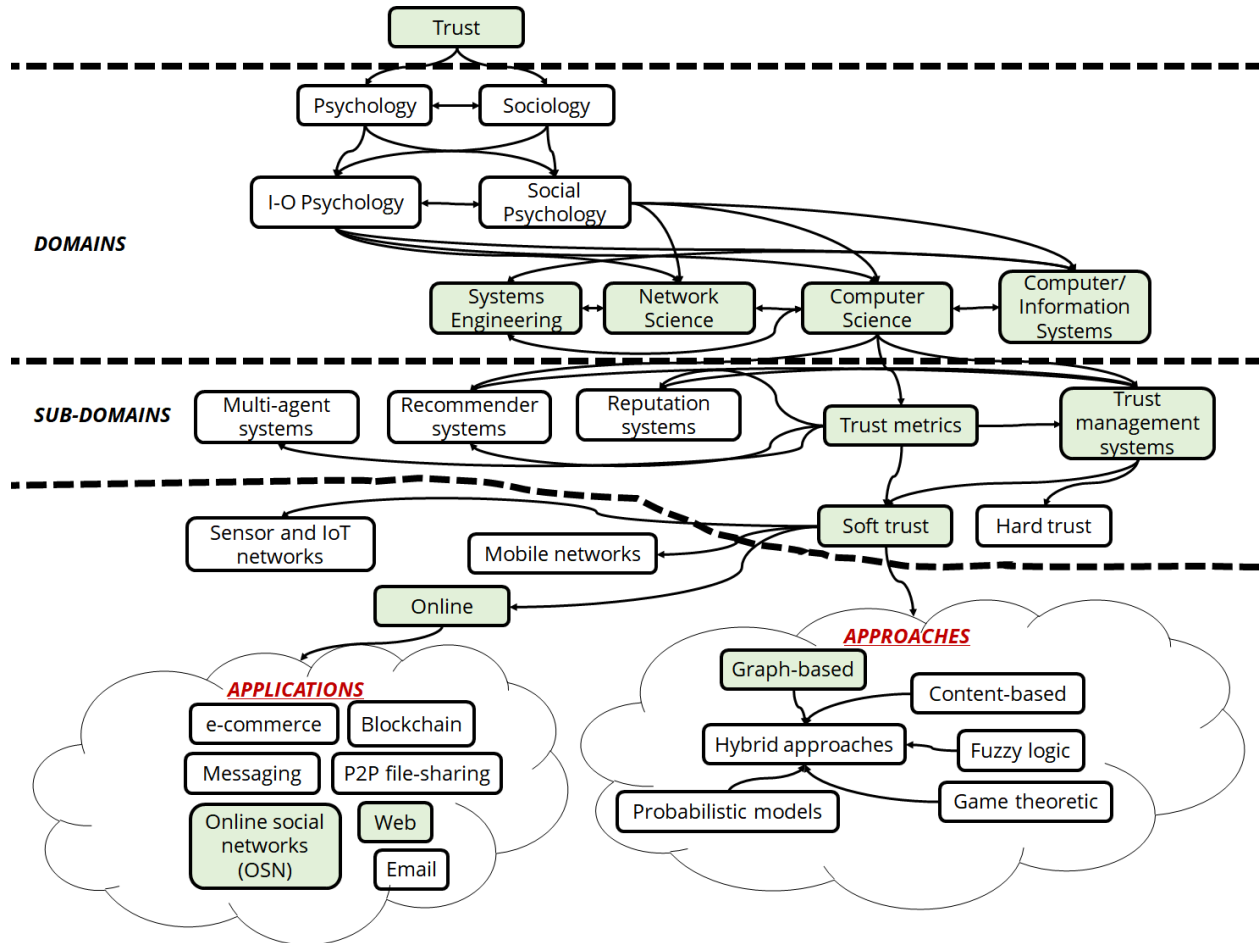
**Table 1:** Research questions investigated in this dissertation

<b>RQ4-1:</b> How do various trust metrics' accuracy in estimating the ground truth trust values vary with different graph structural characteristics?
<b>RQ4-2:</b> How do various trust metrics' code execution times vary with different graph structural characteristics?
<b>RQ4-3:</b> Can we identify approximate ranges of values that balance accuracy with speed of code execution of trust metrics for use in online networks?
<b>RQ4-4:</b> Can we develop a generalized framework for selecting which trust metrics or trust management systems are best suited for use in different applications?
<b>RQ5-1:</b> What is the topology and structure of an online misinformation trust network? How is it similar to, and how does it differ from a regular information trust network?
<b>RQ5-2:</b> Does trust lead to the network structures that we see in an online misinformation network, or do misinformation networks lead to the types of trust relationships that we observe?
<b>RQ5-3:</b> How do the structures of misinformation trust networks influence their behavior?
<b>RQ6-1:</b> What is the structure and topology of a trust network in free and open source software (FOSS) projects?
<b>RQ6-2:</b> Does trust lead to the network structures that we see in online FOSS networks, or do FOSS networks by their nature give rise to the types of trust relationships that we observe?
<b>RQ6-3:</b> What is the relationship (if any) between the trust network and security incidents in FOSS projects?

## **Scope**

To more clearly illustrate the boundaries of what is and what is not included within the scope of network-based trust for this dissertation, we produce Figure 1. The figure illustrates the different disciplinary perspectives on trust, and which of them are directly included in this dissertation. Objects colored green in the figure are those which are included as part of the scope of this dissertation; objects which are shown in the figure but

are colored white have a strong influence on the work of this dissertation, but are not directly included as part of the scope of this dissertation.



**Figure 1:** Visual overview of the universe of trust; green objects indicate those that are within the scope of this work

### **III. Systematic analysis of trust management systems for online applications**

Over the past decades researchers from computer science and related fields have proposed a wide range of methods for computing trust in online settings. These methods are referred to by terms including computational trust (typically reserved for when trusted nodes are generated using cryptographic methods), trust metrics, trust computation, trust management, trust management system, trust management framework, trust management model, trust model, and similar. These terms each have slightly different meanings, but they all have the aim of enhancing understanding the levels of trust within complex systems composed of people, machines, or people and machines acting together.

For the purposes of this chapter, only trust metrics that can be readily applied in *online* settings are considered. These range from online social networks (OSNs), online communication networks (which oftentimes work in direct conjunction with OSNs, as in the case of WhatsApp, which has features of both an OSN and a communication network), online file sharing networks, online privacy and authentication (as in PGP, or Pretty Good Privacy), e-commerce, and others – including more recently, a great deal of interest in trust metrics from the blockchain communities.

Generally, a trust *metric* is used to measure or estimate trust within a system, while a trust *management system* or trust management framework will go a step further by

utilizing the results of trust metrics to improve the performance of a system. Trust management systems generally are made up of two components: a trust computation component (generally utilizing a trust metric or metrics), and a trust manipulation system (used to make use of the trust computation to improve some aspect of the network's performance).

Scholars and practitioners alike are interested in understanding and improving trust online for a variety of reasons. In the case of OSNs, one important reason for understanding trust from the perspective of the operator of the OSN (in cases where they are proprietary, such as Facebook or Twitter) is to improve the recommendations offered to the user. In these instances, trust is used as another method of improving recommender systems – particularly in cases where there is sparse data available. In cases where, for example, users have not yet provided a sufficient number of reviews of products in certain categories to be able to make content-driven recommendations, trust-based recommendations are often successful at providing useful recommendations. Such recommendations are made to users based on how much they trust (or, how much *implied* trust is computed) they have for recommendations from other types of users.

Trust management systems are also useful as a type of reputation system. One of the earliest applications of online reputation systems at large scale was seen with eBay, where buyers rate sellers based on the quality of products sold as well as the quality of the transaction itself. In reputation systems, trust can be used either explicitly or implicitly to improve the experience for users and/or for the operator of the network. In the explicit

case, some online platforms such as epinions.com or FilmTrust allowed users to directly indicate if they trusted another user, and how much. These explicit ratings could then be used to provide other services to users based on implicit trust; for example, User A may be likely to trust User C's recommendations even though they've never interacted before, because both C and A trust User B.

Another reason for the interest in trust metrics and trust management systems is to improve the performance of the network. Trust management systems have been successfully applied to reducing spam both on websites and in email networks, in reducing the frequency of fake file transfers in online peer-to-peer file sharing networks like Gnutella, in improving security in email encryption schemes, and some researchers are currently investigating how trust can be incorporated into telehealth applications to improve health outcomes [3.1].

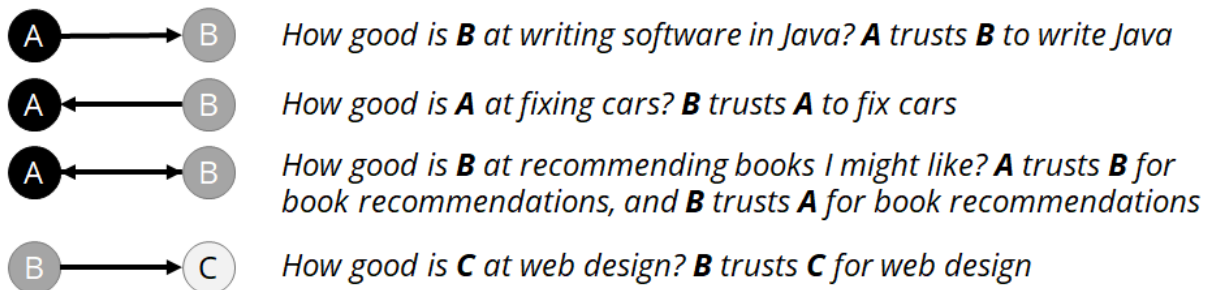
For most trust management systems, a critical consideration is not only computing and applying trust in a network, but also resisting attempts by malicious nodes to attack or game the system. Many trust management systems include mechanisms to reduce the effectiveness of coordinated actions by malicious nodes, such as the Sybil attack wherein attackers create a large number of fake accounts to gain outsized influence in the network.

One of the earliest wide-scale uses of trust online was seen in the Advogato online community. Advogato was an online community focused on the sharing, distribution, and discussion of free software. Launched in the late 1990s, Advogato persisted until 2016, when it was discontinued. One of the key elements of Advogato was its trust metric, which

was intended to reduce the frequency of problems that are typically observed in open online communities. The Advogato trust metric, called *mod\_virgule*, is implemented as an Apache module in the C programming language, and was used to improve the user experience until 2006 [3.2]. Another even earlier application for online trust was seen in the early 1990s with the launch of PGP (Pretty Good Privacy).

### **Types of and properties of trust**

Generally, definitions of trust when applied to online settings includes reference to beliefs, actions, and context. Beliefs are opinions held by users or nodes in the network with respect to other users or nodes in the network; many trust management system model beliefs probabilistically. Actions are carried out by nodes in the network, and can be things like clicking on links, retweeting a status (on Twitter), liking a post (on Facebook, and others), reviewing a transaction, and so forth. The context is the context in which trust is given, a critical distinction because a user that trusts another user in one context may not trust the same user in a different context. In Figure 2, we see examples of how and why different nodes may trust or not trust one another depending on the context.



**Figure 2:** Examples of trust in different contexts

One of the definitions that is typical of the approach taken by many trust metrics and trust management systems comes from [3.3]:

*"trust in a person is a commitment to an action based on a belief that the future actions of that person will lead to a good outcome. The action and commitment does not have to be significant. We could say Alice trusts Bob regarding email if she chooses to read a message (commits to an action) that Bob sends her (based on her belief that Bob will not waste her time). Since trust is usually not a simple binary value (trust or no trust), we consider it to be similar to a probability that a person will commit to an action."*

Just as there are different types of trust management systems, so too are there different types of trust and different properties of trust that have been identified and described by scholars.

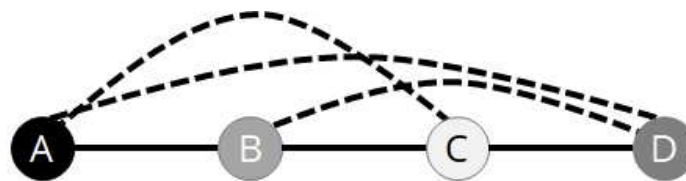
### Quantitative scale of trust

Researchers have proposed varying methods for how to quantitatively represent trust in a network, but by far the most common method is to represent trust on a continuous range from 0 to 1. In most trust systems, a trust value of 0.5 represents a neutral state, values below 0.5 represent low trust, and values greater than 0.5 represent high trust. Additionally, most systems consider a binary understanding of trusted and not trusted – rather than trusted, not trusted, and distrusted. The distinction between not trusted and distrusted is subtle but important; most trust systems that don't include a state for distrust assume that nodes that should be distrusted are ignored, and thus they can be treated as untrusted. Nonetheless, increasingly in the past decade researchers have been

explicitly considering and modeling distrust, particularly in applications with direct security implications. Other systems take a different quantitative scale such as 0 to 10, -10 to 10, and other variations. Fuzzy-based systems represent trust levels linguistically rather than quantitatively, using terms such as “no trust”, “low trust”, “high trust”, or “complete trust”.

### Transitive trust (property)

One property of trust which is frequently used in trust management systems for online settings is transitive trust. Transitive trust makes the assumption that trust, to a certain extent, can be “transmitted” from one node to another node which don’t share a direct connection. For example, in Figure 3 we give a simple illustration of a small trust network composed of nodes **A**, **B**, **C**, and **D**. Each edge, or connection, between nodes indicates a trust relationship. We see that **A** is directly connected to **B**, but is also indirectly connected to **C** and **D**, through **B**. In the presence of transitive trust, **C** effectively transmits a portion of the trust it has for **D** to both **B** and **A**. The dashed lines in the figure indicate transmission of trust between indirectly-connected nodes.



**Figure 3:** Illustration of transitive trust concept in a small trust network. The solid black lines represent a direct trust relationship, while the dashed lines indicate transitive trust.

Two key functions that trust metrics that make use of the transitive property of trust must incorporate are a propagation function and an aggregation function. In fact, the usefulness and validity of any transitive trust measure is directly related to the



connectedness of a network; without proper limits on trust propagation and trust aggregation – limits which may derive from small worldness of the network being investigated – too many nodes may be counted as trusted nodes, causing such a measure to lose its meaning and usefulness.

### Trust propagation function

Much work has been developed by scholars specifically to identify the best way to transmit – or propagate – trust among nodes that aren't directly connected. Part of this includes determining how far along a network path to transmit trust; this is often referred to as the maximum trust propagation distance (MTPD). Numerous theoretical solutions to choosing the optimal MTPD have been proposed, but many of these face the real-world constraint of being too computationally-intense to be useful for practical applications in large networks. [3.4] proposed a useful and novel approach to this issue by setting the MTPD as the ceiling of the mean path length of the network. In their paper, they find evidence for small worldness in the empirical networks they analyze, and they use this result to set MTPD as approximately equal to the mean path length of the small world network. Doing so enables for a reasonable amount of trust propagation within the network, while drastically reducing the problem space and thus, the computational time.

### *Decay of trust*

Related to the propagation distance of trust, researchers have also identified the issue of decay of trust. This closely related idea states that even within an MTPD, trust need not be weighted equally along all links of the trust path. If two nodes A and B are separated

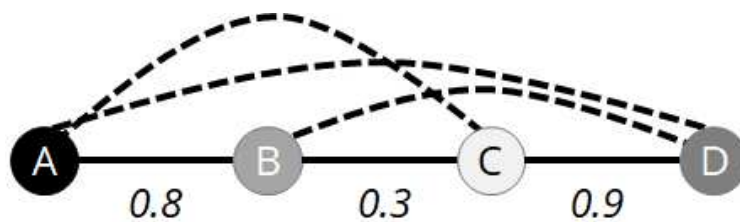
by five links of equal trust value, the first link's trust value would be considered to have little or no decay, while the fifth link in the path would decay to some portion between less than 100% and greater than 0%. Some trust management systems don't incorporate trust decay, while others do so using, for example, a staged decay model where links that are two or fewer hops from the sink experience a small level of decay, links that are three to five hops from the sink experience a moderate level of decay, and links that are more than five hops but less than the MTPD experience a high level of decay.

### Trust aggregation function

A directly related issue to MTPD is how to treat each step along a path between indirectly connected nodes. Logically speaking, it doesn't make sense to give full "weight" of a trust relationship from distantly-connected nodes when computing transitive trust. Thus, most scholars include a method to reduce the trust the further out a source node gets from the sink node. Methods generally utilize various arithmetic operators, with multiplication being one of the most common.

Figure 4 illustrates a simple example of this issue. Many trust management systems quantify trust in the range from 0 to 1. Doing so attempts to capture the belief and uncertainty in that belief that a node has that another node will take an action, as described in the definition above from Golbeck. Thus, the value taken by the trust relationship is often shown as a weight on an edge when modeled as a network, where the trust value represents the strength of the belief the node has in the other. In the figure, A mostly trusts B with a value of 0.8, B somewhat trusts C with a value of 0.3, and C mostly

trusts D with a value of 0.9. In this example, if using the simple method of multiplying trust probabilities across a path, using the principle of trust transitivity we would infer that A trusts D with a value of 0.22 ( $0.8 \times 0.3 \times 0.9$ ). By using this type of method for trust transitivity, scholars have been able to utilize some of the implied trust in a network among indirectly connected nodes, while keeping the inferences relatively reasonable. Some scholars' trust management systems take a more sophisticated approach to transitive trust computation, weighting the weights according to how far they are from the sink node. For example, a source node that is only one edge away from the sink node might have its full edge weight included, while a source node that is three edges away from the sink node may only have half of its edge weight included in the transitive trust calculation.



**Figure 4:** Illustration of transitive trust concept in a small trust network. The values placed on the edges signify the trust that exists between the corresponding nodes.

#### Direct and indirect trust

The simplest form of trust evaluation in an online network is that of direct trust. Direct trust is easily observed, evaluated, and quantified. For example, if on an e-commerce platform a buyer gives a five star rating to the seller they just received an item from, we can directly observe that this buyer trusts (or has trusted) this seller a great deal. Another example

would be on a platform like epinions.com (now defunct), where users explicitly indicated how much they trusted the opinions of other users that they followed.

Indirect trust, on the other hand, is more difficult and subjective to compute (or to infer), yet it is the more important of the two in trust management systems that are to be used as part of recommendation systems or reputation systems. Indirect trust infers the level of trust that two unconnected users would be *expected to have* in one another, were they to actually connect. Indirect trust is inferred in many different ways (described in greater detail later in this chapter) using different features such as history, behavior, or graph structure, among others.

### *Distrust*

In the early trust metric and trust management literature most researchers opted to simplify their analyses and designs by lumping distrust and non-trust into the same bucket, thereby not considering the distinct case of distrust. Doing so simplifies the designs, no doubt – which is a tradeoff directly related to the previously mentioned objective of reducing computational complexity – but for certain applications it can come at the expense of losing important information about the system being considered. More recent research (in the past decade or so) has begun to re-incorporate a concept of distrust into systems design. While untrusted or non-trusted is essentially a neutral trust stance (for example, it may be a node that is unknown to another node, and therefore a trust evaluation isn't made), a state of distrust specifically gives a special (negative) status to nodes that are distrusted, changing how they are accounted for and managed in these

systems. Distrust may arise when actions with negative consequences are performed by these nodes, or distrust can arise based on the origin of these nodes (for example, IP addresses coming from countries that are specifically blacklisted).

### *Global trust and specific (or local) trust*

Another distinction commonly seen in trust management systems is that between global trust and specific (or local) trust. Global trust takes a high level, zoomed out view of trust in a given network, calculating the total trust (direct and/or indirect) that all nodes in the network feel towards all other nodes, summing and normalizing these values. Global trust is useful from the perspective of the owner or manager of an online network, when direct access to all necessary data is not a barrier. Global trust can also be computed and managed in open systems (like P2P systems), but it becomes more challenging. EigenTrust is designed for use in open, P2P systems where no single node has a bird's eye view of the entire network, and achieves its aim of global trust computation by distributing different pieces of the storage and computation responsibilities.

Specific or local trust, on the other hand, is the (direct or indirect) trust that Node A has or should expect with respect to Node Z. Specific trust is of greater interest to users of networks, as this is how they can actually make an assessment of if they should trust a seller of a product in an e-commerce site, a disseminator of a claim on a social media platform, and so on. In the case of specific trust – depending on the trust system being considered – each user would have a trust matrix that records the trust value that they feel towards all other users (many of these would be zeroes).

### Swift trust

Another type of trust is swift trust. Swift trust develops among individuals and in groups based on shared tasks or goals, rather than history or demographic similarities [3.5]. Swift trust is frequently observed in online collaborative communities, such as open source software development projects, and it is becoming an increasingly important type of trust to understand in online contexts as activities move to being online first with, in many cases, people who have never met in person. [3.6] identify two specific types of swift trust: encapsulated interests, and cognitive trust. Encapsulated interests is essentially a rational approach to trust: Alice trusts Bob because she feels she must, and she feels Bob must in turn trust her. Cognitive trust develops based on the characteristics of people working in the project.

### Hard trust and soft trust

Within computer and information systems, there are two “firmnesses” of trust: hard trust and soft trust. Hard trust is explicitly defined and enforced using security and authentication policies; users can only do the things that the policy allows them to do. Soft trust, on the other hand, is [3.7] based on the behavior, experiences, and/or reputations of agents within a system.

Hard trust is generally not included in the scope of this dissertation, but this section provides a very brief reference to some of the hard trust schemas for online contexts. IBM’s z/OS – an operating system for IBM mainframe computers – utilizes concepts of trust including peer trust and transitive trust for its integrated security services [3.8]. Microsoft’s

Active Directory service utilizes concepts of hard trust, including trust transitivity, external trust, realm trust, forest trust, and shortcut trust [3.9]. PowerSploit (a PowerShell post-exploitation framework) utilizes concepts of hard trust for red teams to help organizations improve their security stances in its PowerView module [3.10].

### **Realms of trust**

While most trust management systems are unable to directly distinguish among these different realms of trust, they are nonetheless important to highlight because they ultimately are represented in different online platforms.

Personal trust is observed among individuals, and can be one to one, one to many, or many to one. Organizational or institutional trust is observed within a discrete organization and, while it is built on many personal trust relationships, its result is an emergent level and type of trust that is different than the mere sum of its parts. Organizational trust, depending on the context, may also be observed inter-organizationally – trust from one organization in another, and vice-versa.

Regarding institutional trust, [3.6] state *"it is important to note that trust in open source communities is rather institutional than personal. The number of participants to a given open source project is often very large and the communities are open to exits and new entries. Thus the development of trust cannot be based on repeated interactions of the same individuals who get to know each other over time and learn to trust each other."* Societal trust is more abstract, but its analogue tends to be an important factor in massive online social networks like Facebook or Twitter.

Societal trust relates to the general level of trust that individuals within a society have towards others whom they have never met and don't know within that same society. For example, Independent Society (a coalition of not for profit and philanthropic organizations) conducts an annual survey of trust in civil society.

### **Key objectives of trust models**

While the approaches used to compute and manage trust differ (discussed in greater detail later in this chapter), most trust models share common objectives and principles.

The first and most obvious of these objectives is that they aim to accurately calculate or estimate levels of trust within a network. Some trust systems focus on global trust (understanding the total trust placed in a given node by all other nodes, from the perspective of a network manager or owner, or in P2P systems), others focus on specific trust (trust that one specific node places in another specific node), and some consider both global and specific trust.

Another common objective in trust systems is to resist attack or manipulation. While the motivations for attempting to subvert a trust, reputation, or recommendation system vary depending on the application (e-commerce, social media, etc.), in most networks – and in all open networks – there are always a certain proportion of the total userbase who aim to subvert the trust system using various strategies. The first distinction that can be drawn among attempts to manipulate a network is between uncoordinated and coordinated attacks. One common strategy in the coordinated category is known as the Sybil attack,



wherein a set of fake accounts or profiles are created with the aim of gaining influence in the network. In an e-commerce setting, for example, this can take the form of this large group of fake accounts giving fake reviews of products or merchants, and in social media platforms like Twitter it can take the form of coordinated groups of bot accounts all pushing the same story. Other types of attacks (not an exhaustive list) include self promotion, where a coordinated group of malicious nodes works together to boost their own reputations; slandering, where a coordinated group of users works together to damage the reputation of other non-malicious nodes (analogous to distributed denial of service, DDoS, attacks); whitewashing (may be coordinated or uncoordinated) wherein malicious nodes behave badly for short term gain, then leave the network and reenter with a new identity; and ballot box stuffing, wherein malicious nodes coordinate to provide more ratings, votes, evaluations, or similar actions than would otherwise be expected or permitted [3.1].

A third common objective of trust management systems is that they be computationally efficient. Because the majority of trust management systems for online use are designed to be applied in massive complex networks, systems designers must be conscious of the number of operations and iterations their systems are performing. Many research projects related to trust management have identified optimal systems or algorithms that maximize accuracy of trust evaluations or robustness to attack, but do so in a way that isn't feasible for real-world deployment in terms of computational complexity. Thus, some researchers look to simplify existing trust systems to make them feasible for

real-world deployment, while other researchers develop approximations of their optimal systems that trade a small decrease in effectiveness for large improvements in execution time.

### **Application context**

Generally speaking, the most successful, actionable and useful trust metrics and trust management systems are those designed for and used in a specific application context.

In the first wave of online trust systems which were seen roughly from the early to the mid 1990s, online trust systems were applied to email encryption schemas and online communities.

In the second wave of online trust systems, which were seen from roughly the late 1990s to the mid 2000s, a large proportion of online trust systems shifted their focus to peer-to-peer filesharing networks like Gnutella, Kazaa, etc. Many others focused on e-commerce platforms like eBay, and online communities like epinions.com.

By the late 2000s and early 2010s, online social networks (OSNs) like Facebook and later Twitter started to garner the attention of more researchers who were working in online trust systems, and online communities remained popular targets for design of online trust systems too.

Today (2022), OSNs are still one of the most important categories for application of online trust systems, but other areas like blockchain and messaging networks like

WhatsApp (which have characteristics of OSNs but also of more traditional communications networks) have also started to garner more attention.

Additionally, IoT and sensor networks have become an area of much interest for application of trust systems, but these types generally skew more toward the hard trust end of the spectrum, rather than soft trust which is the focus of this dissertation.

### **Systems or algorithms for measuring and managing trust online**

In the next section of this chapter we present an extensive treatment of the different methodological approaches that have been used for designing online trust management systems and trust metrics. In this section, we first present a short summary of many of the most influential trust systems since 1999 in the form of Table 2. While this list is not exhaustive, to the best of our knowledge it gives the most extensive listing of trust metrics and trust management systems in one place. The list begins with PageRank which, although not explicitly designed as a trust metric, it has nonetheless been used by many researchers as an approximation for trust. The most influential system specifically designed for trust management is EigenTrust (which is similar in many ways to PageRank).

**Table 2:** Selected trust management systems and trust metrics since 1999

<b>System or algorithm</b>	<b>Authors</b>	<b>Year</b>	<b>Citations</b>
PageRank	Page, Brin, Motwani, Winograd	1999	16723
EigenTrust	Kamvar, Schlosser, Garcia-Molina	2003	5513
PeerTrust	Xiong & Liu	2004	2543
TrustRank	Gyongyi, Garcia-Molina, Pedersen	2004	1544
FuzzyTrust	Song et al	2005	614
TrustGuard	Srivatsa, Xiong, Liu	2005	457
TidalTrust	Golbeck	2005	1226
AppleSeed	Ziegler & Lausen	2005	469
TNA-SL	Josang, Hayward, Pope	2006	538
PowerTrust	Zhou & Hwang	2007	1012
SUNNY	Kuter & Golbeck	2007	318
PageTrust	Kerchove, Dooren	2008	109
TrustWebRank (TW)	Walter	2009	1
Lesani & Montazeri	Lesani & Montazeri	2009	70
R <sup>2</sup> Trust	Tian & Yang	2011	53
SWTrust	Jiang & Wang	2011	39
STrust	Nepal, Sherchan, Paris	2011	108
IRIS	Hamdi et al	2012	34
DHTrust	Xue et al	2012	30
@Trust	Meng, Ding, Gong	2012	23
EigenTrust++	Fan et al	2012	25
HonestPeer	Kurdi	2015	46
PHT	Liu, Parks, Seuken	2016	15
TOSI	Liu et al	2016	36
3VSL	Liu et al	2019	24
CoTrRank	Li et al	2019	2
TrueTrust	Meng & Zhang	2020	7
WhatsTrust	Almuzaini et al	2020	2
TMLP	Kou et al	2020	3
DeciTrustNet	Ureña, Chiclana, Herrera-Viedma	2020	6
InterTrust	Wang & Nixon	2021	3
AntTrust	Al-Otaiby, Alhindi, Kurdi	2022	1

In addition to the listing of systems given above, in Table 3 we also give a more detailed accounting of some of the more influential trust metrics and trust management systems. Through this table we intend to give a better high level understanding of what the different features of the systems are in terms of the type of trust outputted (global, specific or both), the time complexity of running the systems, the original application context that they were designed for (though many of them can and have been adapted to different contexts), and the parameters that are required as inputs to be able to run the trust computations. In addition to the systems enumerated in the table, dozens of other systems have been published by researchers and practitioners alike. In the next chapter of this dissertation, we present a decision making framework for practitioners to select a trust model, using this table and the features enumerated in it.

**Table 3:** Features review of selected trust metrics and trust management systems, in descending order of influence (as measured by number of citations)

System or algorithm	Trust computation	Trust manipulation/ trust management	Global, specific, or both	Time complexity	Original Application Context	Input Features: Behavior (B), Context (C), Graph Structure (G), History (H), Similarity (S)
@Trust	x	x	B	$O(n)$	P2P: file-sharing	B, H
AppleSeed	x		S	n.s.	OSN	G
EigenTrust	x	x	G	$O(n^2)$	P2P: file-sharing	B, G, H
EigenTrust++	x	x	B	$O(n^2)$	P2P: file-sharing	B, G, H
FuzzyTrust	x	x	B	n.s.	P2P: e-commerce	B, H
IRIS	x		B	n.s.	Web	B, C, G, H
PageRank	x		G	$O(n+m)$ <i>n=number of nodes, m=number of links</i>	Web	G
PageTrust	x		G	$O(kn\underline{n})$ <i>k, n and <math>\underline{n}</math> are respectively the mean degree, num. nodes w/negative links, total nodes</i>	Web	G
PeerTrust	x	x	B	$O(N)$ <i>N = number of nodes</i>	P2P: e-commerce	B, C, G, H
PowerTrust	x	x	B	n.s.	P2P	G
R <sup>2</sup> Trust	x	x	G	n.s.	P2P	B, H
STrust	x	x	B	n.s.	OSN and online communities	B, C, G, H, S
SUNNY	x		G	n.s.	OSN	C, G, S
SWTrust	x	x	B	n.s.	OSN	B, C, G
TidalTrust	x		S	$O(2n)$	Online communities	B, G, H, S
TNA-SL	x		B	$O(2n)$	Generic (trust transitivity analysis)	B, C, G
TrustGuard	x	x	B	n.s.	P2P	B, C, H, S
TrustRank	x		G	n.s.	Web	C, G

## **Approaches to computing trust**

Different methodological approaches have been applied to computing trust metrics and managing trust in online settings. Each approach brings its own advantages and drawbacks. Most of the trust metrics compute trust using one of the following five approaches, or a combination thereof: 1) graph-based, 2) content-based or behavior-based, 3) probabilistic models, 4) fuzzy logic, 5) and game theory. Although later sections of this dissertation focus attention primarily on graph-based approaches to trust, approaches from the other categories are included here for comparison, benchmarking and, in some cases, because they can be adapted to fit a graph-based approach. A brief overview of each

approach is given here, and in the next section specific papers from these categories are reviewed in greater depth.

Graph-based (or network-based) approaches to trust metrics calculate direct trust and/or infer indirect trust using information about the structure, topology, and/or dynamics of a system modeled as a graph. A popular subset of graph-based trust measurement approaches are also known as flow model algorithms. Typical methods within this approach include equating trust to degree (the connectivity of a given node), various centrality measures (in particular, eigenvector centrality and PageRank), and path-based methods which are particularly well-suited for computing transitive trust. A well-known example of a graph-based trust metric is the EigenTrust system, which relies on computing the left principal eigenvector of a matrix representing a trust matrix. EigenTrust is similar to PageRank, which is a frequently-used centrality measure and ranking technique in network science (and which some researchers use as a trust metric itself).

Content-based or behavior-based approaches to computing trust rely on analyzing the content of transactions and relationships among agents. In particular, these methods tend to rely on artificial intelligence and machine learning techniques such as natural language processing, classification, regression, clustering, decision trees, and similar.

Probabilistic approaches to computing trust in online settings generally make use of either Bayesian networks and Bayesian probability, or subjective logic. While the total number of these types of systems is less than those that use a graph-based approach, one of the most

influential models – Trust Network Analysis with Subjective Logic, or TNA-SL – comes from this category.

Game theoretic approaches use the methodologies and insights from Game Theory to design trust management systems, with one of their key differentiators being the inclusion of punishment mechanisms, and/or incentives, to modify behavior of nodes in the network. Many game theoretic approaches to trust computation attempt to reduce freeriding behavior, which is seen when a node benefits from engaging with a network but doesn't contribute back to the network. An example of a game theoretic approach is seen in [3.11].

Fuzzy logic approaches to trust computation and management avoid the crisp representation (i.e., quantitative representation) of trust values, opting instead for linguistic representations which are closer to what real-world users of most online systems would expect to use. Whereas in most other trust systems a trust measurement might take the value of 0.8, in a fuzzy logic-based trust system it might instead take the form of “highly trusted”. An example of a fuzzy logic-based trust system comes from [3.12].

Additionally, many trust metrics utilize a hybrid approach, combining elements of two or more of the approaches listed above. For example, [3.13] uses a combination of natural language processing (a form of AI) and graph-based approach to quantify trust in GitHub. Another example comes from CoTrRank [3.14], in which the authors propose a trust metric that utilizes the sigmoid function (frequently used as an activation in artificial



neural networks in machine learning) combined with graph-based techniques to rank users on Twitter.

Finally, we also consider other perspectives on online trust which are not, in most cases, trust metrics or trust management systems, but which nonetheless have either been influential in the history of development of trust systems or which have other important effects on how researchers think about trust online.

### Graph-based approaches

[3.15] proposes EigenTrust, one of the earliest and most influential trust metrics. The EigenTrust paper, published in 2003, is one of the most influential in the area of trust metrics having been cited more than 5400 times by other researches as of June 2022. It was originally proposed to mitigate issues surrounding inauthentic file uploads/downloads in peer-to-peer (P2P) networks which were growing rapidly in popularity at the time, thanks to services like Napster, Gnutella and the many derivative services that ran on the Gnutella network protocol. Although P2P networks were the original application, EigenTrust has been extended and applied for use in many other contexts. In such P2P networks, some nodes would host inauthentic files – files with incorrect names and/or descriptions – with intentions varying from self-interested but non-malicious, to malicious nodes attempting to spread worms, viruses, and similar. The networks needed a way to combat the work of these fake files, and EigenTrust was one method proposed for doing so. EigenTrust is a global measure that assigns a trust value to all nodes in the network, based on their upload history. EigenTrust was also designed to be resistant to attempts to subvert the system,

though numerous improvements have been proposed to further reduce the ability of malicious nodes to bypass the system. The authors lay out five key issues that should be addressed by any P2P reputation system: (1) a self-policing mechanism; (2) the maintenance of anonymity; (3) the absence of an advantage conferred by “newcomer” status in the P2P network; (4) minimal overhead to initiate and maintain; and, (5) robustness to malicious collectives of peers. EigenTrust provides a global trust score for each node in a network; the trust score of a node comes from the local trust scores assigned to it by the node’s neighbors. The trust scores given by the node’s neighbors are weighted based on the neighbors’ own trust scores, and in this way it is similar to PageRank. Local trust scores are provided by the nodes in the network; when a download is successfully completed and the file is verified as authentic, the node receiving that file can verify the node sending the file as being trustworthy. EigenTrust normalizes the trust values as a way to mitigate the likelihood of malicious peers working in concert from arbitrarily assigning high trust scores to themselves. Normalizing the trust scores brings several drawbacks and limitations (which were improved upon by later trust metric scholars), including the lack of any absolute interpretation of the trust values, and lack of a way to distinguish between nodes who had many ratings compared to nodes who had few ratings (this information is lost upon normalizing). Nonetheless, the normalization also allows for more efficient computation of the global trust values, which is a non-trivial consideration. EigenTrust also proposes a mechanism to mitigate the likelihood of malicious nodes gaming the system. A selected subset of peers are specified as score managers, who are queried by nodes in the network for trust scores of peers. In the event

of faulty trust scores being provided by malicious peers who are among score managers, a majority vote of the score managers settles the issue. This mechanism reduces the likelihood of malicious nodes succeeding, but it does not eliminate it. The resulting global trust scores have two important effects in P2P networks: it helps to isolate malicious peers, and it provides an incentive for free riders to share files.

In [3.16] the authors propose a new trust metric, PeerTrust, for use in online peer-to-peer (P2P) e-commerce communities, but which could potentially be extended for use in other contexts too. The authors first identify several problems with other trust frameworks at the time of the paper's publication. These shortcomings included an inability to distinguish honest from dishonest feedback, an inability to incorporate different contexts of trust, a lack of incentives for peers to leave feedback on others, and an inability to effectively deal with dynamic behavior of malicious peers. The authors first identify five factors that they believe are critical for improving a P2P trust system: 1) feedback of one peer from other peers, 2) scope of feedback (i.e., total number of transactions a peer has with others), 3) credibility of the peer providing feedback, 4) the transaction context in which trust is being evaluated, and 5) the community context to account for community-related features. The authors then propose their trust measure, PeerTrust, which takes into account each of these five factors. The authors apply their trust metric to a simulated online community. In the authors' series of simulations, when considering an increasing percentage of malicious peers in an overall P2P network they find that different versions of PeerTrust generally perform better compared to conventional (at the time) trust metrics in

cases of malicious peers. However, in the case where the percentage of malicious non-collusive peers exceeds half of the total number of peers, the PeerTrust measure is less effective than conventional methods. In simulations considering the rate of maliciousness (i.e., the number or frequency of malicious acts of a given peer in a given timeframe), the PeerTrust metric is also robust and features improved performance compared to conventional methods, and in fact features increased performance as the rate of maliciousness increases in both a collusive and non-collusive setting. In simulations considering the number of successful transactions facilitated, the PeerTrust metric demonstrates marked improvement over conventional methods in a non-collusive setting, but no noticeable improvement in collusive cases.

[3.17] proposes the TidalTrust algorithm which, as its name implies, treats transitive trust in a network as a flow that moves back and forth from source to sink. In [3.18], the authors propose a new trust metric, PageTrust, which is based on the PageRank algorithm. The PageTrust system extends the PageRank algorithm by allowing for negative links, which are not included in the original PageRank algorithm. The PageRank algorithm utilizes a trust matrix; PageTrust introduces a corresponding *distrust* matrix as its key feature and novel contribution. PageTrust can also be used as a global trust metric, or a local (specific) one, depending on the context and the desired outcome of its application. One of the key contributions of this paper is to include a type of memory for the random walker on a graph – a key concept utilized in PageRank and other eigenvector-based algorithms. The addition of memory allows for inclusion of a distrust matrix.

[3.3] examines and compares the effectiveness of recommender systems that are based on similarity (collaborative filtering) and those based on trust. The author posits that systems that rely purely on similarity of users and/or similarity of items being rated are limited both in their range of potential applications as well as in the effectiveness of the similarity measure; recommendations made based on trust, according to the author, may be more versatile and more useful in applications where data is sparse. This paper is unique in that it utilizes data collected from real users of an online platform in which the users explicitly rate how much they trust other users' opinions (ratings of movies). Most other trust metric papers rely on various methods for inferring trust based on actions of agents in a particular system. By performing a set of experiments with real users of an online movie rating and critiquing platform (FilmTrust), the author isolates specific elements of similarity that lead to better predictions of trust among users of online systems. In a standard similarity-based approach, a recommender system would look at ratings of items that two different users give, and if the overall ratings of these two users are similar then they will be predicted to similarly respond positively or negatively to other sets of items. The author finds that rather than relying simply on overall similarity "*our results suggest that overall difference , the maximum single difference in movie ratings , difference on movies to which the user assigned extreme ratings , and the user's average trust value all are factors in how users assign trust.*" Although these findings are specific to FilmTrust and ratings of movies online, the author makes a compelling case that the findings can be generalized to many other online contexts in which users rely on ratings, evaluations, or opinions of other users. Using these findings, the author then attempts to

predict trust among users, based on their similarity. The authors find that trust among users can be predicted better by using a combination of largest single difference in ratings and agreement in extreme ratings.

[3.19] proposes the PowerTrust system. PowerTrust makes use of a key finding that the authors observed specifically in online peer-to-peer (P2P) evaluations on eBay, but which is generalizable to many networks: a power law distribution. The authors reviewed evaluations that peers (buyers and sellers of goods) gave one another on eBay, and discovered a power law distribution in their reviews. The authors use this insight to build their system, taking advantage of this distribution to improve robustness and reduce computational overhead in calculating trust values in a large network. PowerTrust is intended for use in P2P networks without a central authority. Although in the case of eBay – the inspiration for their system – there is a central authority that can see all trust relationships among peers globally, the trust evaluations are still made on a peer-to-peer basis and thus, the authors argue it is still applicable to distributed P2P systems. The authors define power nodes (from power law distribution) as those with a large global reputation score, and rely on these power nodes. The authors collected interactions from more than 10,000 users on eBay, with peer evaluations made between 1999 and 2005. They looked at the peers' interactions through three lenses: feedback frequency, feedback amount, and the ranking index. Upon analysis of the collected data, the authors find clear indications of a power law distribution (a distribution frequently observed in other online contexts), dynamic growth, and preferential attachment. The PowerTrust system consists

of five modules: regular random walk, look-ahead random walk, distributed ranking module, initial global reputation aggregation, and global reputation updating procedure. The look-ahead random walk (LRW) utilizes a trust matrix  $\mathbf{R}$  and a normalized reputation column vector  $V$ . After sufficient iterations of the LRW,  $V$  converges to the eigenvector of  $\mathbf{R}$ . LRW is similar to other Web-based ranking systems, but in addition to its own local trust scores it also considers trust scores of its neighbors' direct trust scores. According to the authors, using the LRW results in a noticeable increase in computational speed both in random graphs and especially in power law graphs, compared to a standard random walk. The distributed ranking module makes use of a distributed hash table to implement ranking. Similar to the EigenTrust algorithm, PowerTrust relies on score managers which accumulate the global reputation for a given node. The module uses locality preserving hashing to select the most reputable nodes. The initial global reputation aggregation module has each node send all of its local trust scores for its out-neighbors to the respective score managers. Then, in the global reputation updating module, all score managers collaborate to identify the power nodes in the network. The authors apply their system in a simulated environment to analyze its performance along three dimensions, compared to the EigenTrust algorithm: convergence overhead, ranking discrepancy, and aggregation error. Convergence overhead considers how quickly the algorithm runs when aggregating the trust scores; ranking discrepancy considers the accuracy of the algorithm compared to the ground truth trust values; and aggregation error attempts to quantify the system's robustness to malicious peers. The authors show marked improvement in

convergence overhead, ranking discrepancy, and robustness to malicious peers (in both collusive and non-collusive scenarios) compared to EigenTrust.

In [3.20] the authors, recognizing some of the shortcomings of other trust metrics that had been developed up until that time, proposed their own metric called TrustWebRank (TW). TW seeks to correct some of the issues that arise from global trust metrics, like EigenTrust – namely, that when being used in recommender systems, trust metrics are most effective when personalized to a particular agent receiving the recommendations. More specifically, three shortcomings that TW attempts to address are uniqueness of solutions (other trust metrics may lead to multiple solutions to the trust vector); inability to combine/compare direct and indirect trust; and normalization of trust measures (which leads to loss of information from the graph). The authors compare existing approaches from collaborative filtering (CF) for providing recommendations, listing some of the CF approaches' drawbacks as having limited usefulness when data is sparse (for example, if trying to recommend a kitchen appliance when users have only rated books). Like EigenTrust, TW still relies on centrality feedback, but it corrects for some of the shortcomings of other trust metrics up until that time. TW allows for cyclic graphs and high clustering coefficients – both of which are effects observed in real world social networks. Utilizing the concept of transitive trust, TW takes into account all paths (where a path exists) on a graph between two pairs of nodes to infer indirect trust between them, assigns a centrality score to a node based on the centrality of that node's neighbors (similar in this respect to PageRank), and provides each agent with a personalized trust score with respect



to a given node. Additionally, TW moves beyond static analysis by allowing for dynamic updating of the trust metrics based on a utility function. The utility function allows agents to update their trust measures of their neighbors based on the quality of past recommendations from their neighbors.

In [3.21], the authors extend previous work in which trust is propagated in a way similar to PageRank, but the authors of this paper consider the reverse: propagation of distrust. The authors outline three categories of actors who might be interested in sowing distrust in a network: 1) “bad guys” – self-serving actors who have their own agendas, 2) “careless big mouths” – actors who are “vocal and opinionated” while also being “careless and unreliable”, and 3) “polar opposites” – observed in bipartite networks. The authors propose that, in cases where distrust is present, the distrusted node should be ignored, not negatively weighted. Mathematically speaking, this means that distrusted nodes should have a trust score of 0, rather than -1. The authors propose a system that is based on a modification of the PageRank algorithm, in which no node receives a negative trust score. The authors posit that distrusted nodes should be treated as being either arbitrary or adversarial. The authors’ system relies on use of six key axioms: 1) parallel edges are merged or canceled, 2) items that are distrusted are independent from one another, 3) edges can cancel out one another, 4) a method for propagating trust, 5) self trust can be ignored without changing the overall performance of the system, and 6) weighted majority.

In [3.22], the authors propose a trust management system, SWTrust, to help make recommendations of trust relationships to users in online social networks. The authors

propose their system to remedy a shortcoming that they observe in other systems at the time of publication, which is that other systems rely on generating trust from small simulated graphs which don't reflect the reality of the large, complex graphs inherent in online social networks. The authors' system develops three modules as part of its system: 1) preprocessing a social network, PSN, 2) building a trust network BTN, and 3) generate trusted graph, GTN. The PSN module preprocesses a large social network into a smaller one, so that it can be more efficiently analyzed for the purpose at hand. The BTN module searches for trustor-trustee paths within the simplified network outputted by the PSN. Finally, the GTN module selects short trusted paths from the set of possible paths identified by BTN. The authors evaluate the performance of the SWTrust system on a real world network, epinions.com. For the PSN module, one of the challenges is to select the length of the trust paths such that execution time doesn't become unreasonably large while at the same time maintaining sufficient coverage of the data. With respect to this challenge, from the experimental evaluation with the epinions.com dataset the authors find that their PSN module performs well compared to a naïve approach that would select all paths, sacrificing a small amount of coverage in exchange for a drastic speeding up of execution time. With respect to accuracy of estimating trust, the authors find high accuracy regardless of which of several aggregation methods are used.

In [3.23] the authors propose adding important nuances to the consideration of trust calculations in online social networks, namely the inclusion of social relationships, recommendation roles, and preference similarity. Additionally, the authors propose a way

to consider evaluation criteria, i.e. a mechanism for nodes receiving trust scores from a system they interact with to decide how much (if any) faith to invest in these scores. In this sense, the authors' model is a more realistic model of how people interact and build trust based on their interactions. The authors also propose a more realistic trust decay mechanism that is based on social psychology research, which is that trust decays relatively slowly in early hops from the source to the sink of the trust, but then decays rapidly after the early hops. To account for these additional nuances described in the social psychology literature, the authors propose a mechanism for measuring the quality of transmission of trust, which they call Quality of Trust Transitivity (QoTT). Using this concept, the authors then propose a new model for trust estimation called Multiple QoTT Constrained Trust Transitivity (MQCTT). In an attempt to better account for theories of trust from the social psychology literature, the authors give their own definition of trust as "trust is the belief of one in another, based on their interactions, in the extent to which the future action to be performed by the latter will lead to an expected outcome." The authors give this definition to account for the importance of context in trust transitivity: A may trust B in Context 1, but in Context 2 A does not trust B. The authors propose a number of measures that are then taken as inputs into their QoTT measure. These include social intimacy degree, role impact factor, and preference similarity. Social intimacy degree measures how close (socially- and/or psychologically-speaking) one person is to another, with the implication being that a higher social intimacy degree will translate to a stronger trust transitivity. Role impact factor takes account of if the recommending node is a domain expert or not. And, preference similarity makes use of the social psychology finding that people trust others

more who are more similar to themselves. These measures cannot be feasibly calculated in all contexts, but they are feasible in many of the most frequently-used online social networks and have only become more feasible since the paper was first published. The authors also describe three distinct phases of trust decay: an initial slow decay phase in which trust decays slowly in the first three hops from the source to the sink, a fast decay phase in which trust decays much more quickly in hops four through six, and finally another slow decay phase in which trust decays asymptotically to zero beyond six hops from the source to the sink. Utilizing these properties of trust and the principles which influence its transmission, the authors define their measure for QoTT. Using QoTT, the authors develop their trust model MQCTT and apply it to a real online social network, the email network from the disgraced US company Enron. The authors compare their model to existing models in the literature at the time, each of which represented a multiplication-based strategy, and averaging-based strategy, and a confidence-based strategy. The authors found that MQCTT more accurately models the real-world nuances of trust based on the social psychology literature.

In [3.4], the authors provide empirical evidence that online trust networks do indeed exhibit characteristics of small worldness. Up until the time of this paper's publication, many of the computational trust metrics that had been developed for use online implicitly assumed that the networks that the metrics would be applied to exhibited small worldness, but there had not been empirical verification of this assumption using real-world data. The authors find that many online networks (as long as they are sufficiently dense) do exhibit

small worldness behavior. Small worldness is characterized by high mean clustering coefficients and low mean path lengths. Small world networks also tend to demonstrate scale free properties, verified through the presence of power law distributions, and this is the attribute that the authors use to verify the presence of small worldness in online trust networks. Because users can generally join and leave online networks at will, the nature of irregular growth of online trust networks leads them to behave as complex networks. The authors rely on scale freeness because it considers the networks from a dynamic perspective rather than a static one (which the centrality and path length measures do) – a critical attribute of trust networks being that they are dynamic. The authors utilize five real online social networks to explore their research questions: Advogato, Epinions, Kaitiaki, Robots, and Squeakfoundation. The networks range in size from 64 nodes (Kaitiaki) to 49,288 nodes (Epinions). Upon analysis, the authors find that the networks do exhibit scale freeness, to varying extents, verifying a critical and heretofore previously untested assumption of many computational trust metrics. Another challenge with many computational trust metrics is how far to propagate trust within the network (referred to as maximum trust propagation distance, or MTPD). Through their experimental results, the authors also verify that because of the small worldness of online trust networks, it is only necessary to know the direct trust of each node and not how the trust propagates. Using this insight, the authors propose a computationally optimized trust metric that reduces computational complexity compared to previous trust metrics.

The authors of [3.24] propose a Digital Hash Table Trust Overlay Network (DHTON) to model a peer-to-peer (P2P) trust network and store its reputation scores, together with a distributed reputation system, which the authors dub DHTrust. The authors identify two weaknesses in other computational trust metrics at the time of publication: 1) the possibility that trust score managers (as in PowerTrust) may themselves be malicious, and 2) a potential for incongruency between the provision of services in P2P networks and the reputational scores that are reported. To address these issues, the authors propose their system, DHTrust. DHTrust begins with distributing trust locally to trust managers. This local trust distribution is based on the network structure, rather than randomly as in some other systems. In this way, DHTrust may more accurately reflect local trust values. Next, the authors' system implements a distributed hash table – dubbed DHTON in this paper – as a method to model the network's structure and to store and distribute local reputation information. DHTON is itself based on a Chord network. Finally, DHTrust modifies existing reputation score calculation methods by merging the two separate questions of service quality and trust scoring into one. The authors evaluate the performance of DHTrust using simulations with 4000 peer nodes – 20% of which are malicious – and an identifier length of 12 for the Chord network for implementing the distributed hash table. In the simulations, the authors compare DHTrust to two other established computational trust metrics: EigenTrust and PowerTrust. The authors report lower convergence overhead for DHTrust compared to the other two trust metrics, for varying network sizes, and improved quality of service for DHTrust compared to EigenTrust. It should be noted that this system is quite

similar to TrustGuard, which is not referenced by this paper but precedes this paper by seven years.

The authors of [3.25] identify weaknesses with respect to malicious actors in the well-known EigenTrust algorithm, and propose an improved version called EigenTrust++ which seeks to mitigate potential negative effects from malicious actors. The authors propose that a trust system should be resistant to attacks, thereby enabling good participants to maintain their good reputation, decreasing the reputation of malicious participants, and providing new participants with a mechanism for building their reputations over time. The authors identify three specific vulnerabilities in the EigenTrust model: its local trust rating, feedback credibility, and the use of a circle of friends. The authors propose EigenTrust++ to address these three weaknesses. To address the first weakness, EigenTrust++ updates the original transaction rating formula from EigenTrust, taking into account the total number of transactions between a pair of nodes. This modification makes the algorithm more robust to scenarios where there are many, or few, transactions between a pair of nodes (i.e., in EigenTrust a situation of 10 only positive interactions receives the same treatment as a situation of 1000 positive interactions with 990 negative interactions, even though the second scenario logically raises concerns based on the large number of negative interactions). To address the second weakness, EigenTrust++ instead uses a “feedback credibility-based trust metric”, relying on similarity to aggregate local trust. The authors posit that two good nodes will give similar feedback to a common set of peers they have interacted with, and two malicious nodes will do the

same, but a good node and a malicious node will give different feedback to a given set of peers they interact with. Using this insight, the authors use feedback similarity as a weight in the overall local trust value. This modification means that a high global trust value can only be achieved by obtaining high feedback credibility *and* high local trust from peer nodes. This mitigates the scenario in which, for example, a malicious node gives a low local trust score to a good node. To address the third weakness, EigenTrust++ updates the original trust propagation model from EigenTrust (which relies on a uniform distribution) to instead utilize a differential probabilistic propagation model. EigenTrust++ considers the proposition that trust should be more likely to propagate among a circle of friends, which would encourage trust among good nodes and cut off trust propagating to or from malicious nodes. In EigenTrust, this is not the case as propagation is treated uniformly. The authors show that the algorithmic complexity of EigenTrust++ is equivalent to that of EigenTrust, with computational complexity  $O(n^2)$ . After describing their proposed modifications to the original EigenTrust algorithm, the authors evaluate its performance and compare it to that of EigenTrust using simulations. The simulations rely on the same parameters as those used in the original EigenTrust paper. The authors report comparable performance of EigenTrust++ compared to EigenTrust for Threat Models A and B (independently malicious nodes, and chains of malicious collectives, respectively), and marked improved of EigenTrust++ compared to EigenTrust for Threat Models C and D (malicious collectives with camouflage, and malicious spies, respectively).



In [3.26] the authors propose a trust model for use in Web services (such as Web APIs) that is based on the insight that many online networks demonstrate small world properties (high mean clustering coefficients and short mean path lengths). The authors' system relies on three core concepts: 1) a modification to the subjective logic model to include a "forgetting time" of trust values, 2) a trust evaluation model based on small world network properties, and 3) a trust computation mechanism for the Web services. The authors evaluate the performance of their system using three different experiments implemented through simulation, and compare performance to that of two competing systems (RATEWeb and a context-based trust computing model). The simulation results show that the proposed system effectively and correctly identifies high-quality and low-quality service providers; that when quality of service is quantified and published, service providers compete to improve quality; and, compared to the two previously mentioned competing models, the authors' system performs better in terms of detection capability and stability.

[3.27] does not propose a new trust metric nor do they use any of the trust metrics proposed by other scholars, other than various centrality measures: closeness centrality, betweenness centrality, Eigenvector centrality, and PageRank. This paper applies trust measures to two online social networks (epinions.com and Ciao) and characterizes the networks' structure and dynamics in light of trust. The paper poses three research questions, which the authors seek to answer through analysis of two different datasets from online social communities (epinions.com, and Ciao): 1) what are the properties of

trust networks?, 2) what is the relation between trust and network topology?, and 3) what are the dynamics of trust formation? With respect to the first research question, both the epinions.com and the Ciao networks exhibited classic characteristics of complex networks: degree distribution strongly different from the Poisson distribution, small-worldness, and high clustering coefficients. With respect to the authors' second research question, they find that Eigenvector centrality and PageRank centrality provide the best measures for computing trust because they are equally applicable to networks with simple structures and those with non-simple paths. In addressing the third research question, the authors report an interesting finding: as trust networks grow over time, the mean path length decreases, which is consistent with network science laws related to densification and shrinking diameters. At the same time, clustering coefficient and reciprocity grow over time.

[3.28] considers how best to rank nodes in networks that are growing dynamically. Although the authors don't specifically consider trust networks, this paper is useful for trust metric scholars because many trust metrics rely on centrality feedback-based metrics, which PageRank is one example of. One of the most popular ranking algorithms, PageRank, is generally a useful tool but one which faces shortcomings because of its static nature. PageRank was originally designed for ranking web pages, but has grown beyond its original scope to be used in ranking scholarly papers, traffic flow, and images, among others. These applications of PageRank generally ignore temporal aspects of networks as they grow and evolve over time, and the authors seek to understand how temporal patterns affect the

usefulness of PageRank. The authors develop simulations to understand the performance of PageRank when temporal aspects are considered, and compare PageRank to a simple in-degree measure. They base their simulations on a Relevance Model (RM), in which a node's relevance decays over time. Using the RM with PageRank, in cases where there is slow decay or no decay of relevance over time, recent nodes are weighted lower than older nodes with more connections, which leads PageRank to be biased towards older nodes. On the other hand, when using the RM with PageRank in a fast decay scenario, old nodes quickly lose relevance, which means that as new nodes attach to the network they are more likely to attach to other new nodes, leading PageRank to be biased towards more recent nodes in the case of fast decay. The authors run a different simulation in which fitness is considered, the Extended Fitness Model (EFM). Fitness is analogous to "quality", in the sense that in PageRank nodes which have more links with other high-quality nodes are considered to be higher-quality themselves. When using EFM, the authors find that PageRank outperforms simple out-degree measures in cases where relevance decay and activity decay are approximately balanced with one another. In cases where relevance decay is much greater than activity decay, or the reverse (activity decay  $\gg$  relevance decay), however, the authors again find that PageRank underperforms simple in-degree.

After performing the simulations, the authors next turn their attention to two real networks to understand how consideration of temporal effects influences the performance of PageRank: Digg.com, and the citations network between American Physical Society (APS) papers. To estimate the accuracy of PageRank, the authors use a measure of total

relevance. For the Digg network, the authors find that in-degree outperforms PageRank for total relevance, but only slightly. For the APS citations network, the authors find that PageRank is outperformed by the in-degree measure, with in-degree having a much higher correlation with total relevance than does PageRank. The authors conclude that PageRank should be used only with careful consideration for how temporal effects influence a given network, reiterating that PageRank will overestimate the importance of recent nodes in cases where decay of relevance is faster than decay of activity, and PageRank will overestimate importance of old nodes in cases where decay or activity is faster than that of relevance.

An open source trust management framework called Trust Graph was launched in 2015 and is targeted primarily for blockchain-based applications. Trust Graph describes itself as *"an open protocol for sourcing & rendering Trust relationships; It is a toolkit for building and reading distributed Trust Graphs; An ambitious plan to create interoperability between existing and future Trust Networks; Compatible with existing rating schemes (scores, percentages, star ratings, etc); [and] Open Source (Apache licensed)."*

In a 2015 white paper laying out the vision for Trust Graph [3.29] (called Trust Exchange at the time), one of the co-founders of the project laid out a vision for a decentralized reputation system with the following properties: *"Interoperable; Interconnected; Compatible with permissionless and anonymous accounts; Contextual; Chronological; Resilient (censorship resistant, immutable, decentralized); Re-interpretable (raw*

*data is open to user and use specific re-interpretation); Not feared; Compatible with the spectrum of identity" [3.30].*

The core building block of the Trust Graph system is what the creators term "trust atoms". Trust atoms encode reputation data in its raw form, and are signed with the private key of the person making the evaluation. Included in each trust atom are fields on the source of the evaluation (who is making the evaluation?), the target of the evaluation (who is being evaluated?), a trust value in the range of 0-1, content describing the rating, context, and a timestamp. Each trust atom is then cryptographically hashed using the SHA2-256 cryptographic hash, and signed with the source's private key. Upon signing of a trust atom, a signed JSON in the JSON-LD Verifiable Claim format is created. Trust Graph also allows for interoperability using RDF or IETF Reputons. In addition to the trust atoms, the current reference implementation of Trust Graph utilizes IPFS (Interplanetary File System) to store and share the trust atom hash files [3.31].

In [3.32] the authors propose the Personalized Hitting Time (PHT) trust mechanism. PHT is an extension of GHT (general hitting time), which adds additional resistance to Sybil attacks. GHT, which itself is similar to the PageRank algorithm, reduces the vulnerability of PageRank to being manipulated by the reports of the node being evaluated. However, the authors show the GHT is still vulnerable to Sybil attacks. The authors propose their system based on hitting time, and show that it is an exact algorithm optimized for trust in the presence of Sybil attacks by strategic agents (agents who attempt to surreptitiously improve their reputation in the system). The authors' exact algorithm is  $O(n^4)$ , which they

recognize to be prohibitive for practical applications. Thus, the authors also propose a Monte Carlo-based approximation method which they show to be much faster than the exact method. The authors evaluate the performance of PHT using a series of simulations. They demonstrate that in the presence of Sybil attacks, PHT results in no ability by the malicious nodes to alter their ranking, whereas in generic PageRank the Sybil attack drastically increases the strategic agents' ranks, personalized PageRank is shown to be somewhat resistant but still vulnerable to Sybil attacks, and GHT is shown to have essentially no resistance to Sybil attacks. The authors also demonstrate that in both dense graphs and sparse graphs, PHT continues to return a high degree of informativeness as the proportion of strategic agents in the network grows, whereas GHT, PageRank, and personalized PageRank all degrade as the percentage of strategic agents grows.

[3.33] extends the work of computational trust metrics to use for WhatsApp, which is an online messaging system and social network. Until the publication of this paper, most computational trust metrics did not consider the WhatsApp use case, making this one of the paper's primary contributions. Because of WhatsApp's ubiquity in messaging (particularly outside of the United States), it has attracted malicious actors who seek to use the platform for spreading malware with the aim of committing cybercrime, spreading misinformation, and other malicious intentions. The authors give one example of a malware attack spread via WhatsApp in which users receive a message with an interesting-sounding headline. The user clicks on the associated URL, compromising their account and causing their account to automatically send messages to their contacts asking them to

send money. The authors propose their system as a trust score that would accompany the receipt of any message on WhatsApp, helping the user to decide if they should trust the message – or even if they should open it in the first place. The authors propose WhatsTrust as their system for managing trust in WhatsApp, or any similar online messaging-based social network. WhatsTrust is based on Subjective Logic (SL), utilizing it to calculate local trust values between pairs of nodes. The WhatsTrust system is composed of two primary components, each of which is comprised of several smaller parts: the system component, and the node component. The system component is a centralized component that maintains two global lists on reputable users and uncertain users. The authors explain how positive and negative ratings are earned by nodes (a positive rating may be achieved, for example, when a recipient of a message adds the sender of the message to her list of contacts; a negative rating may be achieved when the recipient of a message blocks the sender of a message). The node component contains two parts: a trust calculator & rating provider, and a local trust list. The rating provider decides where to query trust information from (locally, from friends, or globally), and then the trust calculator uses the trust information provided to it by the rating provider to compute a trust value. The use of a local trust list (for certain calculations) enables faster and more efficient calculations. After describing the proposed WhatsTrust system, the authors evaluate its performance through simulation of several different scenarios emulating WhatsApp interactions. They compare the performance of WhatsTrust with that of EigenTrust and TNA-SL. Simulations were conducted with varying network sizes to understand the system's scalability, and different proportions of malicious nodes were considered to understand the system's robustness. In

54 different simulation scenarios representing different combinations of collusiveness, maliciousness, and network sizes, WhatsTrust outperforms EigenTrust and TNA-SL in 46 of the scenarios. The authors also compare execution times for the different trust systems, and find that WhatsTrust is much faster than TNA-SL. WhatsTrust's execution time is slower than that of EigenTrust, but in several of the scenarios it is only marginally slower. The long execution times of TNA-SL owe to its chained matrix multiplication operations.

[3.14] proposes a new trust metric – CoTrRank – and applies it to rank users on Twitter. The paper is unique because it uses two networks to evaluate trust, rather than only one. It does so by considering trust based on a user/tweet basis, rather than user/user, as others have done previously; thus, the way the authors model the graph it can be thought of as a bipartite graph in which two different classes of objects connect to each other. The authors also propose a method for statistically mapping actions of Twitter users to an appropriate trust degree. The authors collected tweets from Australia and tested their model on it. The collected data is divided into a user network, which captures “follow” relationships among users, and a tweet network, which captures the retweet and reply relations among tweets. The two networks are joined by the “mention” and “post” relations that take place between users and tweets. In the authors' method, they first normalize the trust values of both users and tweets. Normalization poses challenges that have been discussed in several other papers included in this review, namely, that it doesn't properly account for differences in nodes with many interactions and nodes with few interactions, and it also leads to domination by nodes with high trust values. To account for



these problems, the next step in the authors' method – which is one of its primary novel contributions – is to map trust values to trust degrees, using the sigmoid function. To employ the sigmoid function, users are ranked based on their trust values, previously calculated. Users with a trust value below a chosen threshold are excluded. The remaining users who were not excluded based on the threshold are divided into three different groups based on their ranking. The parameters of the sigmoid function can then be calculated. The authors apply their method to the real Twitter data they collected, and compare its performance to seven other established Trust Metrics. The authors present a sample of specific users to illustrate the effectiveness of their metric compared to the other, existing trust metrics. For the selected users, the CoTrRank metric ranks both of the users higher than the other metrics do, which they go on to demonstrate makes logical sense based on the trust scores of the followers of the first set of users. One of the primary causes for the other metrics' lower ranking of these users is because they don't consider all available information that a platform like Twitter provides.

In [3.34] the authors propose a new trust management systems inspired by ant colonies, for use in peer-to-peer (P2P) networks of sensors or devices. The authors' proposed system, AntTrust, differs from other bioinspired trust management systems in that it is a problem-specific system, as opposed to metaheuristics, which are more computationally complex and expensive. The problem that AntTrust addresses is how to locate a trustworthy node by imitating how ants solve similar problems in real ant colonies. AntTrust relies on four main categories of trust factors: 1) trust values, 2)

recommendations, 3) feedback from other nodes, and 4) global trust values. AntTrust is inspired by the behavior of foraging ants. When ants leave the nest in search of food, they leave a trail of pheromones on their return journey if they have successfully found food, which allows their fellow ants to find the same food source. In AntTrust, a file requester is analogous to the ants' nest, a request for a file is analogous to an individual ant, and neighboring nodes are analogous to a food source. When a requesting node successfully receives a file from another node, it provides a positive rating, which is analogous to the ants leaving a pheromone trail. The AntTrust system's architecture is composed of a rating manager, a trust manager, a feedback manager, a recommendation manager, and a transaction manager. The transaction manager identifies the most trustworthy file providers for peers, considering the four trust factors mentioned previously. The rating manager submits positive or negative ratings for each transaction, based on the validity of files as determined by the transaction manager. The trust manager calculates new trust values after each transaction, and the value is a function of the validity/invalidity of the file transferred. The feedback manager shares ratings and feedbacks among nodes, and the recommendation manager retrieves recommendations about specific nodes. The AntTrust algorithm initially selects a file provider randomly, when the network is being initialized and there are not yet any transaction histories upon which to rely. After a transaction completes, the file receiver rates the file provider, and a reward (in case of a valid transaction) or punishment (in case of an invalid transaction) is applied to the file provider. The file provider selection algorithm (a separate algorithm) identifies trustworthy nodes based on the four factors described previously. Neither the AntTrust nor the file provider

selection algorithms contain any nested loops, leading the time complexity of AntTrust to be  $O(n)$ . After describing the AntTrust system, the authors evaluate its performance using simulations and compare performance to that of the other popular computational trust metrics EigenTrust and TNA-SL, and competing bioinspired trust metric TACS (Trust Ant Colony System). The simulations considered the success rate of the algorithm and its execution time. AntTrust's success rate was considerably better than that of the three competing algorithms in all simulation scenarios, with the difference in performance growing larger as the number of malicious peers in the simulated network increased. With respect to execution time, AntTrust was comparable to but took slightly longer to complete than EigenTrust, and AntTrust had a much faster execution time than TNA-SL or TACS.

#### *Content-based and behavior-based approaches*

[3.35] proposes a new trust model they call Arbitration Trust or @Trust, based on feedback arbitration. @Trust aims to mitigate shortcomings of other trust models that had been published up until the time of this paper, including fragility to attack or manipulation by malicious peers, high computational overhead, inaccurate trust estimations because of removing false positives, and other issues. @Trust relies on three main principles: 1) arbitration nodes are created and assigned, which arbitrate evaluation feedback of nodes and make decisions based on majority rule, thereby identifying false feedback, 2) all nodes receive service credibility, feedback credibility, and arbitration credibility scores, each of which play different roles in decision making and punishment, and 3) arbitration nodes make rapid decisions on behalf of other nodes, improving the speed of transactions in the

network. Whenever a transaction occurs, nodes generate an evaluation of the quality of the transaction and send these evaluations to the arbitration nodes. The arbitration nodes then make a decision with respect to this feedback based on a weighted majority rule. Arbitrations are also rated by management peers. Credibility values for service, feedback and arbitration are also calculated, which feed into this system's trust decision making mechanism. The authors make a subtle but important distinction in their system compared to others, which is that other systems generally only consider general credibility of an agent when deciding whether that agent should provide a service or not. The authors' system, on the other hand, considers credibility for each service offered by an agent, rather than the agent's general credibility, which allows for finer-grained decisions with respect to trust. The authors analyze the system's time complexity, breaking it into two constituent parts: communication complexity and computational complexity. Communication complexity relates to sending and receiving of messages about states of different nodes within the network. The authors describe conceptually how their system would be used in a structured P2P network using the Chord network, and then they evaluate their system's performance using simulations and comparing against performance of other systems, namely RBTrust and OFTrust. The key dimensions that the authors are interested in evaluating in the simulations are the system's ability to arbitrate evaluations, ability to resist attacks or manipulations, and its ability to punish malicious behavior. The authors find markedly improved success rate (the rate of downloads that are successfully completed in the P2P network) compared to the two other systems, an increase in the upload rate (which is taken as a proxy for reducing freeriding behavior (or, said another

way, punishing freeriding behavior) compared to the two other systems, and an improvement in resistance to two types of attacks (whitewashing attack and collusion attack).

[3.36] points out that most existing trust management systems rely on – and assume availability of – data related to behavior of agents within a system. This, however, is not always a given, as for many of the systems for which trust management systems are proposed certain data is available only to a proprietary provider and not to scholars. In their system, the authors propose a more generic approach to estimating trust that relies on network monitoring techniques adapted to this context, analyzing (Internet protocol) packets exchanged among agents in a system. The authors propose a formal model which relies on representing a packet along with the formal language that accompanies doing so, syntactical properties, and semantic properties. In addition to their formal model, the authors also develop a suite of network monitoring and testing tools that use the formal model to automatically evaluate trust among agents within the network. These include a tool for forwarding filtered network traffic to a remote server and a tool that gathers and tests packets, using the formal model. The second tool makes use of two algorithms which implement the formal model: the first algorithm is for evaluating the packets, and the second algorithm provides a timeout function. The authors evaluate the performance of their model using simulations of industrial DNS (Domain Name Service) network interactions, which involves clients and servers. Importantly, because the authors are not proposing a new method for evaluating trust but rather an approach to utilizing generic

data when proprietary data may not be accessible, the results they report from their simulation focus on the resource performance of the system – not on the accuracy in measuring trust. They find a modest increase in resource consumption (CPU and RAM usage, and network traffic).

In [3.37] the authors use a semi-supervised learning approach to rank how trustworthy or untrustworthy different news sites are. Their novel approach lies in the use of only the words in a news article's URL, rather than the text of the article. This is a useful approach because it could be integrated directly by search engines, and it is fast. Using natural language processing (NLP), they calculate a similarity score between two different pages, and build a similarity graph using this. They also remove (prune) all pages from their graph that don't meet a minimum threshold. From the similarity graph, they apply a biased PageRank, in which the bias is increasing weight to known fake news websites. They then build a ranking of distrust of news sites. The authors apply their method to a real dataset of news websites representing both reliable and unreliable sources of news, and compare the performance of their method that of a standard support vector machine (SVM). The authors find a significant improvement in performance compared to the SVM, as measured by precision, recall, and F1 scores.

### Probabilistic approaches

Many trust metrics up until the time of [3.38]'s publication modeled trust in a simplistic way. Doing so enables computation of the trust metric, but it sacrifices richness of information since trust is multifaceted, depending on many characteristics between

agents and changes depending on the context of where the trust is being applied. Thus, the authors propose a Bayesian-based trust model which takes into account different characteristics of agents within a trust network, allowing for greater flexibility and nuance. The authors' metric was designed for use in P2P file-sharing networks, but it could potentially be applied in other settings too. The authors begin constructing their model by defining four different broad categories of where trust relationships may be found: 1) trust between a user and her agent, 2) trust in service providers, 3) trust in references, and 4) trust in groups. Additionally, the authors differentiate between trust in another agent's competence, and trust in another agent's reliability (which, in this paper, considers both truthfulness and trustworthiness). The authors develop a framework that utilizes a naïve Bayesian network representing trust between an agent and a file provider in a P2P file-sharing network. The authors' model considers leaf nodes under each root node, in which leaf nodes represent the root nodes' capabilities in different contexts (in this paper the contexts are sharing of music, video, document, image, and software files, but these could vary depending on the context and application). For each leaf node, conditional probabilities table (CPT) is constructed, which are computed based on satisfaction derived from a given file sharing transaction, whether the file shared was legitimate (e.g., was a file that was shared and said to be a music file really a music file?). Download speed and file quality are also considered in this model, but these are parameters that can be modified for the specific application at hand. The authors then apply their model to a simulated P2P file-sharing network to test its effectiveness. They find that, when combining both trust and reputation systems, their Bayesian-based model exhibits marked improvement in

performance (as measured by the proportion of successful recommendations from one agent to another) over similar non-Bayesian models (ranging from approx. 5-10% performance improvement, depending on the number of interactions that take place). One of the limitations of this model is that its usefulness is highest in networks in which agents have multiple, repeated interactions with one another. In networks where agents have few and/or infrequent interactions (the authors mention buyers in e-commerce as one example, in which a buyer may only ever interact with a given seller once), this method is less effective. However, even in these types of contexts, if small worldness characteristics are present in the network, the model can still be useful.

Building on the previous work of the TidalTrust algorithm, [3.39] proposes a new algorithm for measuring trust in online social networks, called SUNNY. One of the primary contributions of the SUNNY algorithm is to include not only an estimate of the trust value among nodes in a network, but also to incorporate confidence levels of these trust values. In SUNNY, confidence is modeled from -1 to 1, with a -1 corresponding to a state of “disbelief” and a 1 corresponding to a state of “belief”. The authors state that a confidence model (which is a conditional probability of belief and disbelief) can be generated as inputs from domain experts, or by using approximation techniques such as similarity or sampling. SUNNY is a probabilistic, stochastic, local trust metric that is personalized to each node, and is designed for use in online social networks although it may be useful in other contexts, too. Experimental results that applied the SUNNY algorithm to the same network as TidalTrust (the FilmTrust network) found a noticeable improvement in the algorithm’s



performance in estimating trust values compared to the ground truth values. SUNNY generates a Bayesian network, then creates estimates of lower and upper bounds on confidence values of each leaf node with reference to the sink node. SUNNY then makes a decision regarding each leaf node as to whether to include it in the final trust computation. The result of these decisions can be either include (in which the leaf node is included in the trust calculation with respect to the sink node), exclude (the leaf node is not included in the trust calculation), or unknown (this is an intermediate state in which the algorithm has not yet made an include/exclude decision). Once a decision has been made with respect to each leaf node as to whether it will be included or excluded in the trust calculation, SUNNY then performs a backward search from each leaf node to the sink node to compute a final trust value.

[3.40] is a broadly influential paper in the area of trust metrics. The trust network analysis with subjective logic (TNA-SL) models trust relationships as subjective opinions, and simplifies complex graphs into series and parallel graphs to enable and simplify analysis. Like other trust metrics, TNA-SL makes use of the concept of transitive trust in which trust is transmitted from one node to another that isn't directly connected. The authors also add an important nuance related to trust: trust scope. Trust scope considers the context in which trust is being considered, and only applies it within that context (not universally). The authors introduce a framework for considering trust in more granularity: belief, disbelief, and uncertainty ( $b$ ,  $d$  and  $u$  respectively), where  $b + d + u = 1$ . The authors discuss operators for trust network analysis, namely, transitivity and fusion. Transitivity

involves a discounting function, that is, the farther one moves from an original node, the more the transitive trust imbued by that original node is discounted. Fusion or consensus involves combining opinions – especially when the opinions conflict with one another – in a fair and equal way. This is also known as Bayesian updating. Finally, the principal contribution of this paper is the proposal of edge splitting for trust analysis. This method splits dependent trust path edges into multiple edges to avoid path dependency, while also preserving information in the graph.

In [3.41] the authors propose a new trust metric which they call pervasive trust. The authors make use of two key axioms related to trust for their paper: *"1. We will treat trust simply as a probability that a given assessment about an agent is true or false (e.g. fair/reliable or not); 2. We further assume that this belief is transitive, i.e. if agent a trust agent b, which in turn trusts agent c, then a will also trust c, to some extent."* Pervasive trust makes use of the concept of transitive trust, used for estimating trust transmission in a network. The authors describe their approach as being *"derived directly from the simple notion of trust transitivity, is easy to interpret, propagates absolute values of trust, and makes no assumption whatsoever about the network topology, and direct trust distribution."* The authors define pervasive trust and how to measure it. Pervasive trust infers trust values for *pairs* of nodes (rather than globally, as in EigenTrust and others) and then computes transitive trust based on paths between source and target nodes. Rather than using *all* possible paths in calculating transitive trust, though, pervasive trust considers only those with the largest weights (i.e., the greatest trust) to the in-neighbors of the target node. After proposing their trust metric,

the authors apply it to a real example – PGP (pretty good privacy) – and analyze its performance. PGP is a schema for encrypting and decrypting electronic communications, and makes use of a “web of trust”. A web of trust is a decentralized trust model used to assert authenticity of public keys and owners of those keys (i.e., the bindings of keys) whereby the nodes themselves that a key-node binding is legitimate. The web of trust stands in contrast to a centralized trust model, like certificate authorities, which are used in traditional public key infrastructure (PKI) schemas. The authors use publicly-accessible data on PGP transactions; additionally, to test the robustness of their trust metric, they use the same PGP data but randomly reassign edges within the network. After applying their metric of pervasive trust in an authority-focused model (similar to the PKI/certificate authorities model) they find higher overall trust at a network level and more inter-connectivity among different communities within the network. In this model, highly-connected nodes (generally the certificate authorities) become more trusted. By contrast, when applying their trust metric to a community-focused schema (like PGP), they find overall lower trust in the network, and more focus on intra-community connectivity. The nodes with the smallest and the largest in-degrees receive the lowest trust values, while those with intermediate connectivity exhibit higher trust values. The real PGP data in fact show a mix of both authority-focused and community-focused models present simultaneously within the PGP network.

### Fuzzy Logic approaches

[3.42] proposes a trust management system – FuzzyTrust – based on fuzzy logic inferences. The authors' system represents trust estimations linguistically, and includes a module for computing and storing a source node's trust score, the same for a target node's trust score, and a global reputation aggregation weighting module. The authors compare the performance of their system to that of EigenTrust, using a real world dataset from eBay. The FuzzyTrust system demonstrates lower computing overhead and slightly improved detection rate of malicious peers as compared to EigenTrust.

In [3.43] the authors identify several shortcomings with other proposed trust management systems that had been published up until that time. These shortcomings include loss of information when aggregating trust paths in graph-based models like TidalTrust, the inherently difficult nature of trying to model a subjective human concept like trust using quantified values (typically in the range from 0 to 1). The authors propose a fuzzy logic-based system, which is an extension and improvement of the previously-proposed TidalTrust algorithm. The TidalTrust algorithm relies on calculating trust based on the strongest shortest path between two nodes; that is to say, of all of the possible paths between **A** and **F**, which of these paths is a shortest path and also imputes the highest level of transitive trust between **A** and **F**. The authors of this paper propose a modification to the strongest shortest path method, instead opting for an "all lengths strongest path" method – that is, considering which path between **A** and **F** provides the highest trust level, regardless of if that path is long. While this method increases

computational complexity, it provides a more precise inference of the trust levels between nodes, as it can, for example, account for complex paths between two nodes which may pass more than once through a neighboring node. Additionally, the authors' system utilizes a linguistic approach to measuring trust, rather than an explicitly quantified one, using low, medium-low, medium, medium-high, and high levels of trust instead. To evaluate the performance of their system, the authors construct simulated trust networks and compare their fuzzy method with mean aggregation methods and subjective logic aggregation methods. They find marked improvement in estimating reliability of nodes compared to TidalTrust.

[3.12] focuses on the specific application of trust management systems to recommender systems. The authors propose a fuzzy logic-based method which exploit both trust and distrust to make better recommendations for users. The authors raise several issues with the way that other systems model trust. One issue is the crisp modeling of trust, which makes computation easier but which can lead to information loss about the trust relationships. In real online networks, users generally will express trust in other users (if they are using a system which has such functionality) linguistically (e.g., low, medium, high) rather than numerically (most trust systems use a range from 0 to 1). Finally the authors also point out that in real-world recommendation systems data tends to be sparse, and other works don't consider how to propagate and aggregate trust for inferring indirect trust in cases of network/data sparsity. The authors propose fuzzy computational models for estimating trust and distrust in online settings. Five different models are proposed with

different ends in mind: 1) trust and distrust based on similarity, 2) trust and distrust based on knowledge, 3) fuzzy trust and distrust modeling, 4) fuzzy trust-distrust propagation of transitive trust, and 5) a fuzzy approach to aggregation of transitive trust. The similarity-based trust/distrust model treats users' interest in items considered by the recommender system as fuzzy variables. The knowledge-based model utilizes ratings that users give of one another after completing transactions, representing users' interactions in terms of satisfaction, reciprocity, and reliability. The third model translates crisp trust evaluations (in the range of 0 to 1) into seven sets of fuzzy evaluations ranging from no trust to complete trust. The authors evaluate the performance of their models using a real world dataset, the MovieLens dataset which includes 100,000 ratings of movies, with each user having rated at least 20 different movies. They report superior performance of their methods when incorporating fuzzy computational models of trust and distrust into recommender systems, improving their overall performance and also outperforming other trust/distrust-based models.

#### Game Theoretic approaches

[3.44] proposes an incentive mechanism to encourage sharing and cooperation in wireless sensor networks, based on Bayesian games. The authors observe that some trust management systems ignore the actions of malicious nodes, effectively treating them as completely malicious when in reality malicious nodes will frequently act with some level of rationality wherein they sometimes take positive actions to appear better than they are. Additionally, the authors observe that for good (non-malicious nodes) there is the frequent

problem of freeriding, where good nodes don't cooperate even with other good nodes. The authors' incentive mechanism both encourages limited cooperation with malicious nodes in cases where they are not acting maliciously, and also encourages greater cooperation among good nodes. The authors evaluate the performance of their mechanism using simulation, and find that their system delivers improvements on both dimensions previously described.

In [3.45] the authors propose a game theory-based model for measuring trust in social networks. The authors consider two types of trust: specific trust (trust between two actors within a network), and global trust (overall levels of trust within the network as a whole). The authors' model relies on three main factors: interaction history, recommendation, and user behavior. In addition to this model, the authors also propose a punishment mechanism to reduce prevalence of the free-rider problem. The authors posit that existing trust mechanisms in online platforms (which can be considered social networks, in certain respects) such as Amazon and eBay rely on global trust, but this method can be vulnerable to false feedback or recommendations. The authors describe the free-rider problem – which is observed when participants (nodes) in a system of exchange seek to benefit from the system without contributing back to it – and how it affects networks generally and trust in networks specifically. The authors classify nodes into four different types: service, feedback, recommendation, and managed nodes. In the authors' system, service reliability refers to the trustworthiness of an agent in the system to effectively deliver the service in question, feedback effectiveness refers to the

trustworthiness of feedback given (about other nodes) by feedback nodes, and recommendation credibility refers to the trustworthiness of a given recommendation. The authors' system includes two algorithms, one for measuring these credibility values from direct interactions, and another for inferring these values in the case of indirect interactions. Critically, the authors' system also provides three different punishment strategies, used to punish free-rider nodes and encourage them to stop their free-riding behavior. The authors estimate their system's performance using simulations, with networks of 1000 nodes where 30% are trustworthy nodes, 30% are mixed-trustworthiness (sometimes trustworthy, sometimes not), 10% are fully malicious nodes, 20% random malicious nodes, and 10% veiled malicious nodes. Performing their simulations to examine the effectiveness of the different punishment strategies, the authors find a clear improvement in quality of service in the networks when including game theory-based punishment strategies to reduce the prevalence of free-riding.

In [3.46] the authors highlight shortcomings with other trust management systems for self-organizing wireless networks (this paper's focus). These include failure to filter selfish feedback and distrusted recommendations, and a focus on experimental data while lacking a theoretical foundation. The authors model both a single stage game and a repeated game, utilizing their proposed payoff matrix (punishment mechanism). The authors' system focuses on incentivizing sharing of indirect trust information within the network; that is, encouraging nodes to share the direct knowledge they have of other nodes based on their interaction, thereby improving the overall trust estimation of the



entire network. Based on simulation results, the authors' system efficiently promotes cooperation in the case of sharing of indirect information in both single stage and repeated games.

*Hybrid approaches (using two or more of the previously-identified approaches together)*

In [3.47] the authors propose a semi-automated method – TrustRank – for filtering spam Web pages (Web pages which seek to artificially increase their ranking from search engine algorithms, like PageRank) that relies on human experts manually scoring Web pages, then using these manually scored pages as seed together with the link structure of the Web can discover good (non-spam) Web pages. The authors' system starts with a simple but important realization, which is that good pages rarely point to bad pages, while bad pages may point to good pages, and point to many bad pages (termed approximate isolation by the authors). TrustRank utilizes a trust function, a trust propagation mechanism, and trust attenuation (also referred to as “trust decay” elsewhere in this dissertation). The TrustRank system takes as inputs a transition matrix, the number of Web pages in the network, a limit on the number of oracle invocations (i.e., the limit on the number of manually-classified pages), a decay factor, and the number of iterations. TrustRank first evaluates the desirability of pages to serve as seeds, it then rank orders the pages based on their desirability, selects the best (top) seeds, normalizes their scores, and then calculates the TrustRank scores for individual pages. The authors evaluate the performance of TrustRank using a real world dataset, the (now defunct) AltaVista Web index from 2003, which contained several billion pages representing more than 31 million

Web sites. The authors compare the performance of TrustRank to that of PageRank, and find that TrustRank presents more good pages to users in search results than does PageRank, and it presents fewer bad pages than does PageRank, concluding that TrustRank effectively filters out spam pages.

In [3.48] the authors extend the work of EigenTrust. EigenTrust addressed the problems of spread of inauthentic, corrupted, or malicious files in P2P networks, but a slightly different issue that isn't addressed directly by EigenTrust is the issue of incentivizing nodes to share files in the first place. The authors of this paper propose a "*strategyproof partition mechanism that provides incentives for peers to share files.*" The authors propose, in essence, identifying communities of like nodes within a broader P2P network, partitioning these communities into peer groups, and incentivizing downloads from *within* a given peer group. The authors' system starts with the assumption that all nodes – whether they are malicious nodes or not – are seeking to maximize their own utility function, whatever that means for them. For the purposes of the system, the utility of a node is its trust score. The system modifies EigenTrust to incorporate strong non-manipulability by cut partitioning nodes, though this comes at the expense of a decrease in the core effectiveness of the EigenTrust system. The authors discover and note a critical tradeoff in this type of design decision: if partitions are made in a way that leads to small numbers of nodes in the partitions, the trust values will be less meaningful and thus less useful to users, yet at the same time if there are more partitions then the error in trust values (compared to the original EigenTrust system) increases.

In [3.49] the authors point out vulnerabilities in trust management systems that had been proposed up until the time of publication (e.g., EigenTrust, XRep, others). Some of these vulnerabilities include shilling, in which malicious nodes submit dishonest feedback, or a malicious node initially exhibiting good behavior to establish a positive reputation, and then acting maliciously once the positive reputation is established. The authors propose TrustGuard as a system to combat these and other potential malicious attacks on trust management systems. At a high-level, the TrustGuard system consists of three components: a trust evaluation engine, a transaction manager, and a trust data storage service. The trust evaluation engine is used whenever a potential transaction is to occur between a pair of nodes, wherein a node seeks information from the network about its potential pair, aggregating this network information into a trust value. The transaction manager is responsible for detecting and preventing fake feedback; feedback that passes the transaction manager's check is sent to the data storage service, where trust values and feedback are securely stored. TrustGuard includes three primary mechanisms to reduce vulnerability of a trust management system to manipulation: 1) strategic oscillation guard, 2) fake transaction detection, and 3) dishonest feedback filter. The strategic oscillation guard attempts to prevent a node from alternating between honest, positive behavior and malicious behavior milking the positive reputation it has built up. The fake transaction guard attempts to identify illegitimate feedback (for example, a malicious node made attempt to provide negative feedback to a good node for transactions that never occurred), and the dishonest feedback filter attempts to distinguish honest feedback from dishonest feedback by malicious nodes. The dishonest feedback filter utilizes a personalized similarity

measure (PSM), in contrast with a trust-value based credibility measure (TVM), as others such as EigenTrust utilize. Another interesting element of note for TrustGuard is its “fading memory”, which allows for negative interactions with malicious nodes to lose weight as time goes on. The fading memory mechanism was included primarily as a technique to improve computational performance, keeping in mind that the paper was published in 2005; it may be unnecessary today, given current computing capabilities and cost, however, the concept of a fading memory does have support from social psychology researchers. After proposing and describing the TrustGuard system, the authors evaluate its performance with respect to the three categories of vulnerabilities (strategic oscillation, fake transactions, and dishonest feedback) using simulations, with 1024 nodes and a randomly selected proportion of those nodes being designated as malicious. The authors find that TrustGuard is resistant to strategic oscillation strategies for several different time intervals (that is, the time between malicious actions by malicious nodes), imposing a cost on malicious nodes for this type of behavior. The authors also find that their PSM-based dishonest feedback filter is much more robust than TVM-based ones both in cases of non-collusive malicious nodes and in cases of collusive malicious nodes.

In [3.50] the authors propose a new method for calculating direct trust relationships within an online social network, which then can be used for imputing indirect trust values among members of the network. The authors point out that trust is a crucial ingredient for ensuring quality of service and security in online social networks. The authors identify a gap in the computational trust metrics at the time, namely, that these metrics rely on a

direct trust calculation to manage trust but they don't specify how to calculate the trust values. The authors propose a new method for calculating direct trust as an input into the trust systems, relying on Interactions, Relationship types, and Interest Similarity (IRIS). The authors base their system on the taxonomy used in the FOAF (Friend of a Friend) ontology, part of semantic Web. Using the FOAF ontology allows for computation of direct trust values based on users' interactions, relationship, types, and interest similarity. The authors apply their method to real FOAF data, foafPub, which contains more than 200,000 records. Using this dataset, the authors calculate the performance of IRIS compared to the actual trust value (which is known from the dataset). They find a marked improvement in performance (as measured by F-score) compared to other methods including TidalTrust, and naïve measures such as max minus mean, max minus min, etc.

In [3.51] the authors identify several problems prevalent in "Web 2.0" platforms at the time of publication (which have only intensified since then), including "*digital deception, digital content redundancy, digital copyright confusion and malicious content diffusion.*" The authors mention that DRM (Digital Rights Management) technologies have seen limited success in addressing some of these problems, but DRM requires strict controls which are by definition at odds with the open and largely uncontrolled nature of online social networks. The authors propose a new system that relies on the strong ties and weak ties theories from sociology, wherein networks display characteristics of positive assortativity or homophily in the case of strong ties – that is, nodes that are similar to one another in some way tend to connect to one another – and negative assortativity in the case of weak

ties – that is, nodes with different characteristics tend to connect. In a real world context, weak ties may be seen more frequently among colleagues for example, while strong ties are more frequently found among friends or family members. The authors' system utilizes the strong ties and weak ties theories to partition a network into corresponding parts, namely, one where strong ties are predominant (representing approximately 20% of the overall network) and another where weak ties are dominating (representing the remaining approximately 80% of the overall network). After the network has been partitioned, the system relies on trust agents, who evaluate content and behavior of a given node and share this evaluation with a different node. Content is evaluated based on four parameters: 1) principle, 2) technicality, 3) freshness, and 4) artistry. The first two are considered objective measures while the second two are considered subjective, and thus the authors give more weight to the first two factors (without discarding the second two). Behavior evaluation factors also rely on four parameters: 1) discipline, 2) execution, 3) service, and 4) enthusiasm. Similar to content evaluation, the authors give greater weight to the first two objective factors and less weight to the second two subjective factors. The authors evaluate the performance of their system in a simulation and an emulation of a real world network. In the simulation, they find that as the number of weak ties in a network increases, the content evaluation becomes more important than behavior evaluation, while in networks with increasing numbers of strong ties, the reverse is observed (behavior evaluation is seen as more important than content evaluation). This makes logical sense, as it stands to reason that in cases where strong ties are more prevalent the nodes will regulate their own based on behavior of their neighbors (whom they know well), while in the case of a

prevalence of weak ties the nodes, not knowing their neighbors as well, need to rely more heavily on content to evaluate their neighbors' trustworthiness. For the emulation, the authors utilize the real-world topology of Facebook networks to test their system's performance. They find strong agreement between their model's predictions and the Facebook topology in indicating that an increase in trust placed in a user corresponds with an increase in both weak ties and strong ties to that user.

In [3.52] the authors propose a system for ranking influence of nodes in a social network using trust measurements. As such, this paper proposes a trust measurement system, but not a trust management system. The system is dubbed Trust-Oriented Social Influence evaluation, or TOSI. Measuring influence in an online social network can help to provide better recommendations to other users, and the authors point out shortcomings of other non-trust-based social influence estimation methods. The two primary issues addressed by TOSI are the issue of inaccurate influence recommendations because not all contexts of a node's potential influence are considered, and the issue of susceptibility to attacks by malicious nodes – in this case, primarily spammers. TOSI considers network structure, social trust, social relationship, and preference similarity when providing influence rankings. The authors use two real-world networks – epinions.com and DBLP – to evaluate the TOSI system's performance, and compare TOSI's performance to that of SoCap (the state of the art influence estimation system). The authors identify the top 1000 most influential nodes based on their actual number of influenced nodes, and use this as the ground truth dataset. They compare the ground truth dataset to TOSI and to SoCap

and find that TOSI significantly outperforms SoCap in how many of the ground truth top 1000 influential nodes it can identify. They also find that TOSI is quite robust to spam farm attacks while SoCap is unable to identify spam farm attackers, and finally that TOSI features much faster execution time than that of SoCap.

[3.13] proposes a method for estimating trust among software developers on GitHub. The authors take a content-graph hybrid approach for estimating trust values. The contribution of this paper is to provide an automated and scalable method for estimating trust among developers in OSS (open source software) projects, particularly when two developers have no previous interactions. The authors propose that their method could be used by developers to consider with whom to work on projects, whether they should accept or reject pull requests in GitHub, and as a check on human judgement of pull requests. The authors take a two-part approach for estimating trust. First, they compute direct trust between developer pairs with known historical interactions. The authors utilize natural language processing (NLP) applied to comments made related to pull requests, and classify each interaction as strongly positive, weakly positive, neutral, weakly negative, or strongly negative. With the direct trust scores thus established, the authors then use these to infer indirect trust by constructing a community-wide network and propagating trust. To do so, the authors rely on the subjective logic model, described elsewhere in this section. In the subjective belief model, each interaction between nodes takes on specific values of belief (b), disbelief (d), and uncertainty (u) which all sum to one. The authors map different combinations of b, d and u to states of trustworthy (high belief, low disbelief, low



uncertainty), untrustworthy (low belief, high disbelief, low uncertainty), and lack of trust (high uncertainty). Additionally, the subjective logic model utilizes transitivity and cumulative fusion (also discussed in greater depth elsewhere in this section). The authors apply their method to real data collected from GitHub, all using the Python programming language. The authors seek to address two research questions: 1) building an understanding of the relationship between trust and pull request acceptances, and 2) building a predictive model for pull request acceptance. To address these questions, the authors perform several different experiments to evaluate the overall performance of the model with respect to the research questions. The experiments vary the methods used for determining trust propagation, trust aggregation, and trust path length cutoff. The authors find that there is a statistically relevant relationship between a positive trust score of a developer and whether that developer's pull request will be accepted, but it is a small effect. For developers with low trust scores or lack of trust, there appears to be no relation (positive or negative) between their trust scores and whether or not their pull requests will be accepted.

In [3.53] the authors describe one of the major challenges of designing and applying trust metrics in online settings, namely, that of how to measure trust across multiple heterogeneous networks. To address this challenge, the authors propose a method that uses semantic technologies and data fusion to evaluate trust across different online networks. The authors' system incorporates semantic web technology to model heterogeneous networks in a standard format, then fuses these data into one set for further

analysis of trust. The system is composed of four primary modules: 1) a data acquisition module, 2) a coreference resolution module, 3) a trust data fusion module, and 4) a trust evaluation module. The data acquisition module extracts RDF (Resource Description Framework, a standard for data sharing on the Web) data between given users across heterogenous networks, focusing on information that will be useful in building links across the different networks. The coreference resolution module finds URIs (Uniform Resource Identifiers – part of the RDF and the key connection between RDF and the Web) of users across different networks, matches them, and assigns them a unique URI that is specific to this trust management system. The trust data fusion module aggregates the values of trust for nodes from the different networks. Because of the often complex nature of trust measurements in online contexts, and because of the fact that trust data may be sparse in one network and rich in another network for the same user, the authors recognize that simple additive or averaging data fusion techniques would be likely to lose, inflate, or deflate important data. For this reason, they instead select a weighted order weighted average (WOWA) method. Finally, the trust evaluation module – using data from the previously-described modules as input – can calculate trust among nodes by using a trust propagation algorithm. With their proposed system, the authors design a simulation and an experiment using real world data to evaluate the system's performance. The simulation portion of the evaluation is used to evaluate the performance of four different data fusion techniques in best reflecting the ground truth of trust values. In the simulations, pairs of networks are fused using the different methods. The authors find that the WOWA method performs best, regardless of both participant overlap (the percent of participants found in

both networks) and tie overlap (the percentage of ties between specific pairs of users that are observed in both networks). The authors use this finding from the simulation to further evaluate their system on real-world data, seeking to understand if the multiple networks approach to trust measurement produced more accurate results than use of a single network. The authors use coauthorship and collaboration frequency networks as their real-world dataset test. Twenty-six survey participants were recruited and divided into rating participants and rated participants. The survey participants were given questionnaires, and their responses to the questionnaires were used to establish a ground truth of trust between users to which to compare the trust management system's inferences. In the case of overlapping users (those present in both networks), the authors indeed find that their system is more effective at accurately measuring trust compared to when only one network is used for the same task. However, in the case of non-overlapping users (those found in only one or the other network), the authors' method resulted in worse performance compared to when only one network is used to measure trust.

The authors of [3.1] note that in modern online social networks, spread of low quality information or misinformation has become a prevalent problem. The authors propose their system, DeciTrustNet, as a trust and reputation management system to help combat these issues. Their system relies on double evaluation feedback (both the target and the sink of the evaluation are evaluated), inclusion of both global and specific trust for all users, leveraging the position of a user in the network to estimate their trust (similar to PageRank), and considering how the user's reputation has evolved over time (not just their

current reputation). The authors evaluate the performance of their system using simulation compared to performance of two competing systems, SocialTrust and PCR. The simulations show that DeciTrustNet generally performs as well as and in some cases better than the competing systems, and much better than when trust is not considered for varying proportions of malicious nodes. In another simulation, the authors consider the effect of nodes switching from malicious to non-malicious state and vice-versa. For this scenario, they find their system's performance remains robust until the point when about 60% of users exhibit this behavior switching.

#### *Other perspectives of trust online*

While they don't easily fit into one of the categories of approaches to understanding or managing trust online, there have nonetheless been several other attempts to understand and improve trust in online settings. These projects stem from diverse sources, including government, academic, private sector, and various open source software communities. Not all of them include a trust metric and/or a trust management system; many of them aim, instead, to clarify conceptual understandings, to propose roles and rules related to trust, and similar aims.

The term "web of trust" has been used by many different individuals and organizations with varying goals and definitions, but here we discuss the web of trust as relates to PGP (Pretty Good Privacy) and other similar encryption and public key infrastructure (PKI) programs. In contrast to traditional PKI programs which rely on specified certificate authorities (CAs), in web of trust encryption programs each user is also

by definition a CA. Web of trust is generally more of a hard trust measure, though it features similarities to some soft trust measures too. Many who study the web of trust in PGP and similar systems use a measure that is frequently used by soft trust metrics, namely, mean shortest distance (MSD).

In this book chapter [3.6], the authors examine the influence of trust on contributors to open source software projects. The authors identify two categories of motivations for open source contributors: intrinsic and extrinsic. Intrinsic motivations may be enjoyment based (the contributor does so because she likes doing it), or obligation based (the contributor feels that she owes something back to the project). The authors state that trust is a critical ingredient in enhancing cooperation and success of open source development projects, and that trust in this context tends to be more institutional than personal. Because many open source projects consist of different contributors who can enter and exit whenever they wish, the development of trust in open source projects doesn't rely as much on repeated personal interactions as it does in other contexts. The authors posit that trust in open source software development instead takes the form of swift trust, which is a type of trust observed in teams that work together for a limited amount of time. Within swift trust, the authors identify two types: encapsulated interests, and cognitive trust. Encapsulated interests takes a rational approach, described by "Alice trusts Bob because Alice believes it is in Bob's best interest to trust Alice." Cognitive trust is based on an estimation of the characteristics of the people that are working together on a project. The authors state that development and presence of trust is critical in attracting

new members because new members will be concerned about the ability of the group to resolve conflicts suitably. Because of the nature of open source software development, potential new members can easily observe the behavior of the group, and thereby make their own estimates of how much to trust (or not) that group. On the side of the trustor, the authors state that “extrinsically motivated trust based on encapsulated interests is sufficient”. On the side of the trustee (the sink of the trust), there needs to be a certain minimum number of intrinsically-motivated contributors to enable development of swift trust from trustors. The authors recognize that institutions must be in place to encourage participation of intrinsically-motivated contributors, and ensuring that the contribution costs are low enough to ensure that even intrinsically-motivated contributors don’t leave.

In [3.54] the authors propose Trust- $\chi$ , which is an XML-based framework for online trust negotiation in P2P settings. Unlike other trust negotiation-focused research up until the time of this paper’s publication, Trust- $\chi$  proposes a comprehensive trust negotiation system which address all aspects of the trust negotiation process (other works until this paper considered only one part of the overall process). During trust negotiation, a requestor (the node requesting access to a resource) and a controller (the node which controls access to the requested resource) exchange certificates and trust tickets. Trust- $\chi$  provides a framework for exchanging and evaluating these certificates which includes a policy base module, a tree manager module, and a compliance checker module, for both requestors and controllers, all specified in the  $\chi$ -TNL (trust negotiation language) which is based on XML. The policy base module stores the given nodes disclosure policies (which

can differ for each node). The tree manager module records the state of the negotiation at hand, and the compliance checker module tests compliance with the policies, then determines replies to requests based on compliance.

The National Strategy for Trusted Identities in Cyberspace (NSTIC) was announced in 2010 [3.55], established as a steering group by President Obama to *“administer the process for policy and standards development for the Identity Ecosystem Framework in accordance with the Guiding Principles in [the] Strategy. The steering group will also ensure that accreditation authorities validate participants’ adherence to the requirements of the Identity Ecosystem Framework”* [3.56]. NSTIC was guided by four core principles: 1) privacy-enhancing and voluntary, 2) secure and resilient, 3) interoperable, and 4) cost-effective and easy to use [3.57]. NSTIC was eventually transitioned to NIST (National Institute of Standards and Technology), where nearly two dozen pilot implementations of trusted identity were carried out through the Trusted Identities Group of the Applied Cybersecurity Division of the Information Technology Laboratory [3.58]. The Trusted Identities Group was eventually subsumed by the Identity and Access Management group, where some of its work lives on today.

The authors of [3.59] make a comprehensive survey of trust management systems available up until that time. The authors categorize trust management systems into six distinct categories, depending on the approach taken by the system: 1) direct trust, 2) indirect trust, 3) socio-cognitive trust, 4) organizational trust, 5) trust-aware interaction decision-making, and 6) performance assessment. Direct trust models rely on observable

behaviors of agents. Indirect trust models rely on third-party testimony of other agents. Socio-cognitive trust models utilize social relationships among agents to estimate trust. Organizational trust models use affiliations with organizations to estimate trust. Trust-aware decision making models seek to provide formal frameworks for how to *use* trust estimates to make decisions. Performance assessment models utilize real world datasets to evaluate the performance of other trust models. The authors identify areas for future research in trust models, identifying nine attributes that trust models should incorporate to ensure their success: 1) accurate for long-term performance, 2) weighted toward current (rather than historical) behavior, 3) efficient, 4) robust against attacks, 5) amenable to statistical evaluation, 6) private, 7) smooth, 8) understandable, and 9) verifiable.

The authors of [3.60] extend an existing computational trust framework (Jøsang's Subjective Logic, or SL) by proposing an improved method for initial trust computations which serve as the starting inputs to the trust framework. The authors characterize trust frameworks as a form of soft security mechanism, that is, they implement the social idea of trust in computing environments as a way to improve the quality of interactions among agents. The authors first conceptualize computational trust metrics as having two primary components: trust computing, and trust manipulation. In trust computing, trust values are calculated using measures such as reputation scores, and in trust manipulation new trust values are computed (or imputed) using different methods. The authors point out that, at the time of the paper's publication, most models focused primarily on one or the other of these pieces (trust computation or trust manipulation). The authors briefly summarize SL



as having four primary components: belief, disbelief, uncertainty, and a base rate (referring to prior interactions). The authors propose and describe a formal language structure for representing the input elements for belief, disbelief, and uncertainty, and how to compute their initial values which can then be plugged into the SL model. The pre-trust computational framework makes use of a selected state of an agent, together with a neighborhood function which defines what the selected agent knows when in a given state. Given a certain number of propositions about an agent (for example, that the agent has knowledge in a relevant field in which a transaction is to take place), all possible states of trust can be calculated based on the propositions. And, importantly, the framework considers trust in a given context, rather than how much one agent trusts another generally.

NIST's (National Institute of Standards and Technology) Information Technology Laboratory developed and published *SP 800-207: Zero Trust Architecture* in 2018 (with subsequent updates in 2020) [3.61]. While not a trust metric nor a trust management system and, in fact, taking the opposite approach of assuming no trust in computer systems, Zero Trust Architecture (ZTA) is nonetheless related to the work described previously because it considers many of the same challenges and problems that trust management systems aim to solve, but in a different way. ZTA provides a cybersecurity framework that recognizes the shifting landscape of modern enterprise technology infrastructures and platforms, and the corresponding effects that these have on an organization's cybersecurity stance. One of the key tenets of ZTA, which is an evolution

from older cybersecurity stances, is that it focuses on protecting resources themselves, rather than on network segments. The publication provides deployment models and use cases. ZTA lays out seven core tenets that together provide a stronger security stance: 1) all data sources and computing services should be considered as resources, 2) all communications should be secured, regardless of where in the network those communications are or pass through, 3) access to resources should be granted on a per-session basis, 4) access to resources should be granted on a dynamic basis, 5) the enterprise should monitor security of all resources, 6) authentication and authorization are strictly and dynamically enforced before they can be used, and 7) the enterprise should collect as much data as possible about the resources to be used to further improve security. More directly related to this dissertation, ZTA does include a trust algorithm as part of its proposed policy engine (which is one of the key logical components of an overall ZT architecture). The trust algorithm is the specific process used by the policy engine to decide whether or not to grant access to a resource. ZTA doesn't define the specific trust algorithm as it will vary for each organization, but in general this type of trust measurement system can be considered as a type of hard trust (previously discussed in this dissertation). Different versions of ZTA have experienced growing popularity in industry and government in recent years.

### *Social and psychological perspectives on trust*

While not the focus of this dissertation, it is impossible to ignore the extensive research work related to trust (as well as mistrust and distrust) that has been developed by

researchers in fields including psychology (particularly social psychology and industrial/organizational psychology), sociology, management, and education. Most of the trust systems reviewed in the preceding sections of this chapter are based either directly or indirectly on insights related to trust from researchers in these fields, and trust as a concept in online settings loses meaning and specificity if not first considered from this foundational perspective. [3.62] identifies improved outcomes for teaching and learning when professional communities of practice are fostered for teachers and principals in schools, and – interestingly for our work – indicates evidence in support of closer network relationships (from our perspective, this means denser networks, and more reciprocal networks) developing in the presence of higher trust levels. [3.63] provides a thorough review of organizational trust concepts up until the time of the paper’s publication (1999). This paper discusses different ways of thinking about trust, which may be a psychological state, a choice behavior – or both. [3.63] also considers factors that lead to the formation of trust. These include dispositional trust, which is someone’s natural propensity to trust someone else with whom they have no prior history. History-based trust is affected by past interactions between two (or more) people. Third parties may be conduits of trust (this is essentially the same transitive trust property discussed earlier in this chapter). Category-based trust results from mutual belonging of two people to the same group, which could be a social or professional organization. Role-based trust results from the position of authority held, for example, by a supervisor. And, finally, rule-based trust is based on shared understandings and acceptance of rules (which may be explicit or implied) about what types of behaviors are appropriate. [3.63] also considers the benefits of trust in

organizations. These include reduced transaction costs (as referenced in Chapter I and [1.1]), spontaneous sociability (cooperative, altruistic behavior of people within groups with high trust), and voluntary deference (which refers to the acceptance of and implicit trust in authority figures in hierarchical structures). Finally, [3.63] identifies several barriers to formation of trust, which include suspicion, violation of expectations, the ability of certain technologies such as surveillance systems targeted at employees to undermine trust (this is surely even more relevant today than it was in 1999, when the paper was published), and the fragility of trust (following the old adage that trust is hard to build but can be undermined in an instant).

### **Open challenges in the design and deployment of trust metrics and trust management systems**

While online trust metrics and online trust management systems have been around for more than two decades now, there remain open challenges in the design and application of these systems in online contexts.

One of these challenges is how to deal with ever-expanding online networks. Facebook today has nearly three billion monthly active users, Twitter has more than 300 million monthly active users, and in 2020 WhatsApp claimed more than 2 billion monthly active users. As networks expand their complexity grows exponentially because of the new links or potential links created with each new node that attaches. For this reason, the analysis and management of trust in these massive online networks becomes more difficult with time, requiring more sophisticated and smarter techniques – together with more powerful computing resources.

Another open challenge in design and application of online trust management systems is how to deal with new types of online networks that are increasing rich in media like videos – examples of such networks include TikTok, YouTube, Instagram, and others. The inclusion of video in a network makes the challenge of trust inference more difficult and resource-intensive.

The fact that many users of online services are active not just in one, but several online networks and social media platforms introduces the challenge of understanding trust across these heterogeneous networks. In addition to social media platforms, other online networks based on blockchain or private, closed networks could benefit from data fusion across these networks to better understand trust.

A perpetual challenge for designers of online trust management systems is their ability to resist to attack and manipulation. This challenge won't go away, as the attractiveness of a network for malicious actors only grows as the network grows, as alluded to earlier. We see an ever-accelerating prevalence and sophistication of cybersecurity incidents and attacks on networks, and trust management systems will need to continue to evolve and improve to meet these challenges.

Finally, an important open challenge in online trust metrics and trust management systems is the need to develop a science of trust metrics. Thus far, most of the work of trust systems (including this dissertation) has been empirical and for the most part not generalizable to other networks (i.e., a system designed for WhatsApp won't have strong usefulness for YouTube). Although many scholars have done work in this area already,

more remains to be done to put forth a theoretical description of trust online that can be used to make and test predictions and is more generalizable across different applications of online trust.

In the next chapter, we spend time considering how to compare disparate trust metrics and trust systems, and propose a decision making framework for how designers, managers, or owners of online networks can systematically go about selecting a trust model – or designing their own – by first defining the context in which it will be required to operate and the types of data they will have access to.

#### **IV. Benchmarking performance of trust metrics, and proposed framework for matching a trust metric with an application**

##### **Introduction**

As has been detailed in Chapter III, there are myriad trust metrics and trust management systems available for online applications with a wide range of different features, effectiveness, and applicability (and although Chapter III is quite detailed, it still isn't a complete list). In Chapter III we described a wide range of features that different trust management systems require as input to estimate trust, making different trust management systems more or less effective when different types of data are available. Given this wide array of options, how are practitioners to go about selecting the trust management system best suited for use in their specific application?

In this chapter, we seek to address this issue through two main approaches. First, we develop a set of experiments that we use to benchmark the performance of a select few methods under varying conditions. Although the specific trust metrics that we use for benchmarking in this chapter will not fit every potential application, we believe the methods proposed can be adapted to experiment with different trust metrics not included in the scope of this chapter's experiments to determine their suitability for a specific application. While most of the researchers who produced the papers described in the previous section provide meticulous testing of their metrics and systems and compare

them to the performance of other systems, there is a structural incentive for them to report superior performance of their systems compared to others. Thus, this chapter serves to provide a neutral and objective review of systems' performance under different conditions. While we include a measure of our own design in this chapter's review, it is included simply for purposes of comparison; the aim of this dissertation is not to propose a new, faster, or more accurate trust management system, but rather to derive insights from real-world networks using trust measurement methodologies. Second, in this chapter we also develop a decision making framework that can be used to narrow the field of candidate trust management systems that a platform owner or platform manager might consider for their own specific applications. This decision making framework is developed using the results of the benchmarking experiments outlined in the first part of this chapter.

### Research questions

The research questions pursued in this chapter are foundational for the rest of this dissertation, as they will influence the different research strategies we apply in later chapters. In this chapter, we present and explore the following research questions related to performance of trust metrics in online settings:

**RQ4-1:** *How do various trust metrics' accuracy in estimating the ground truth trust values vary with different graph structural characteristics?*

**RQ4-2:** *How do various trust metrics' code execution times vary with different graph structural characteristics?*



**RQ4-3:** *Can we identify approximate ranges of values that balance accuracy with speed of code execution of trust metrics for use in online networks?*

**RQ4-4:** *Can we develop a generalized framework for selecting which trust metrics or trust management systems are best suited for use in different applications?*

To answer these research questions, in later sections of this chapter we propose a series of experiments that consider different variables related to graph structure, and measure the performance of different trust metrics while these variables change.

### **Related work**

In addition to the trust metrics and trust management systems that received an exhaustive review in Chapter III, in this section we briefly discuss work related to benchmarking performance of trust metrics.

Most papers that present a new proposed design for a new trust metric or trust management system (those described in Chapter III) provide benchmarking of the new metric's performance against existing metrics, or against different versions of itself. However, these benchmarking results are typically provided with the aim of providing evidence in support of the effectiveness of the trust metric under consideration – not for

providing a neutral examination of effectiveness of multiple trust metrics. A limited number of trust metric benchmarking studies have, however, been conducted.

In [4.1] the authors propose and evaluate the effectiveness of a testbed for trust models. This paper primarily develops a formal model for evaluating different attack strategies on trust management systems for online platforms. [4.1]'s authors consider attack resistance (and tradeoffs in resistance to different attack strategies) of EigenTrust, PeerTrust, and Appleseed.

In [4.2] the authors, as part of their systematic literature review, present a system for classifying different trust systems along various dimensions. This work discusses some performance benchmarking results from some of the systems reviewing in the paper, but it does not conduct or present a neutral evaluation of multiple trust metrics against graph structural characteristics.

In [4.3] the authors present an excellent treatment of trust management systems in multi-agent settings, and propose a useful categorization system for classifying different trust management systems. Additionally, the authors provide performance benchmarking results, but these are limited in their scope.

### **Benchmarking performance of trust metrics**

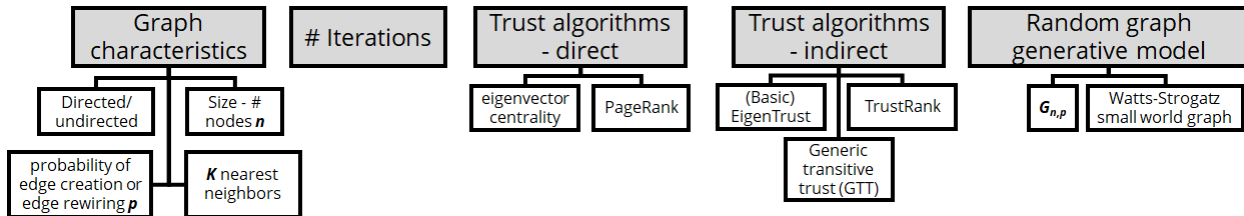
In this section, we first perform benchmarking of selected popular trust metrics and trust management systems to better understand their performance along different dimensions. We describe our methods for selecting which features and which trust management systems to consider in our experiments; our procedure for setting up our

experiments; the results of the experiments; and discussion and interpretation of the experiments' results. In the next section, we will take these results and use them to develop a decision making framework that can be used by practitioners or other researchers for identifying promising candidate trust management systems for use in their particular applications.

### *Design of benchmarking experiments*

Using Table 2 and Table 3 (in Chapter III), we identify categories of features needed to design our benchmarking experiments in this chapter. The categories of features of interest for our experiments are presented in Figure 5 and include graph characteristics, the number of iterations each experiment will run for, which direct trust algorithms to use for estimating trust, which indirect trust algorithms to use for estimating trust, and which random graph generative models to use for creating graphs for use in the experiments. We select these features for inclusion in our experiments because 1) as measures that are purely structural (i.e., derived from the network's connectivity) they fall within the scope of this dissertation as outlined in Chapter II, and 2) because they are measures that will be universally available to platform managers/operators in almost any online platform and, in many cases, also available to the public as open data. Other features (for example, history of private messages exchanged between two users of an online platform) identified in Chapter III are useful for estimating and managing trust, but data on these features are not universally available for all online networks, nor are they typically made available to the public for research. Our success measures of interest for the experimental results are

performance of the algorithms as measured by mean absolute error (MAE), code execution time, and a combined measure of the two.



**Figure 5:** Categories of features considered in design of benchmarking experiments

Using the categories of features presented in Figure 5, we develop three different experiment sets (one of which has two subparts) which each change a different variable. In Table 4, we present summaries of each of these different experiment sets. The first experiment set (1.1 and 1.2) considers variations in network size as measured by  $n$  number of nodes (which, together with  $p$ , directly affects the number of edges generated for the random graphs), and two different random graph generative models (the Watts-Strogatz small-world generative model, and the  $G_{n,p}$  or Erdős-Rényi model). Set 2 varies the probability  $p$  of edge rewiring for the Watts-Strogatz small world network model. And, in Set 3, we vary the number of nearest neighbors  $K$  for the initial generation stage of the Watts-Strogatz small world model.

**Table 4:** Experiment set combinations. Features highlighted in green signify the feature that is variable for that experiment set.

Experimental variations						
	Experiment	$n$	$p$	$K$	# iterations	Random graph generative model
Set 1.1, Variant: # nodes, $n$ , generative model	1.1.1	400	0.02	150	3	Watts-Strogatz small-world
	1.1.2	800	0.02			
	1.1.3	1600	0.02			
	1.1.4	3200	0.02			
	1.1.5	6400	0.02			
Set 1.2, Variant: # nodes, $n$ , generative model	1.2.1	400	0.02	150	3	$G_{n,p}$
	1.2.2	800	0.02			
	1.2.3	1600	0.02			
	1.2.4	3200	0.02			
	1.2.5	6400	0.02			
Set 2, Variant: prob. of an edge, $p$	2.1	400	0.02	150	3	Watts-Strogatz small-world
	2.2	400	0.05			
	2.3	400	0.5			
	2.4	400	0.95			
	2.5	400	0.98			
Set 3, Variant: $K$	3.1	400	0.02	50	3	Watts-Strogatz small-world
	3.2	400	0.02	100		
	3.3	400	0.02	150		
	3.4	400	0.02	200		
	3.5	400	0.02	250		

### Feature categories

Referring back to Figure 5, in this section we provide the rationale for selection of each category of feature to be incorporated into our benchmarking experiments.

Based on our review in Chapter III of features that are typically taken as inputs for computing trust metrics, in the category of graph characteristics we identify the following features as those to be included and varied for our benchmarking experiments: size of graph as measured by number of nodes  $n$ ; whether the graph is directed or undirected; the probability  $p$  of either generating an edge (in the case of  $G_{n,p}$  random graphs) or rewiring an edge (in the case of Watts-Strogatz small world graphs); and  $K$  nearest neighbors to generate (in the case of Watts-Strogatz small world graphs).

### *Trust Measures (Direct and Indirect)*

For our trust algorithms, we select two graph influence measurement algorithms to serve as proxies for estimating direct trust, and three trust metrics developed to measure indirect trust in graphs. Direct influence measures used in our experiments are eigenvector centrality and PageRank, and the measures for indirect trust estimation are Basic EigenTrust, TrustRank, and a generic transitive trust (referred to hereafter as GTT) measure that we develop for use in this dissertation specifically. The direct trust estimation metrics of eigenvector centrality and PageRank were selected based on the findings of [4.4], in which the authors find graph centrality measures PageRank and eigenvector centrality are most appropriate for application to computing direct trust metrics. [4.4]'s authors find information in a trust network flows along simple paths (such as a direct path between a pair of nodes) *and* non-simple paths (such as triangles), and thus they conclude that measures like eigenvector centrality and PageRank can be expected to give the most reliable measures of direct trust in a complex network since trust can flow in simple paths and non-simple paths alike in trust networks.

### *Eigenvector Centrality*

Eigenvector centrality considers centrality of a node with respect to centralities of its neighbor nodes. It is particularly useful and robust for different types of networks because a node can achieve a high importance through having many connections, or by being connected to other important nodes – or both. The eigenvector centrality of a given node is

calculated as in Equation 1 [4.5], where  $x_i$  is the node in question,  $A_{ij}$  is an element in an adjacency matrix,  $k_i$  is the largest of the eigenvalues of  $\mathbf{A}$ , and  $x_j$  are  $x_i$ 's neighbors:

$$x_i = k_i^{-1} \sum A_{ij}x_j$$

**Equation 1:** *Eigenvector centrality of a given node,  $i$*

### *PageRank*

The PageRank algorithm served as the initial foundation for the first version of the Google search product. PageRank was first proposed for ranking importance of Web pages, but because it returns a ranked listing of nodes in a network it has been applied broadly to a diverse mix of other domains. PageRank makes use of the same centrality feedback mechanism in which nodes with higher prestige provide more prestige when linking to other nodes [4.5], with an important difference being that it relies on a random walk strategy (which eigenvector centrality does not).

### *Basic EigenTrust*

Basic EigenTrust is used in this chapter's experiments and, as its name indicates, is a simplified version of the EigenTrust algorithm [4.6] which does not incorporate pre-trusted nodes or attack resilience measures. We refer the reader back to Chapter III for a deeper discussion of the mechanics of how the EigenTrust system works.

An important caveat to this chapter's results and conclusions is that many real world platforms will wish or need to include a robustness mechanism if implementing EigenTrust, which increases the computational complexity of the algorithm, and thus the EigenTrust

values discussed in this chapter should be considered as the minimum possible (MAE and code execution times), with values increasing if using one of the other variations of EigenTrust.

#### *TrustRank*

TrustRank was originally developed for application to Web pages but, similar to PageRank, has been successfully adapted for use in other contexts. For implementing these algorithms, we use as a starting point [4.7]’s implementation of Basic EigenTrust and [4.8]’s implementation of TrustRank, but make significant adjustments to both implementations to accommodate the way we generate and analyze graphs in this chapter.

#### *Generic Transitive Trust (GTT)*

Finally, we also include a generic transitive trust (GTT) measure developed for use in this dissertation. This dissertation doesn’t propose a new trust management system as in many of the papers reviewed in Chapter III; instead, GTT is used as an approximate amalgamation of many of the most common principles observed in many of the trust metrics reviewed in Chapter III (such as maximum trust propagation distance and trust aggregation methods) which is more straightforward to use in analysis in later chapters of this dissertation. Most transitive trust metrics include a trust propagation function, a trust aggregation function, and a trust decay function. GTT makes use of each of these elements. For the trust propagation function, we adapt the method in [4.9]. This paper’s authors verify small worldness in the networks they analyzed and, upon doing so, set the maximum trust propagation distance (MTPD) as the limit of geodesic path length. The authors verify



that the small decrease in accuracy of their trust metric resulting from setting MTPD equal to geodesic path lengths is minimal, while gains in code execution speed compared to the optimal solution are significant. Thus, for GTT we set the MTPD equal to the geodesic path length, rounded up to the nearest whole number.

For GTT's trust decay function we adapt a method described in [4.10], setting trust decay as equal to the reciprocal of the distance between source and sink.

For GTT's trust aggregation function, we aggregate trust values for all nodes in each network into a raw (non-normalized) trust value, which is simply the sum of all of the individual transitive trust values placed in each sink node.

Finally, and critically, for these experiments we sample paths in the networks rather than computing all paths, since for the larger graphs in this chapter's experiments ( $n = 3200$ ,  $n = 6400$ ) the amount of time needed to execute the code would be prohibitively long (based on  $O(n^3)$  where  $n$  equals the number of nodes in the graph; for  $n = 6400$  this would be more than 262 trillion potential trust paths). In order to properly scale the trust metric along with the size of the graph being analyzed, we set the number of samples to equal  $2n$ , such that the expected value of the number of times a trust path is calculated that contains each node as a sink is two. Trust values in GTT take continuous values from 0.0 to 1.0.

### *Graph structural characteristics*

For our experiment sets, we consider graph size as one of the most important features to vary in our experiments. Graph size has a direct effect on the execution time of most trust metrics, and as such it is critical to include in our experiments. To generate

graphs of different sizes, we include  $n$  number of nodes in our experiment sets which, when combined with  $p$  (probability of creating an edge in the  $G_{n,p}$  model) or  $K$  (number of neighbors to which nodes initially connect in small world graphs) determines the number of edges (and thus, the code execution time for trust algorithms) present in the graph. In Experiment Sets 1.1 and 1.2 we vary the number of nodes, beginning with  $n = 400$  and doubling four times from to  $n = 800$ ,  $n = 1600$ ,  $n = 3200$ , and  $n = 6400$ . In the case of the small world graphs, these lead to graphs ranging in size from 30,000 edges (for  $n = 400$  and  $K = 150$ ) to 480,000 edges (for  $n = 6400$  and  $K = 150$ ). In the  $G_{n,p}$  random graphs, the expected values of number of edges ranges from 1,600 (for  $n = 400$  and based on  $p = 0.02$ ) to 409,600 (for  $n = 6,400$  and based on  $p = 0.02$ ).

In addition to graph size, another important feature to incorporate in design of benchmarking experiments is whether the graph is directed or undirected. A directed edge in an online social network might look like, for example, User A “liking” a post by User B. An example of an undirected edge in an online social network, on the other hand, could be the acceptance of a friend request on Facebook or a connection request on LinkedIn. For these experiments, we select to consider only undirected networks, but a future extension of this research should be to incorporate experiments that compare performance of trust algorithms in directed networks and undirected networks.

$p$  and  $K$  are also considered as key graph structural characteristics to be used as inputs in our experiments but, as they are specific to the random graph generative models in which they are used, they are described in greater detail in the next section.

### *Random graph generative models*

For our experiments we utilize two random graph generative models: the  $\mathbf{G}_{n,p}$  random graph, and the Watts-Strogatz small world graph. The  $\mathbf{G}_{n,p}$  random graph, also known as an Erdős-Rényi graph, [4.11] takes the number of nodes ( $n$ ) in the graph and a probability of edge creation between nodes ( $p$ ) as inputs. The Watts-Strogatz small world graph takes as inputs the number of nodes ( $n$ ) in the graph, the number of neighbors that each node will initially connect to when generating the network ( $k$ ), and a probability of rewiring edges between nodes and their neighbors ( $p$ ) [4.12]. The Watts-Strogatz model begins with a ring lattice of  $n$  nodes and  $k$  edges per node, then with probability  $p$  rewires the edges that were initially generated, randomly creating short paths between nodes that were previously only distantly connected (hence the name, small world).

Referring to the framework of [4.12], a regular (completely non-random) graph has  $p = 0.0$  and a completely random graph has  $p = 1.0$ ; small world graphs fall somewhere in between the two, with even small amounts of randomness among connections leading to small worldness. Researchers have confirmed countless examples of small worldness in real world networks (especially in social contexts), and thus we use the small world graph as our primary generative model to be considered in our experiment sets, but also include one experiment set that utilizes the  $\mathbf{G}_{n,p}$  generative model to compare the performance of trust metrics in networks of that type. Both generative models are applied in this chapter using their implementations in NetworkX [4.13] and [4.14].

For selecting  $K$  in the small world graphs, we refer to Dunbar's Number [4.15]. Dunbar's Number is a purported approximate cognitive limit to the number of meaningful social relationships the average person can maintain. Many researchers have estimated the number to be approximately 150 [4.16], [4.17], but the number has also been estimated to be both higher and lower than 150 by other researchers, in particular as a result of modern information and communication technology expanding our circle of potential social relationships, and their very meaning [4.18], [4.19], [4.20]. Thus, we take  $K = 150$  for generation of the small-world random graphs in Experiment Sets 1.1, 1.2, and 2, and in Experiment Set 3 we vary  $K$  in increments of 50 from  $K = 50$  to  $K = 200$ .

#### Ground truth trust

For each random graph generated in our experiments, we first generate random trust values between all node pairs in the graph, i.e., trust is only generated when an edge exists between two nodes so that the trust generation process stays true to the actual structure of the random graphs. Random trust values are generated uniformly between 0.0 and 1.0. The total global trust for each sink node is computed as the sum of all trust values that source nodes place in this sink node, and then normalized. We perform this exercise as the first step of our experiments for all random networks specified previously, in Table 4. These trust values are used as the ground truth trust values against which the accuracy of the trust algorithms selected for our experimental analysis will later be compared.

## Success measures

We consider two success measures in this chapter's experiments: accuracy of the trust algorithms compared to the ground truth trust values that were generated as described in the previous section, and code execution speed.

To measure algorithm accuracy, we utilize Mean Absolute Error (MAE). Because the ground truth trust values are continuous values in the range of 0.0 – 1.0, we select MAE as our accuracy measure rather than one that would normally be utilized for measuring accuracy in binary classification tasks such as precision, recall, F1 score, or similar measures. MAE has been studied extensively as a simple, easy-to-calculate, and easy-to-understand measure of error that is readily comparable against other MAEs [4.21]. MAE is measured as in Equation 2, by summing the total absolute error (absolute value of errors) in a sample and dividing by the sample size.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |e_i|$$

**Equation 2:** Mean Absolute Error (MAE), from [4.21]

To measure code execution speed, we make use of the *time* module in Python3 [4.22]. At the start of each new iteration of execution of a trust measure, we initiate a new start time, and stop the time at the end of that iteration. We write the execution time of each iteration for each trust measure to a list for later analysis.

## Analysis

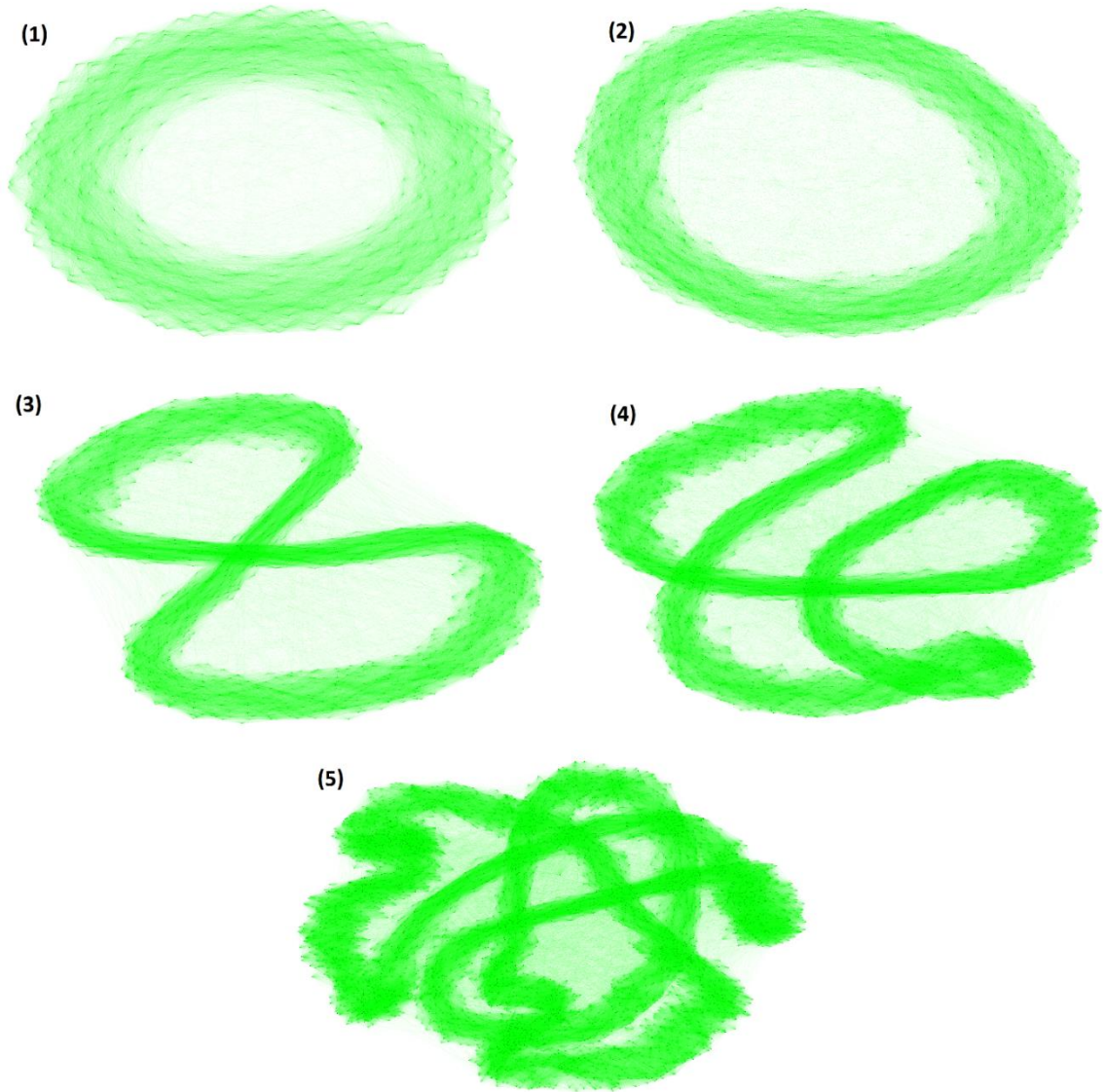
We first generate our random graphs for use in our various experiment sets. To get a sense for the size and structure of the graphs, we present visualizations of each of the different random small world networks and  $\mathbf{G}_{n,p}$  networks used for Experiment Sets 1.1 and 1.2 in Figure 6 and Figure 7, respectively.

The first thing to point out about the visualizations is the marked difference between the small world graphs (in all of their variations) and the  $\mathbf{G}_{n,p}$  random graphs. Referring to the Helmholtz Principle from the Gestalt Theory of human perception as described in [4.23], we note that the human eye (and brain) is astoundingly good at picking out structure from randomness in images. While in the small world graphs, noticeable patterns or structures can be identified with a quick glance, there is no discernible structure in the random graphs of Figure 7. This (lack of discernible structure) is a key premise of why it is useful and important to use random graphs for comparison in this type of work: if the network under investigation differs markedly from the random equivalent version of itself, then there is some mechanism (or mechanisms) of interest responsible for a particular structure or behavior.

As is to be expected, in the smallest of the  $\mathbf{G}_{n,p}$  graphs there is relatively little density and connectivity (as measured by edges), and the graphs densify as they grow in size. In the case of the small world graphs, we observe a structure similar to the ring lattice described previously in the section outlining graph generative models, with more pathways forming between disparate parts of the graphs as they grow in size.

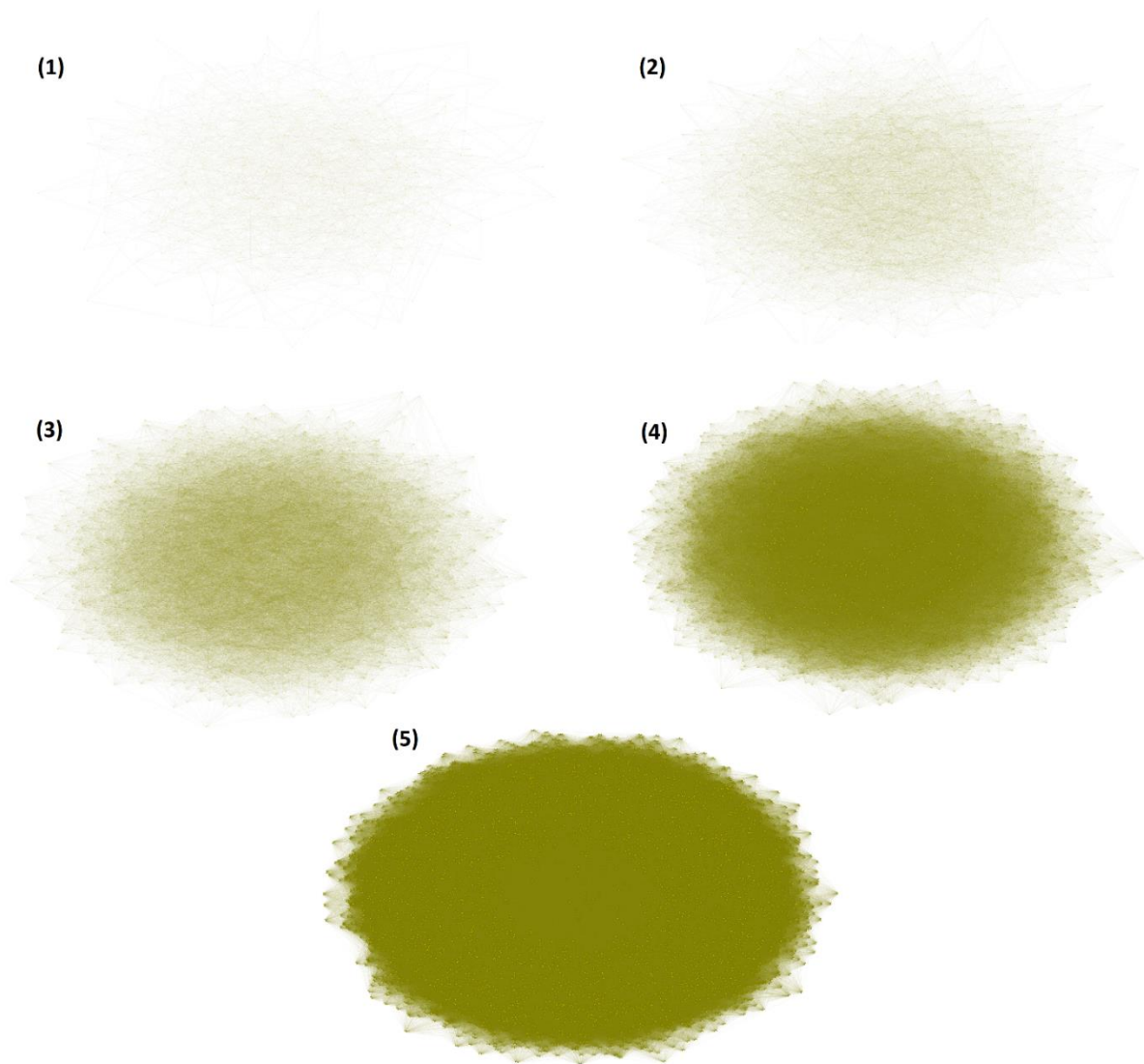
In Figure 8, we present visualizations of the random small world graphs used for Experiment Set 2, where the variant is  $p$ , probability of edge rewiring. As is to be expected, we see that for the graphs generated with a small  $p$  (0.02 and 0.05) they continue to look like small world graphs, whereas for the random graphs with larger  $p$  (0.50, 0.95, and 0.98) they look much more like a  $\mathbf{G}_{n,p}$  random graph (compare to the  $\mathbf{G}_{n,p}$  random graphs in Figure 7).

Finally, in Figure 9 we present visualizations of the random graphs used for Experiment Set 3, in which the variant is  $K$  (the number of nearest neighbors connected to nodes in the first iteration of the small world graph generation process). In these graphs, we observe a densification of the graphs as  $K$  increases, but even for the high  $K$  random graphs they continue to resemble small world graphs, rather than  $\mathbf{G}_{n,p}$  random graphs (as was seen in Figure 8).

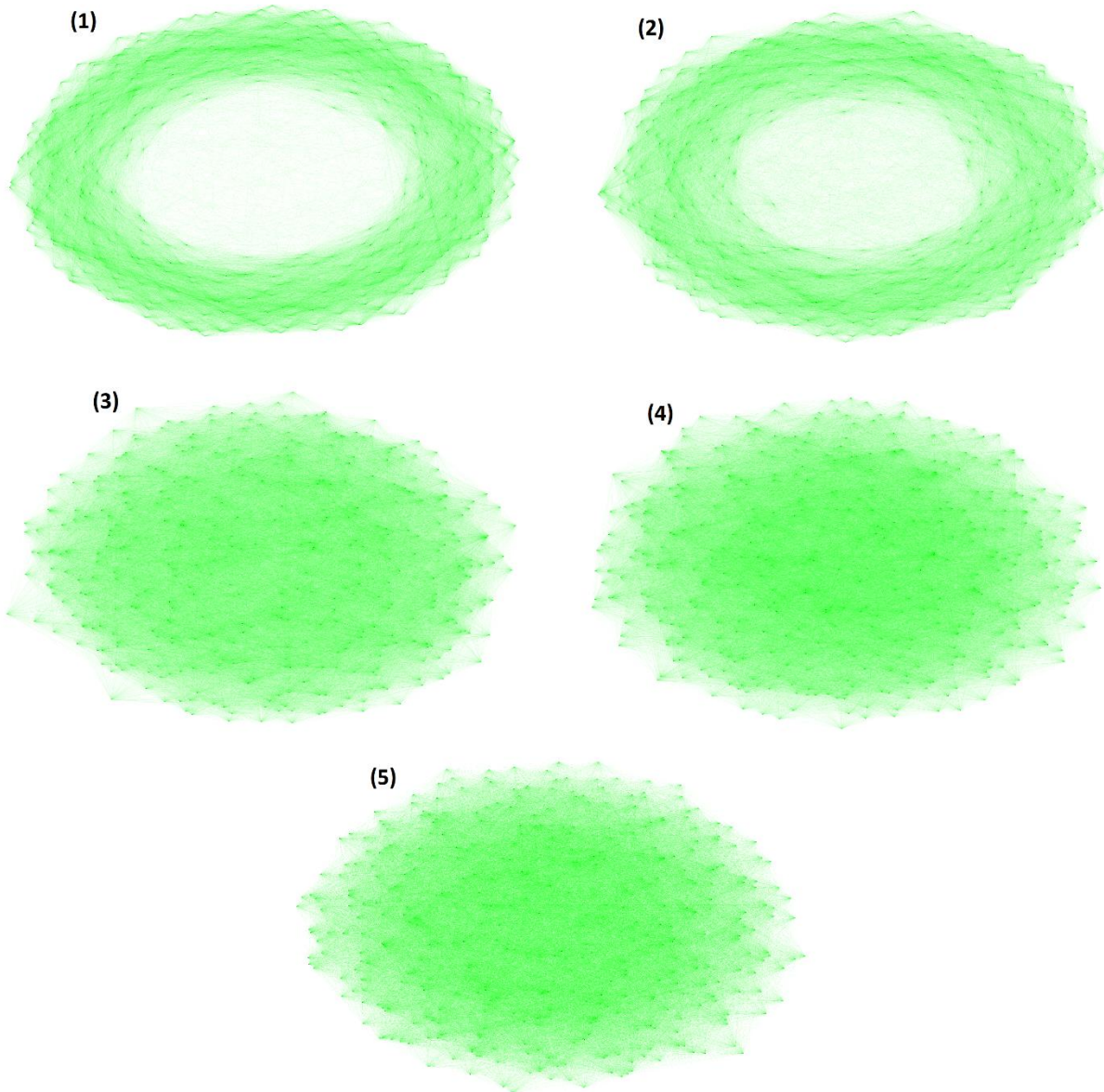


**Figure 6:** Visualizations of small world graphs of different sizes used for Experiment Set 1.1; panel (1) with  $n = 400$ , panel (2) with  $n = 800$ , panel (3) with  $n = 1600$ , panel (4) with  $n = 3200$ , and panel (5) with  $n = 6400$ .  $p=0.02$  and  $K=150$  for all. Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1.

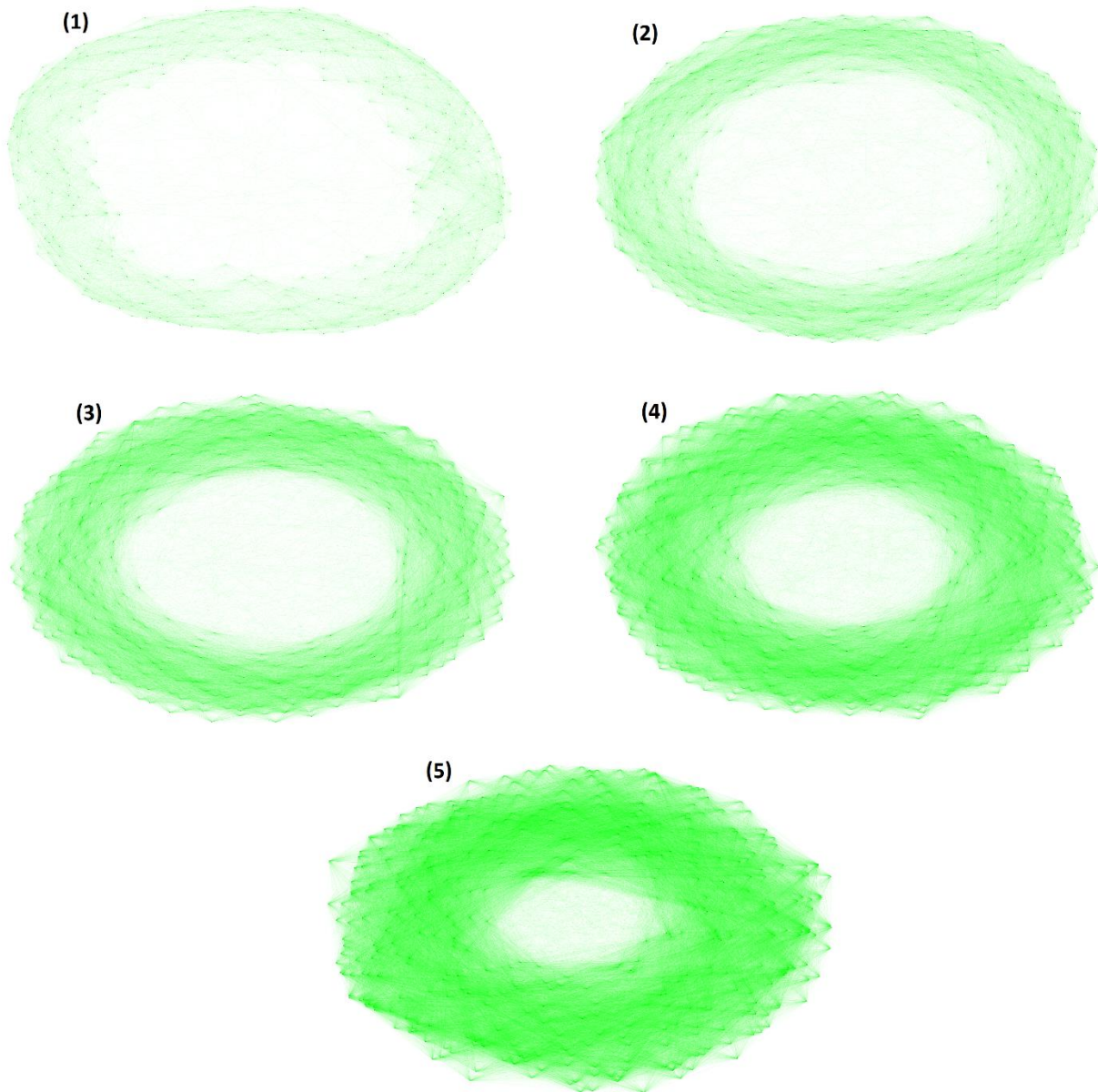




**Figure 7:** Visualizations of  $G_{n,p}$  random graphs of different sizes used for Experiment Set 1.2; panel (1) with  $n = 400$ , panel (2) with  $n = 800$ , panel (3) with  $n = 1600$ , panel (4) with  $n = 3200$ , and panel (5) with  $n = 6400$ .  $p=0.02$  for all. Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1.



**Figure 8:** Visualization of random graphs used in Experiment Set 2. In Experiment Set 2, we hold constant  $n = 400$  and  $K = 150$  while varying  $p$  from 0.02 to (Panel 1), to 0.05 (Panel 2), 0.50 (Panel 3), 0.95 (Panel 4), and 0.98 (Panel 5). Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1.



**Figure 9:** Visualizations of random small world graphs used in Experiment Set 3. In Experiment Set 3, we hold constant  $n = 400$  and  $p = 0.02$  while varying  $k$  from 50 to (Panel 1), to 100 (Panel 2), 150 (Panel 3), 200 (Panel 4), and 250 (Panel 5). Visualizations produced using NetworkX and Matplotlib in Python3 with spring layout, node size = 10 and edge width = 0.1.

Because each trust metric evaluated in this chapter measures trust differently, we can't make meaningful comparisons between the trust values estimated with different trust metrics by directly comparing their raw values. Instead, to be able to compare different systems' performance, we consider rankings of each specific node with each trust measure. We compute the distance between rankings of each node produced by the trust metrics, compared to the rankings of those same nodes in the ground truth trust set.

An important limitation to this approach includes the fact that we lose information and precision by considering rankings rather than raw trust values. For example, if Node 1 has a ground truth trust value of 0.8, Node 2 has a ground truth trust value of 0.55, and Node 3 has a ground truth trust value of 0.05, our method of comparing rankings will fail to capture the magnitude of difference in trust values between differently-ranked nodes. In this example, the difference in ground truth trust values of Node 1 and Node 2 (0.25) is only half the difference between the ground truth trust value from Node 2 to Node 3 (0.5), but by considering rankings we only see that Node 1 is the most trusted, Node 2 is the second-most trusted, and Node 3 is the least-trusted.

In addition to analyzing the success measures described previously (MAE, code execution time, and a combined measure of the two), we also analyze the correlations between several graph measures and success measures to better understand how specific graph characteristics affect trust metric performance so that we can use these as part of our selection model in the second part of this chapter. These include correlation between code execution speed and number of nodes, code execution speed and number of edges,

code execution speed and geodesic path lengths, code execution speed and graph mean clustering coefficient, and correlation between MAE and all of the previously listed characteristics.

Random networks generated for use in experiment sets

Using the categories of features and the methods described in the preceding sections, we generate our random networks to be used in each experiment. Table 5 presents structural characteristics of the random graphs that were generated for use in Experiment Sets 1.1 and 1.2; Table 6 presents structural characteristics of the random graphs generated for use in Experiment Set 2; and Table 7 presents structural characteristics of the random graphs generated for use in Experiment Set 3. We verify the small worldness of the small world graphs by observing high clustering coefficients in the small world graphs compared to those of the  $G_{n,p}$  graphs, and short geodesic path lengths (comparable to those of the  $G_{n,p}$  networks).

**Table 5:** Structural characteristics of randomly small world and  $G_{n,p}$  networks used in Experiment Set 1.1 and 1.2.

<b>Small-world</b>					
<b># nodes, n</b>	400	800	1,600	3,200	6,400
<b># edges, m</b>	30,000	60,000	120,000	240,000	480,000
<b>Clustering coefficient</b>	0.71099	0.704454	0.70354	0.70263	0.70194
<b>Geodesic path length</b>	1.6	2.0	2.4	2.6	2.8

<b><math>G_{n,p}</math></b>					
<b># nodes, n</b>	400	800	1,600	3,200	6,400
<b># edges, m</b>	1,669	6,453	25,541	102,004	409,414
<b>Clustering coefficient</b>	0.01991	0.02089	0.70354	0.01995	0.02001
<b>Geodesic path length</b>	3.0	2.7	2.4	2.3	2.06

**Table 6:** Structural characteristics for randomly-generated small networks used in Experiment Set 2.

<i>Small-world</i>					
<b># nodes, n</b>	400	400	400	400	400
<b>K nearest neighbors</b>	150	150	150	150	150
<b>probability of rewiring an edge, p</b>	0.02	0.05	0.50	0.95	0.98
<b># edges, m</b>	30,000	30,000	30,000	30,000	30,000
<b>Clustering coefficient</b>	0.71099	0.66429	0.38751	0.37546	0.37523
<b>Geodesic path length</b>	1.6	1.6	1.6	1.6	1.6

**Table 7:** Structural characteristics for randomly-generated small networks used in Experiment Set 3.

<i>Small-world</i>					
<b># nodes, n</b>	400	400	400	400	400
<b>K nearest neighbors</b>	50	100	150	200	250
<b>probability of rewiring an edge, p</b>	0.02	0.02	0.02	0.02	0.02
<b># edges, m</b>	10,000	20,000	30,000	40,000	50,000
<b>Clustering coefficient</b>	0.69509	0.70292	0.71099	0.71827	0.72858
<b>Geodesic path length</b>	2.4	1.9	1.6	1.5	1.4

### Results and Discussion

In this section, we present and interpret the results of each experiment set. For all experiment sets, we measure performance of each trust measure included as part of this chapter's scope; code execution time for each trust metric; and a combined measure of the two. We measure performance using mean absolute error (MAE), with the mean being taken over the set of three iterations that were performed for each experiment. The combined measure, which we call Balanced Performance in this chapter, is taken by multiplying MAE and code execution times by one another, with the interpretation being that a lower value of balanced performance signifies a more desirable result (as the values of MAE and code execution times approach zero, so does that of Balanced Performance).

### Key findings

This section summarizes key findings, which are discussed in greater detail in the following sections.

First, we find that the trust metrics evaluated in this chapter are, in general, more accurate when applied to  $\mathbf{G}_{n,p}$  random graphs than when applied to small world graphs. This result is expected, as small world graphs feature more complex paths than random graphs, which can lead to both decreased accuracy and increased code execution times. Nonetheless, since many real world networks to which these measures might be applied feature small worldness, it should be expected by the researcher that accuracy will be lower than would otherwise be achieved in a  $\mathbf{G}_{n,p}$  random graph.

In this chapter we find that the direct trust measures of eigenvector centrality and PageRank tend to be more accurate than the indirect trust measures for smaller networks when applied to small world networks, but as the network sizes grow the accuracy of the direct trust measures and two of the indirect trust measures (GTT and EigenTrust) tend to converge to approximately similar values. On the other hand, in  $\mathbf{G}_{n,p}$  random graphs, EigenTrust's accuracy is only marginally lower than that of the direct trust measures.

For small world networks, as the network size grows, the code execution times for EigenTrust and TrustRank increase exponentially, but the code execution times for GTT and both direct trust measures grow more slowly.

When considering a variable  $p$  (the probability of rewiring an edge) for small world networks, we find that accuracy of both direct trust measures and EigenTrust increases marginally (MAE decreases marginally) directly with increasing  $p$ . Accuracy of GTT is highest for very low and very high  $p$ , with an increase in MAE for values of  $p$  between 0.05 and 0.95.

For increasing  $K$  in small world networks, accuracy of all trust metrics evaluated in this chapter except for TrustRank remain approximately constant. For TrustRank, however, accuracy increases (MAE decreases) markedly for high  $K$  compared to low  $K$ .

### Experiment Set 1.1

In Experiment Set 1.1, we use the Watts-Strogatz small world generative model, varying the size of the random graphs from  $n = 400$ , doubling them until they reach a size of  $n = 6400$ , while holding  $p$  constant at 0.02 and  $K$  constant at 150.

Referring to Figure 10, we present performance of each trust measure used as part of our experiments. The first observation from Figure 10 is that, with the exception of TrustRank, each of the trust measures' performance are comparable to one another, operating in a band of between 0.27 and 0.33. Second, we observe that for both direct trust measures (eigenvector centrality and PageRank) considered in this chapter, for our generic transitive trust (GTT) measure, and for Basic EigenTrust, performance is approximately steady regardless of the network size. For TrustRank, on the other hand, we observe a marked decrease in performance as the size of the network grows.

In Figure 11 we present code execution time for each of the trust measures, with a semilog scale (vertical axis is logarithmic scale). As is to be expected, we observe an increase in execution time for each trust measure as the size of the network being analyzed increases. However, there are some important differences among the trust measures, with the execution time for the Basic EigenTrust algorithm increasing the fastest as the size of the network increases. Our GTT measure is the slowest for the smallest graph, but by the

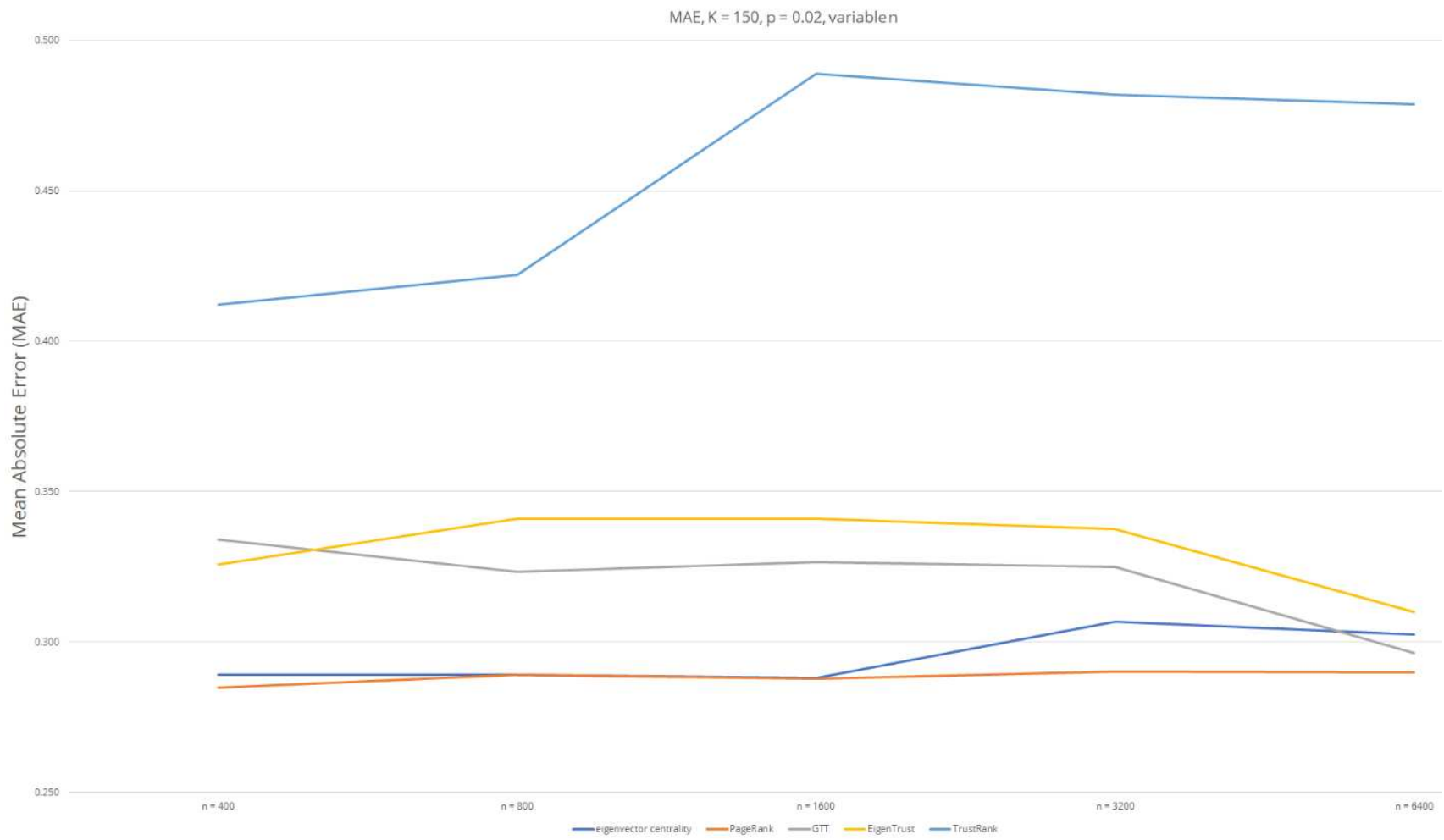


time we reach the largest network considered in these experiments it is the fastest of the indirect trust measures (i.e., it is slower than both eigenvector centrality and PageRank which are both direct trust measures, but it is faster than the other indirect trust measures).

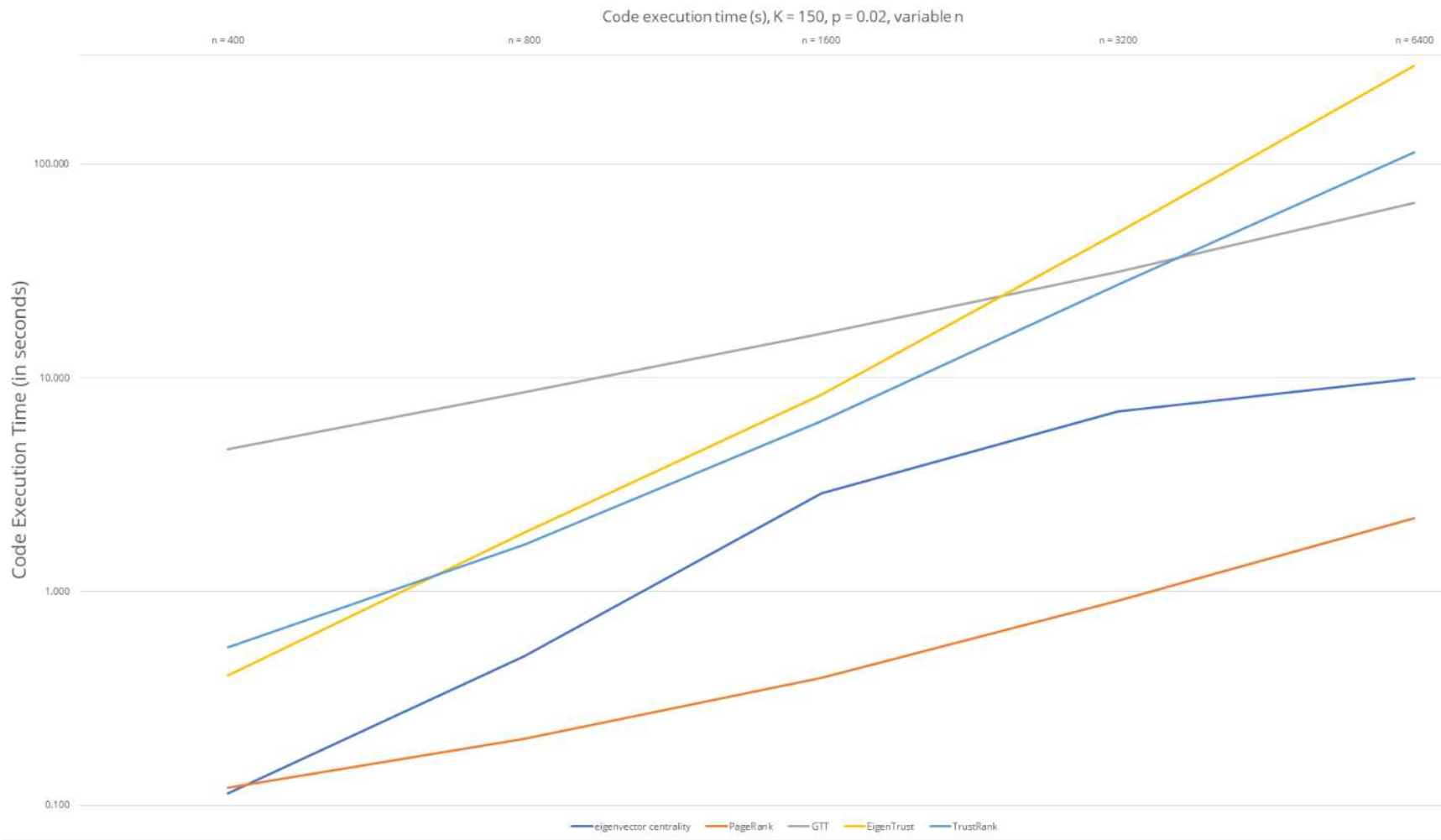
In Figure 12, we observe the Balanced Performance (previously described) of each trust metric. From this figure, we observe that for the indirect trust measures of EigenTrust and TrustRank their Balanced Performance increases exponentially (the figure's y axis is logarithmic scale), indicating that their usefulness will degrade when applied to much larger graphs (as would be expected in real world platforms). On the other hand, Balanced Performance of the direct trust measures of eigenvector centrality and PageRank, as well as the indirect trust measure GTT, grows more linearly, indicating they are likely to be more useful for platform managers when graph sizes are large.

These results provide important insight for platform owners or managers who might be considering implementing a trust management system into their network. If accuracy of a trust measure is the most critical outcome to be designed for, then – if the network to which they will apply their measure demonstrates small worldness – the platform manager might consider using one of the direct trust measures, or the indirect trust measures GTT or EigenTrust. If execution speed is of similar importance to the platform manager, then they might consider one of the direct trust measures (eigenvector centrality or PageRank), as they are markedly faster than the indirect trust measures for all network sizes considered in these experiments. And, if the platform manager seeks a

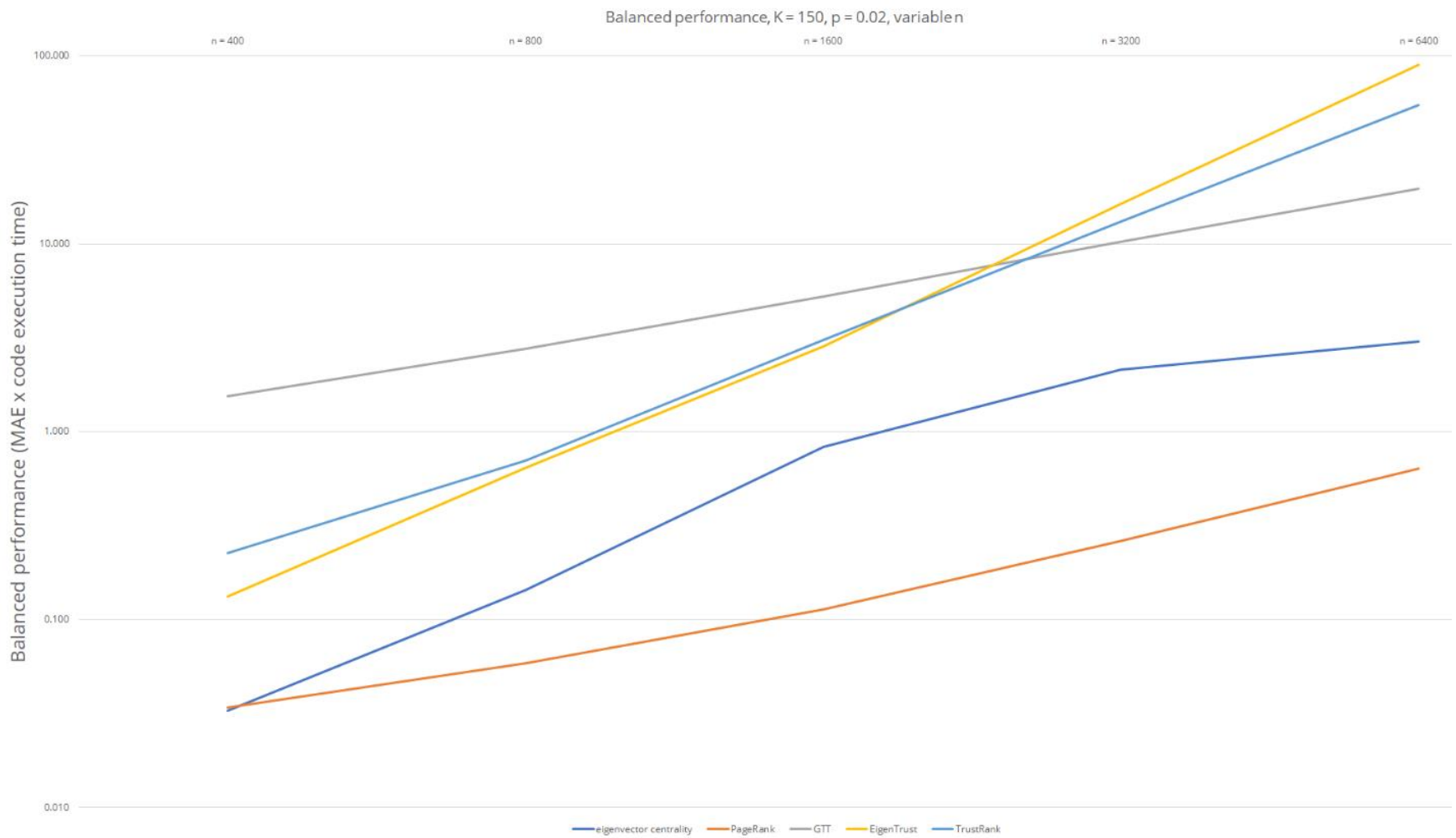
balance between performance and speed, they might opt for either of the direct trust measures, or an indirect trust measure similar to our GTT or TrustRank.



**Figure 10:** Performance of trust metrics as measured by Mean Absolute Error (MAE) for varying sizes of graphs in Experiment Set 1.2.



**Figure 11:** Code execution time (in seconds) for different trust measures in Experiment Set 1.1, for varying sizes of Watts-Strogatz small-world networks. Semilog scale.



**Figure 12:** Ratio of execution time to MAE of trust metrics for varying sizes of graphs in Experiment Set 1.1. Semilog scale.

## Experiment Set 1.2

Experiment Set 1.2 holds  $K$  and  $p$  constant for varying  $n$ , using the  $G_{n,p}$  random graph generative model.

Referring to Figure 13 we see the accuracy of each trust metric when applied to the random graphs generated for Experiment Set 1.2. The first thing to note is that accuracy is significantly higher compared to when they are applied to small world networks for both direct trust measures, as well as for EigenTrust. Accuracy of GTT and TrustRank are similar to the accuracies they demonstrate when applied to small world networks. In the case of GTT, this is expected, since GTT is designed for use in small world networks in mind. We also observe that accuracy remains relatively stable for various network sizes, but GTT's accuracy does appear to improve with the largest network size. TrustRank's accuracy in these experiments is barely better than random chance and, in the case of the largest random graph ( $n = 6400$ ), it is actually lower than for random chance.

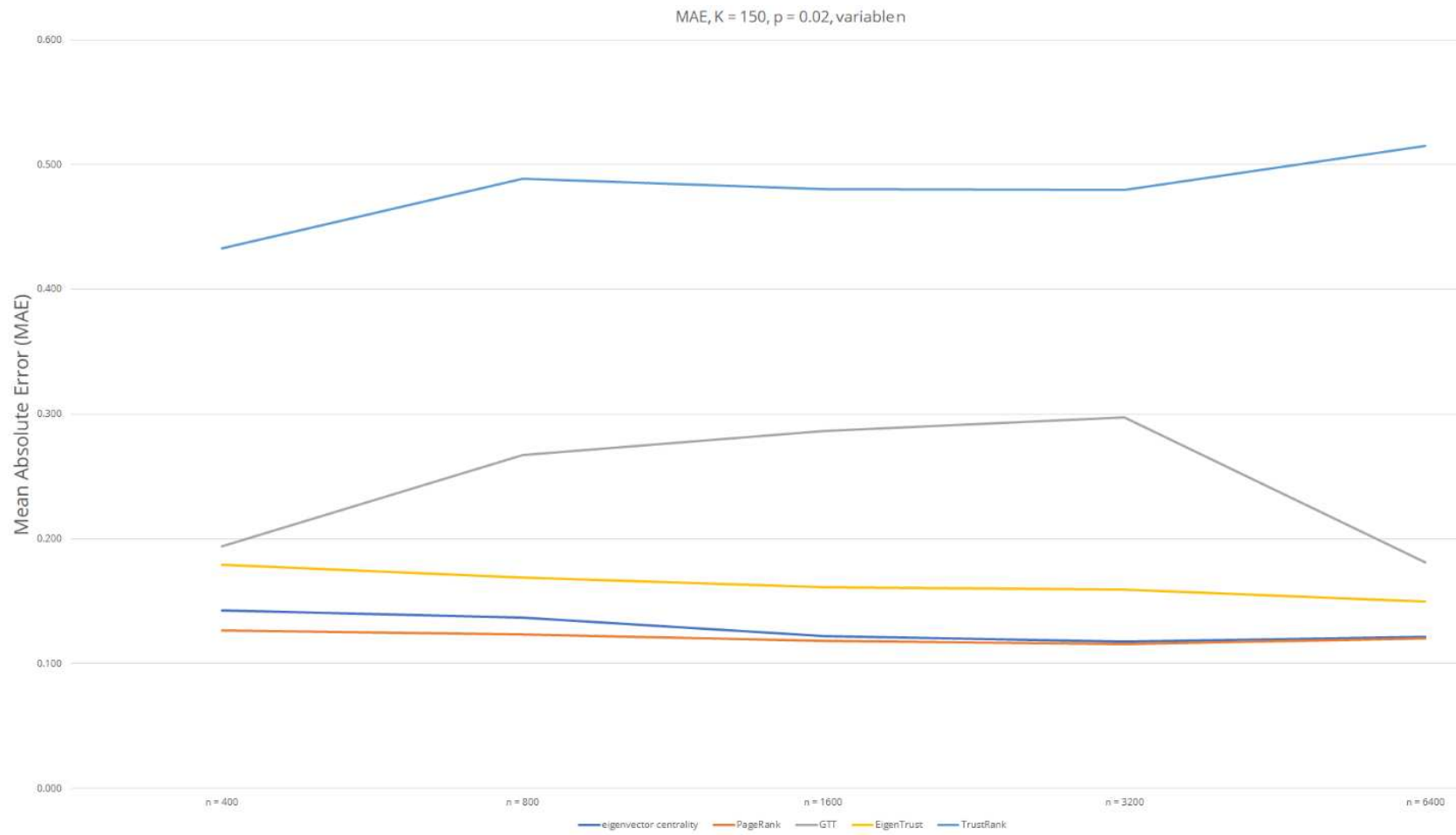
Referring to Figure 14 we see code execution times for each trust metric for each of the experiments that form this set (y axis is logarithmic scale). For all trust measures we observe an increase in code execution times as the size of the networks they are applied to grows. Nonetheless, there are important differences in the rate of increase in code execution speed. The code execution times grow exponentially with the size of the network for EigenTrust, TrustRank, and PageRank. In the case of EigenTrust and TrustRank, code execution times exceed those of GTT by the time we reach the second-largest and largest versions of the graphs in our experiment sets. Code execution speeds for eigenvector

centrality increase more slowly than the previously-mentioned metrics, and code execution speed increases most slowly for GTT.

Figure 15 presents the balanced performance of each metric, which takes into account both accuracy and code execution time; a lower number indicates better overall balanced performance. Eigenvector centrality delivers the best overall Balanced Performance, while the Balanced Performance of GTT increases at the slowest rate, which indicates that for network sizes much larger than the ones analyzed in this chapter it would provide the best balance between accuracy and code execution time. For the other trust metrics, because their MAE remains approximately constant with growing network size and their code execution times increase with growing network size, their balanced performance also degrades as the network grows.

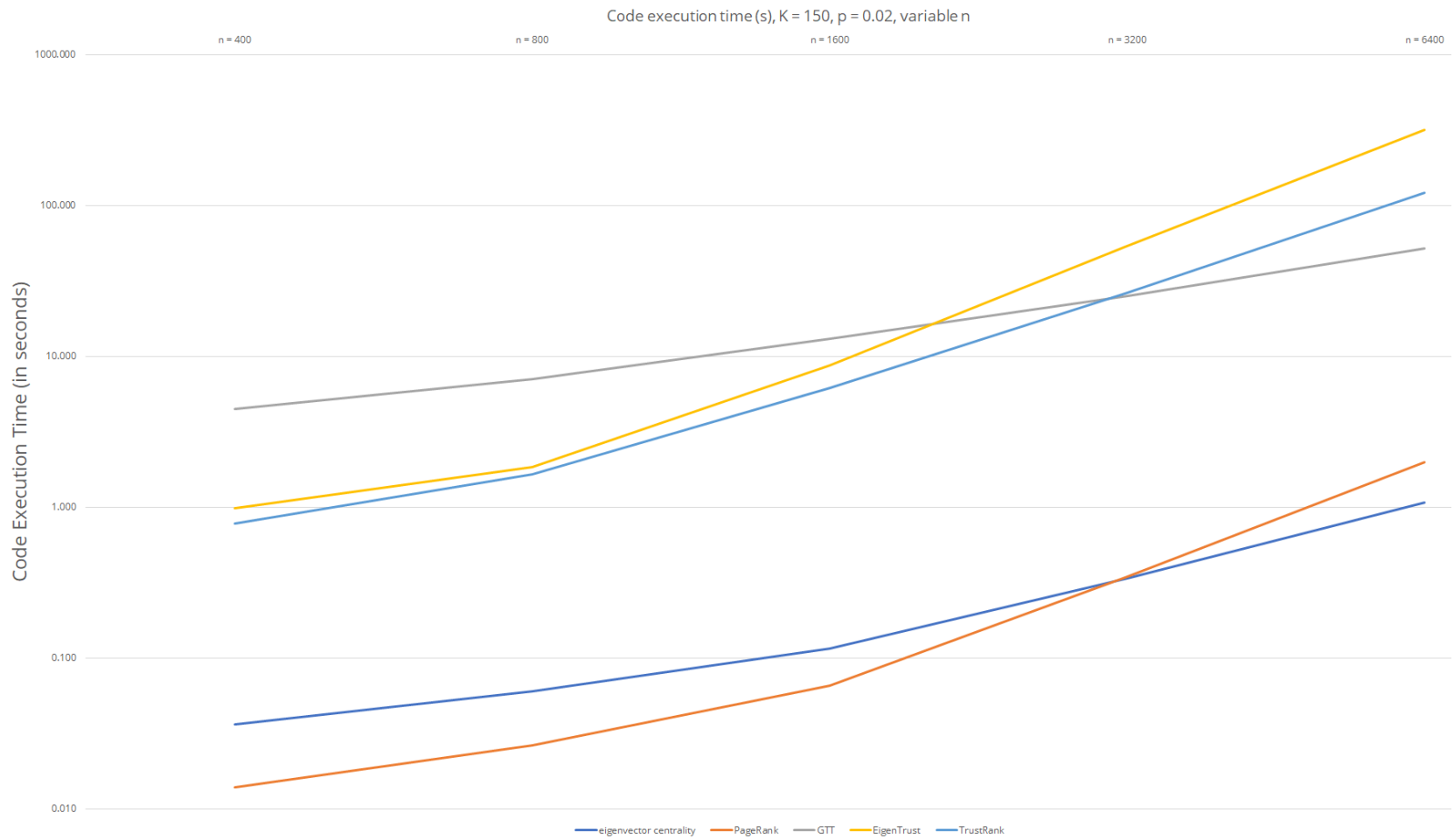
**Table 8:** MAE, code execution time, and Balanced Performance for different trust measures in Experiment Set 1.2.

	eigenvector centrality			PageRank			GTT			EigenTrust			TrustRank		
	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined
n = 400	0.143	0.036	0.005	0.126	0.014	0.002	0.194	4.499	0.873	0.179	0.982	0.176	0.433	0.783	0.339
n = 800	0.137	0.060	0.008	0.123	0.026	0.003	0.267	7.029	1.878	0.169	1.834	0.310	0.489	1.646	0.805
n = 1600	0.122	0.115	0.014	0.118	0.065	0.008	0.287	13.091	3.752	0.161	8.713	1.405	0.480	6.142	2.949
n = 3200	0.118	0.337	0.040	0.115	0.344	0.040	0.298	25.092	7.470	0.159	53.627	8.532	0.480	26.447	12.683
n = 6400	0.121	1.076	0.130	0.120	1.993	0.239	0.181	52.074	9.447	0.150	317.569	47.513	0.515	121.652	62.617

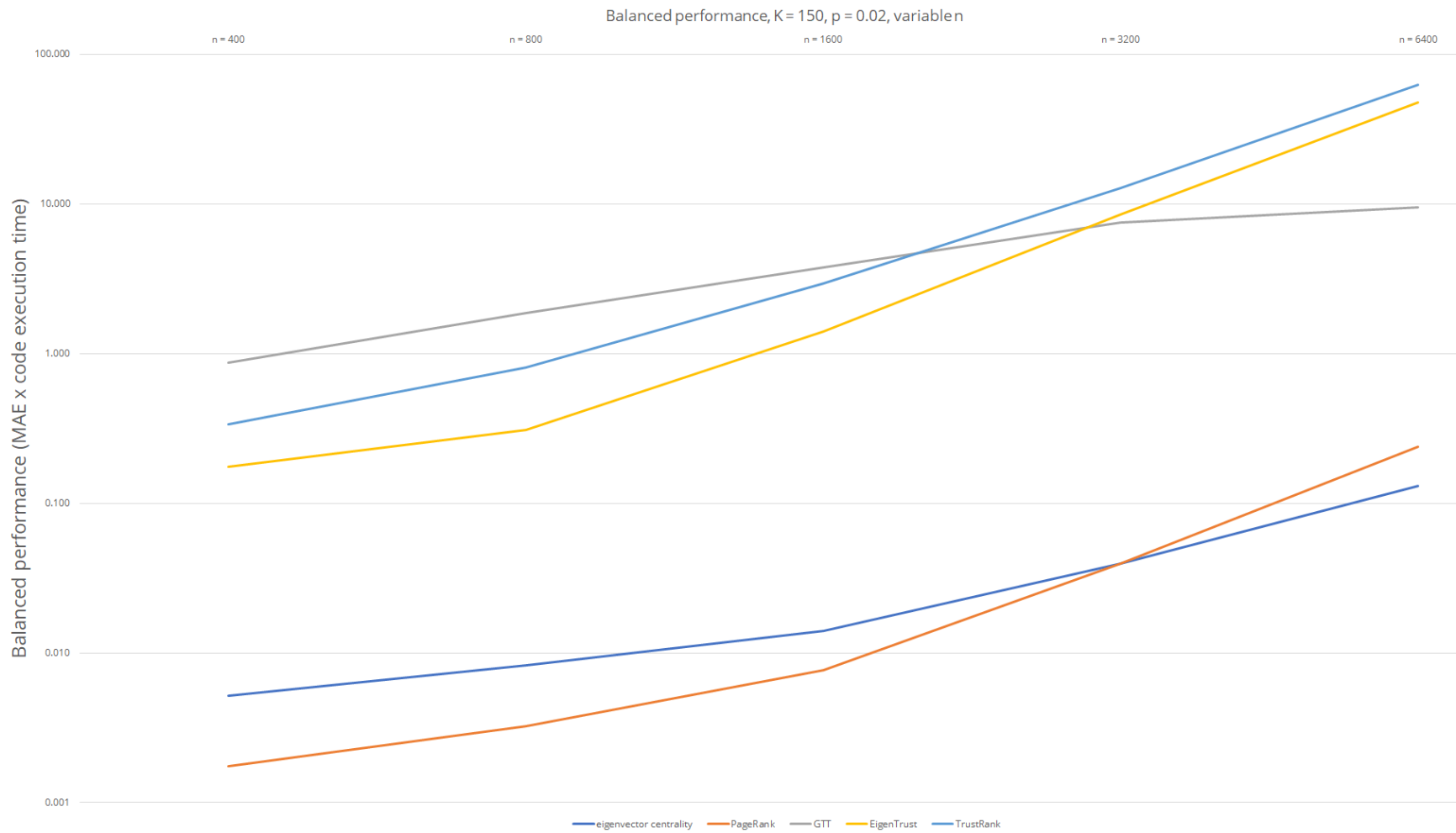


**Figure 13:** Performance of trust metrics as measured by Mean Absolute Error (MAE) for varying sizes of graphs in Experiment Set 1.2.





**Figure 14:** Code execution time (in seconds) for different trust measures in Experiment Set 1.2, for varying sizes of Watts-Strogatz small-world networks. Semilog scale.



**Figure 15:** Ratio of execution time to MAE of trust metrics for varying sizes of graphs in Experiment Set 1.2. Semilog scale.

## Comparing results of Experiment Set 1.1 and Experiment Set 1.2

Comparing code execution times for small world graphs and  $\mathbf{G}_{n,p}$  graphs of the same size, we observe that for the direct trust measures (eigenvector centrality and PageRank) and for one of the indirect trust measures (generic transitive trust, or GTT) the code execution times are faster in the  $\mathbf{G}_{n,p}$  networks than in the small world networks for all network sizes. For both direct trust measures, code execution times only grow slightly even as the size of the network doubles. For GTT, the code execution time grows approximately linearly with the size of the network; a doubling in the network size results in an approximate doubling in code execution time for GTT. On the other hand, for the other two indirect trust measures (EigenTrust and TrustRank), we observe faster code execution times in the small world networks than in the  $\mathbf{G}_{n,p}$  networks, for graphs of all sizes. However, unlike for GTT and the direct trust measures, the speed of code execution for both of these indirect trust measures grows exponentially with growth in network size; code execution times more than double as the size of the network doubles. This indicates a possible limitation of their usefulness in very large graphs.

Examining the accuracy of the trust measures (as measured by Mean Absolute Error, or MAE), we also notice interesting results. First, our experiments show that PageRank delivers the highest accuracy (lowest MAE) for both small world and  $\mathbf{G}_{n,p}$  networks, followed very closely eigenvector centrality. After the direct trust measures, we find a virtual tie in accuracy between GTT and EigenTrust for small world networks, with GTT demonstrating slightly better accuracy than EigenTrust. However, in the case of  $\mathbf{G}_{n,p}$

networks EigenTrust performs significantly better than GTT. This result is expected, since GTT is designed specifically for use in small world networks (as described previously in this chapter). For both small world and  $\mathbf{G}_{n,p}$  networks, TrustRank demonstrates the lowest accuracy (highest MAE), with MAE barely better than guessing for the smaller network sizes, and worse than guessing for the largest network size considered in these experiments,  $n = 6400$ . For all trust measures except TrustRank, we also observe a significant difference in accuracy when applying the trust measures to a small world graph and a  $\mathbf{G}_{n,p}$  random graph. In the case of TrustRank, we observe little difference in accuracy when applied to a small world graph compared to a  $\mathbf{G}_{n,p}$  graph. Accuracy is degraded by approximately half for both direct trust measures (eigenvector centrality and PageRank) as well as for EigenTrust in the small world graph compared to the  $\mathbf{G}_{n,p}$  random graph. For GTT, accuracy is degraded by a smaller amount (approximately one-third) when applied to the small world graph compared to the  $\mathbf{G}_{n,p}$  graph.

Finally, when considering a balanced measure that accounts both for accuracy and code execution time, we observe similar trends as reported in the previous sections. For all measures except for TrustRank, we observe lower performance in the  $\mathbf{G}_{n,p}$  networks than in the small world networks, with little difference between the two for TrustRank. For small world networks, PageRank demonstrates the best balance between accuracy and speed, followed by eigenvector centrality, GTT, TrustRank, and EigenTrust. For  $\mathbf{G}_{n,p}$  random graphs, eigenvector centrality features the best balance between accuracy and speed, followed by PageRank, GTT, EigenTrust, and TrustRank.

## Experiment Set 2

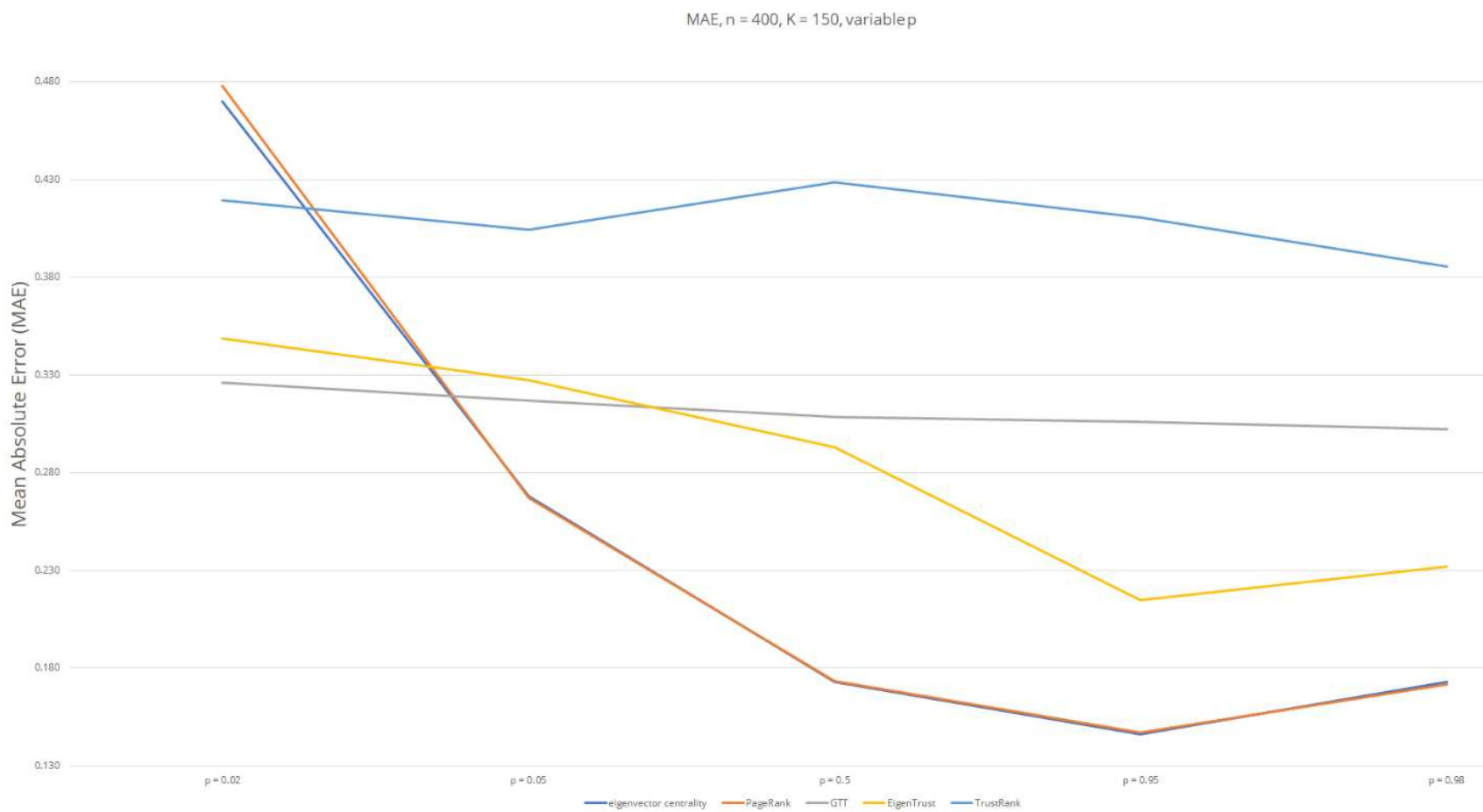
In this section, we present the results of Experiment Set 2. Experiment Set 2 took  $p$ , (probability of rewiring edges in the Watts-Strogatz small world generative model) as the variable, while holding  $n$  constant at 400 and  $K$  constant at 150. In this experiment set,  $p$  varied from 0.02 to 0.98.

In Figure 16 we see the MAE for each trust measure included in this chapter. We observe that, generally, MAE decreases (i.e., accuracy increases) for all trust metrics as  $p$  increases. The exception to this general statement is that for both direct trust measures as well as for EigenTrust, MAE appears to hit a local minimum at  $n = 0.95$  and then begins to increase for  $p = 0.98$ . These results may or may not be anomalous, and should be further explored with larger network sizes in future experimental work.

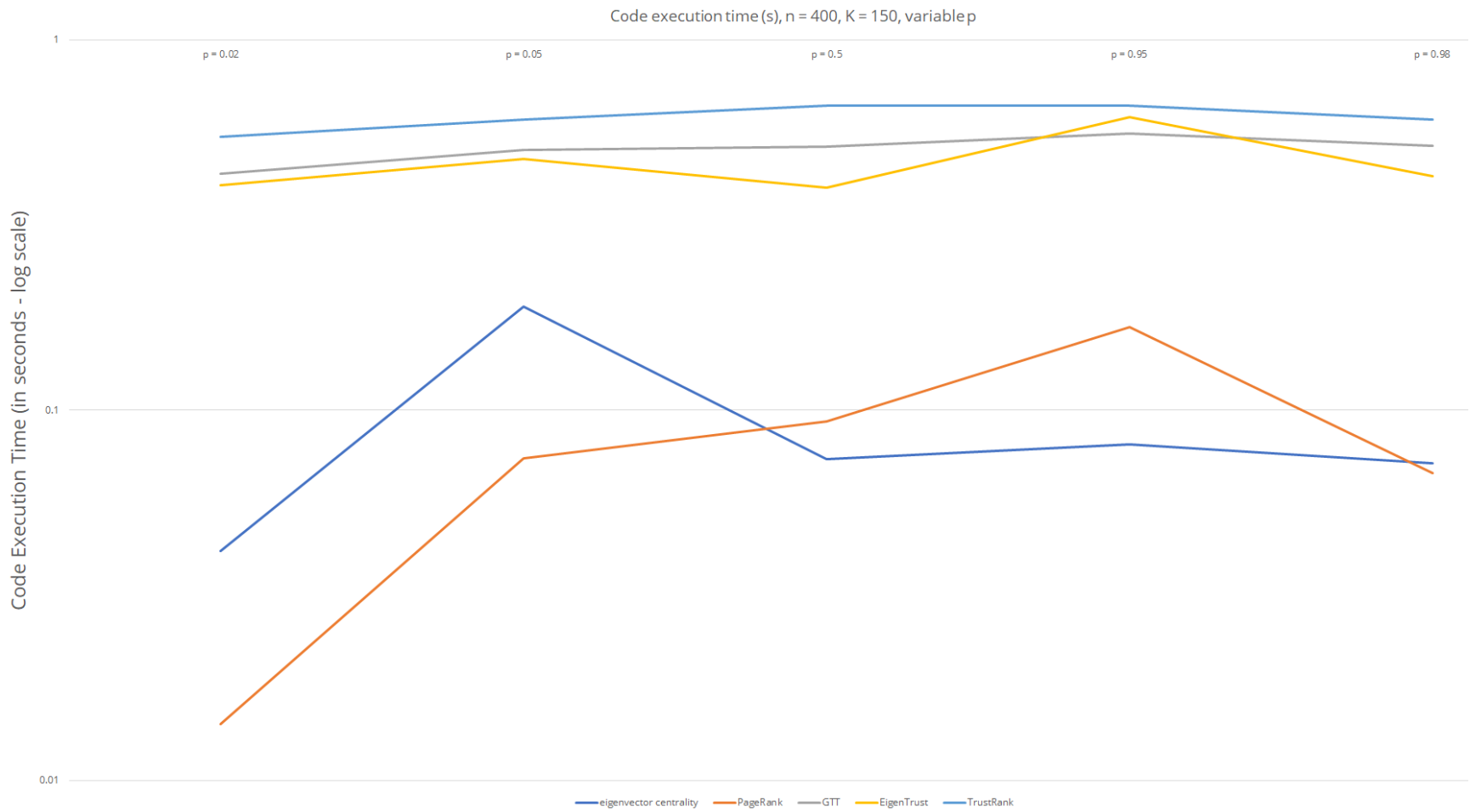
In Figure 17 we present code execution times with a semilog scale (Y axis is logarithmic scale) for each trust metric as  $p$  varies. We observe that for each of the indirect trust measures, code execution time remains relatively steady for all  $p$ , while code execution time increases for the direct trust measures as  $p$  increases.

**Table 9:** MAE, code execution time, and combined performance (MAE  $\times$  execution time) for Experiment Set 2. For all experiments,  $n = 400$  and  $K = 150$ , while  $p$  varies

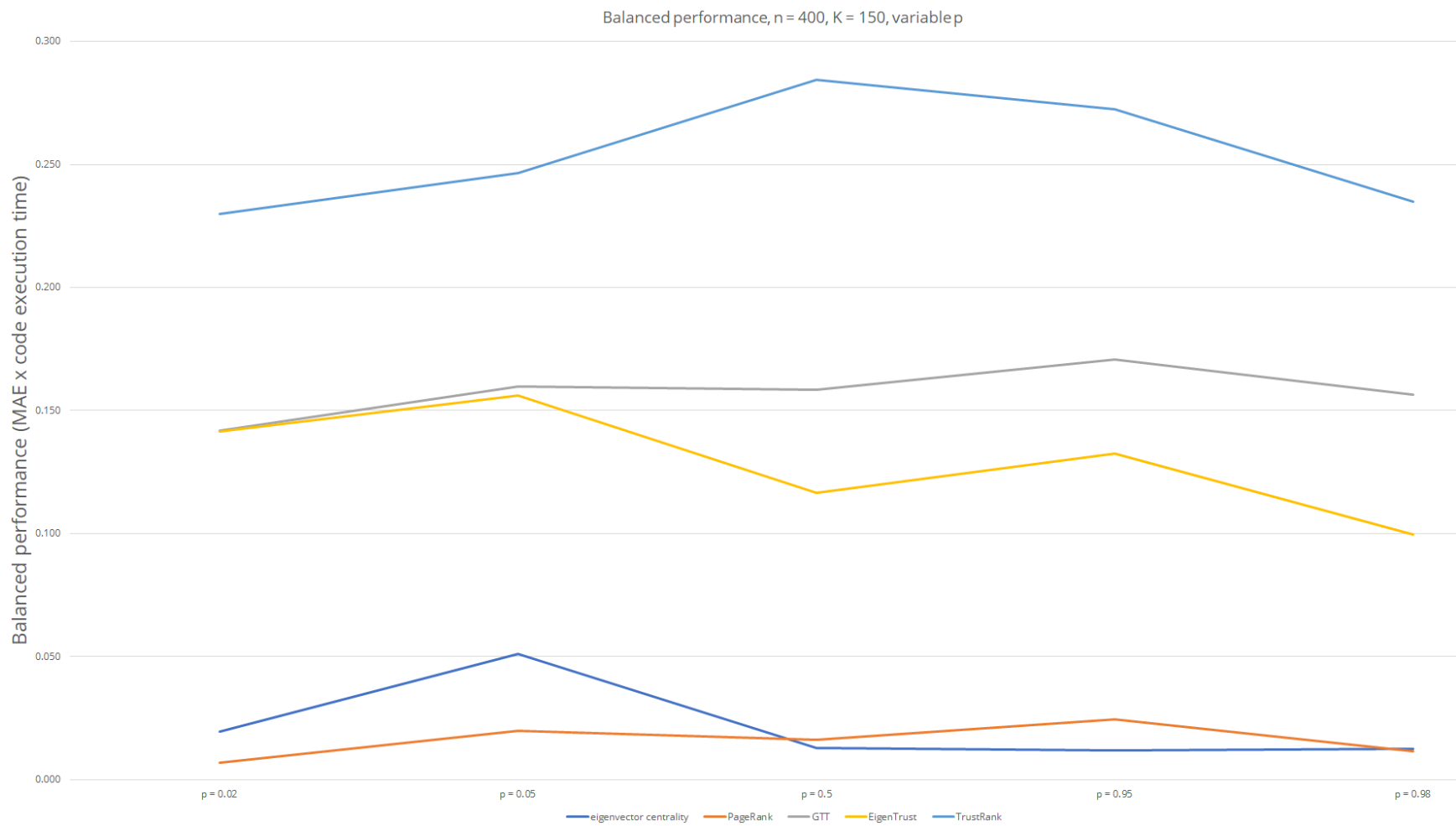
	eigenvector centrality			PageRank			GTT			EigenTrust			TrustRank		
	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined
$p = 0.02$	0.470	0.042	0.020	0.478	0.014	0.007	0.326	0.435	0.142	0.349	0.406	0.141	0.419	0.548	0.230
$p = 0.05$	0.268	0.191	0.051	0.267	0.074	0.020	0.317	0.504	0.160	0.327	0.478	0.156	0.404	0.610	0.247
$p = 0.5$	0.173	0.074	0.013	0.174	0.093	0.016	0.308	0.514	0.159	0.293	0.398	0.117	0.428	0.664	0.285
$p = 0.95$	0.146	0.081	0.012	0.147	0.167	0.025	0.306	0.558	0.171	0.215	0.618	0.133	0.411	0.664	0.273
$p = 0.98$	0.173	0.072	0.012	0.172	0.068	0.012	0.302	0.518	0.156	0.232	0.429	0.100	0.385	0.609	0.235



**Figure 16:** MAE of each trust metric when applied to the random graphs of Experiment Set 2.



**Figure 17:** Code execution time (in seconds) for different trust measures in Experiment Set 2. Semilog scale.



**Figure 18:** Balanced performance (MAE x code execution time) for different trust measures in Experiment Set 2.



### Experiment Set 3

In this section, we present and interpret the results of Experiment Set 3. Experiment Set 3 took  $K$ , (the number of nearest neighbors during the initial graph generation phase of the small world graph generative model) as the variable, while holding  $n$  constant at 400 and  $p$  constant at 0.02. In this experiment set,  $K$  varied from 50 to 250 in increments of 50.

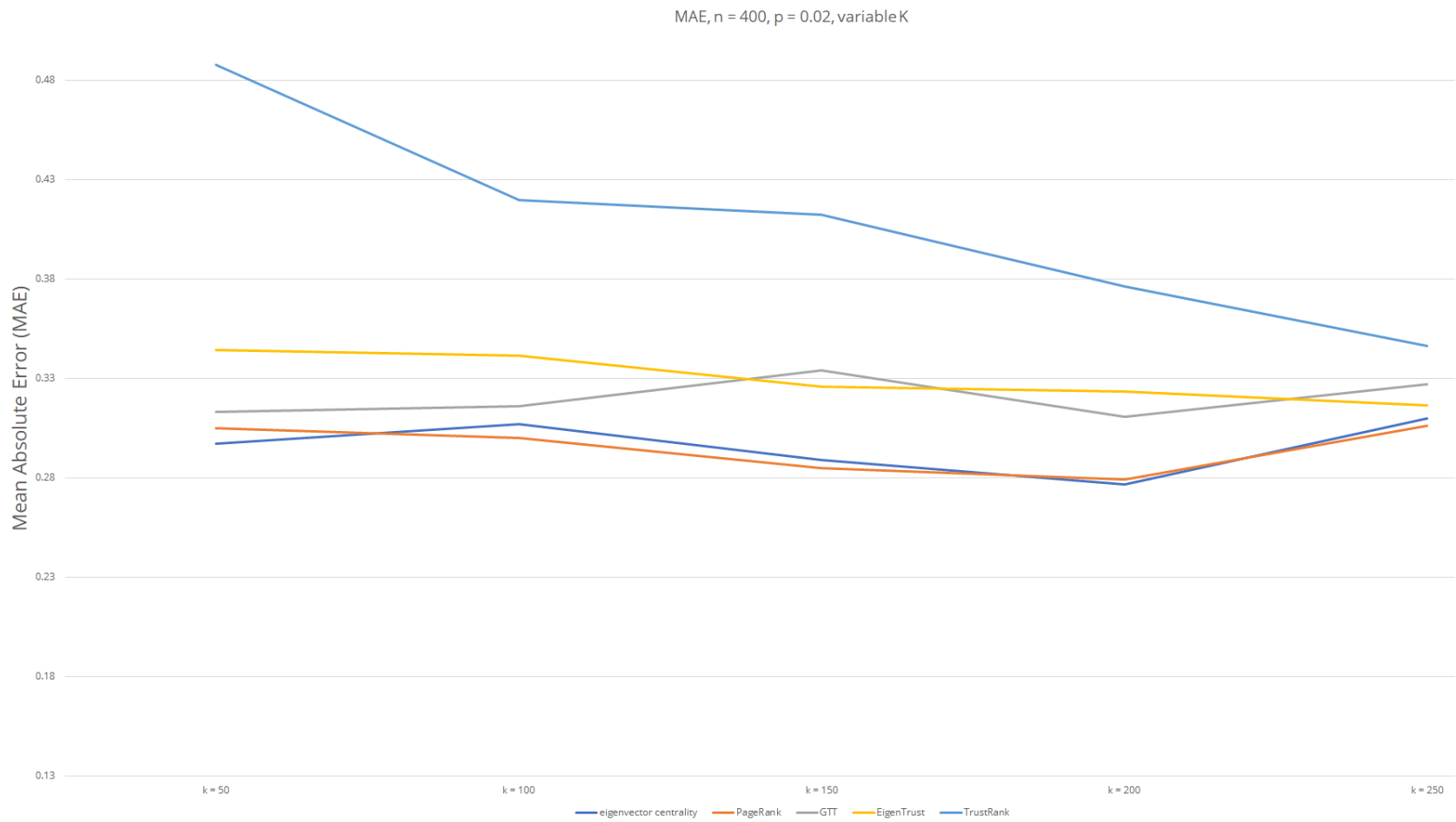
**Table 10:** MAE, code execution time, and combined performance (MAE x execution time) for Experiment Set 3. For all experiments,  $n = 400$  and  $p = 0.02$ , while  $K$  varies in increments of 50, from 50 to 250.

	eigenvector centrality			PageRank			GTT			EigenTrust			TrustRank		
	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined
k = 50	0.297	0.829	0.246	0.305	0.052	0.016	0.313	1.150	0.360	0.344	0.736	0.254	0.488	0.723	0.352
k = 100	0.307	0.553	0.170	0.300	0.283	0.085	0.316	0.829	0.262	0.341	0.748	0.255	0.420	0.706	0.296
k = 150	0.289	0.113	0.033	0.285	0.120	0.034	0.334	0.435	0.145	0.326	0.406	0.132	0.412	0.839	0.346
k = 200	0.277	0.325	0.090	0.279	0.249	0.069	0.311	0.899	0.279	0.323	0.772	0.250	0.376	0.814	0.306
k = 250	0.310	0.255	0.079	0.306	0.339	0.104	0.327	0.901	0.294	0.316	0.436	0.138	0.346	0.483	0.167

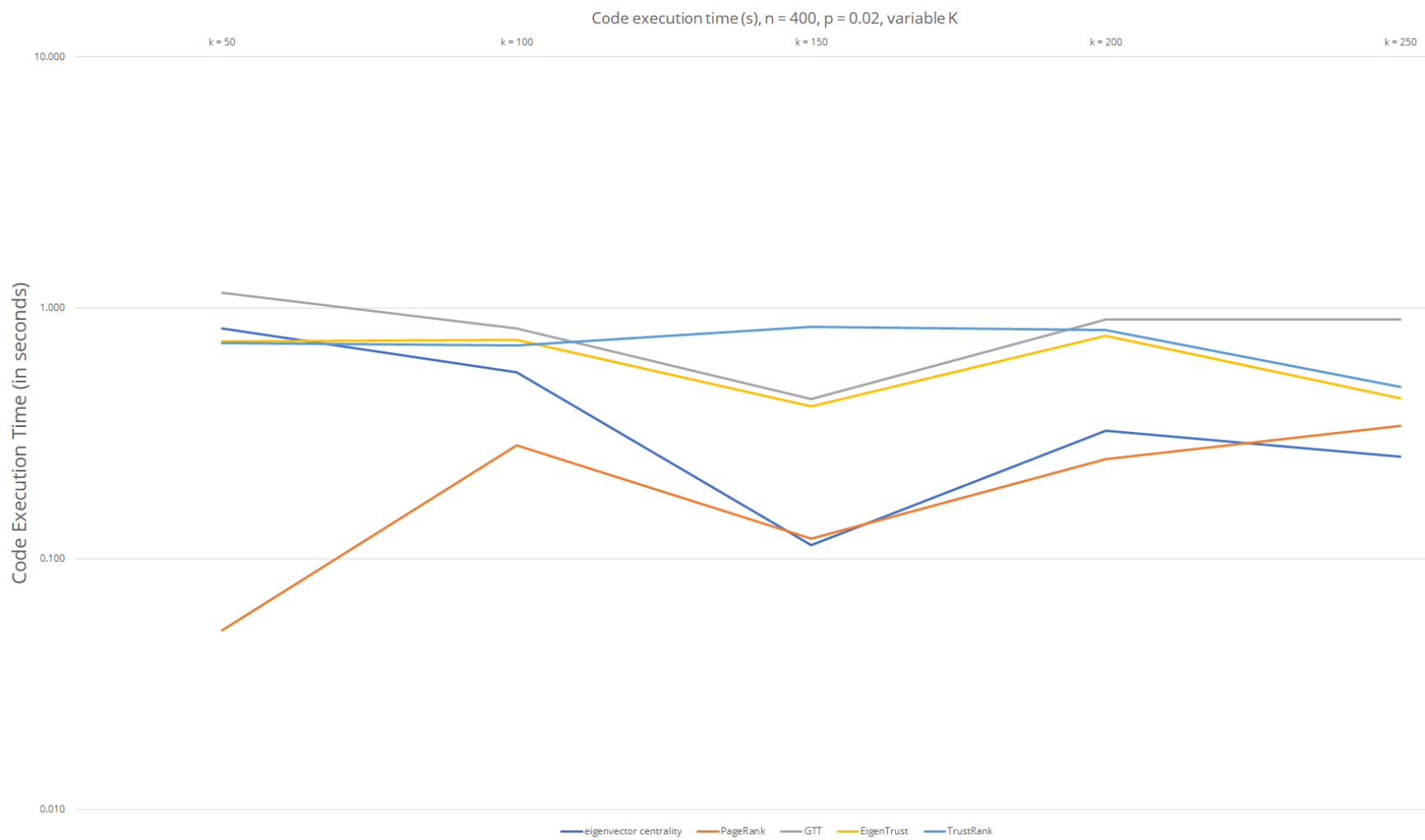
Figure 19 presents the MAE for each trust metric used in this chapter. We see that, with only minor fluctuations, the general trend is for MAE to decrease with increasing  $K$  in small world networks. Accuracy of eigenvector centrality and PageRank are remarkably similar to one another for all  $K$ . For small  $K$ , GTT is the most accurate indirect trust metric followed closely by EigenTrust. For larger  $K$ , EigenTrust is the most accurate of the three indirect trust measures, followed closely by GTT, and TrustRank. Importantly, we observe drastically improved accuracy of TrustRank for higher  $K$  compared to lower  $K$ . We posit that this is an artifact of the metric's design, since it was originally developed for application to Web pages, which often have a large number of links to other pages.

Figure 20 presents code execution times for each trust metric for this experiment set. We observe approximately constant code execution times for each trust metric regardless of  $K$  (with the exception of minor fluctuations which are most likely explained by

random differences in performance of the system environment used to perform the experiments). However, an interesting dip in execution times for  $K = 150$  for all metrics except TrustRank, with code times increasing for smaller and larger  $K$  values surrounding  $K = 150$ . We re-ran the experiments several times for  $K = 150$  to check if this was a fluke of our environment or some other external factor, but each time the experiments continued to return similar results for  $K = 150$ .



**Figure 19:** MAE of each trust metric when applied to the random graphs of Experiment Set 3.



**Figure 20:** Code execution time (in seconds) for different trust measures in Experiment Set 3.

## Correlation between performance and graph structural characteristics

In this section, we present and analyze correlations between the accuracy as measured by MAE and code execution speeds on one hand, and different graph characteristics (size, random graph generative model, probability of edge rewiring in small world graphs, and number of nearest neighbors in small world graphs) on the other hand, for each trust metric evaluated in this chapter. Table 11 presents the correlations between these measures for each of the five trust measures (two direct, three indirect) evaluated as part of this chapter; positive correlation coefficients  $>0.50$  are highlighted in green, and negative correlation coefficients  $<-0.50$  are highlighted in red. In the following subsections, we provide additional discussion and interpretation of these results for each of the trust metrics.

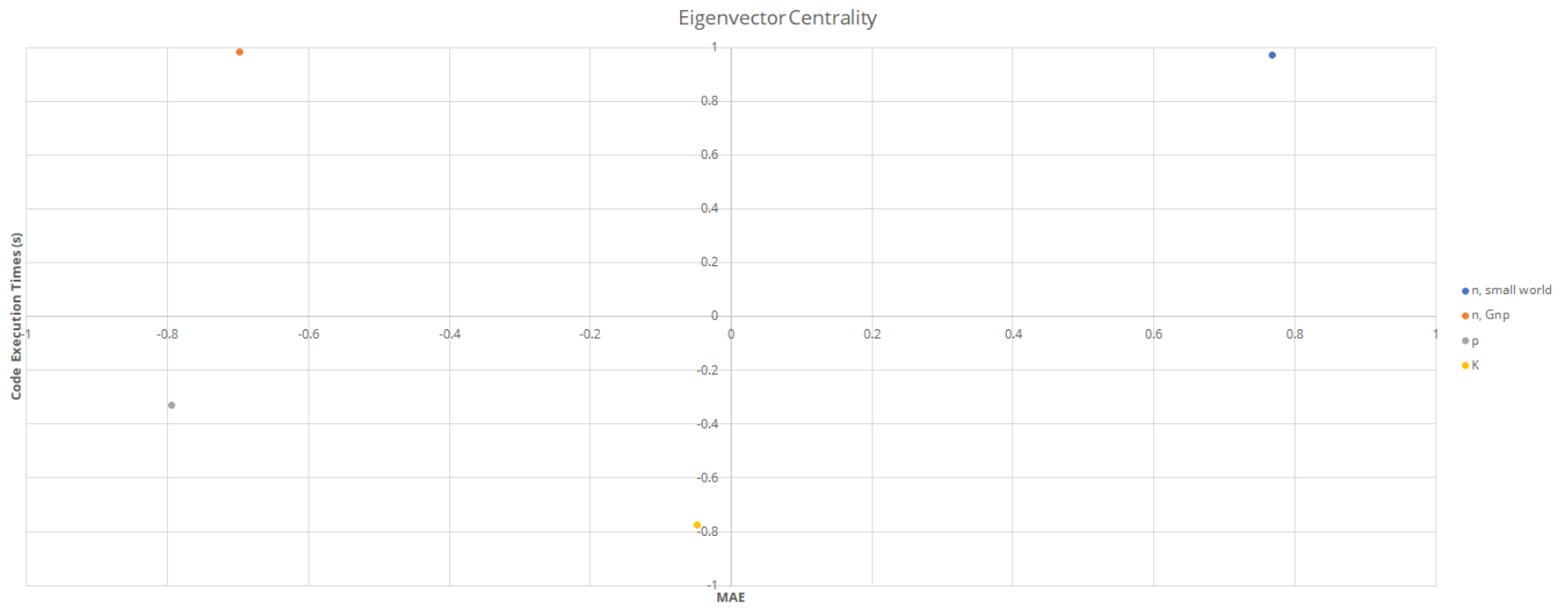
**Table 11:** Correlations between graph structural characteristics and performance of trust metrics

	eigenvector centrality			PageRank			GTT			EigenTrust			TrustRank		
	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined	MAE	time (s)	combined
correlation with network size, $n$ in small world networks	0.768	0.970	0.969	0.646	0.995	0.995	-0.923	1.000	1.000	-0.711	0.954	0.958	0.637	0.971	0.972
correlation with network size $n$ in $G_{n,p}$ random graphs	-0.697	0.982	0.980	-0.502	0.954	0.952	-0.341	0.999	0.955	-0.899	0.954	0.957	0.759	0.966	0.963
correlation with $p$ in small world networks	-0.794	-0.328	-0.674	-0.790	0.655	0.344	-0.917	0.771	0.620	-0.982	0.441	-0.752	-0.441	0.597	0.314
correlation with $K$ in small world networks	-0.048	-0.775	-0.775	-0.239	0.719	0.702	0.362	-0.263	-0.230	-0.970	-0.501	-0.580	-0.969	-0.417	-0.762
correlation with $L$ geodesic path length in small world networks	0.148	0.912	0.911	0.445	-0.694	-0.666	-0.460	0.538	0.517	0.926	0.522	0.598	0.959	0.183	0.595
correlation with clustering coefficient in small world networks	0.004	-0.763	-0.760	-0.185	0.723	0.710	0.385	-0.249	-0.212	-0.968	-0.528	-0.604	-0.965	-0.464	-0.792

### *Eigenvector centrality*

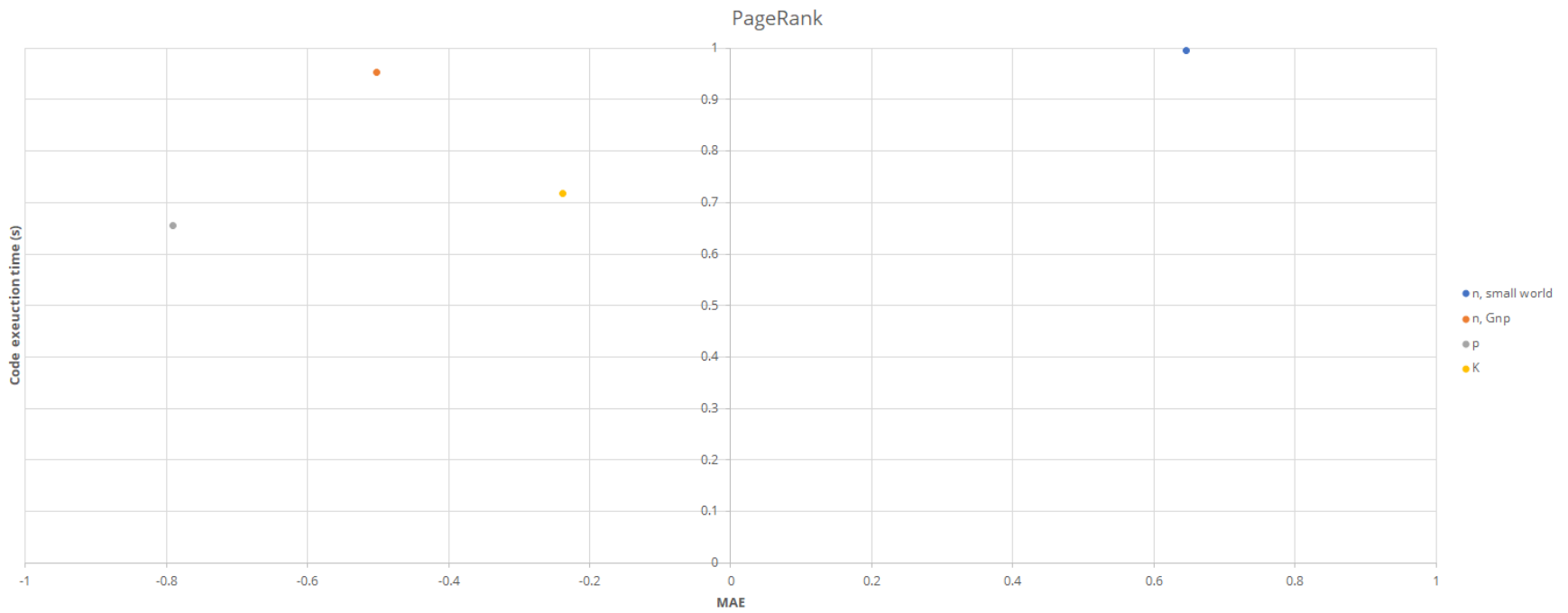
When eigenvector centrality is applied to small world graphs for estimating trust values, MAE varies inversely with increasing  $\rho$  and increasing  $K$ . That is, as the probability of edge rewiring ( $\rho$ ) in a small world graph increases, eigenvector centrality becomes more accurate (lower MAE) in approximating ground truth trust values. And, as the number of nearest neighbors of nodes ( $K$ ) in a small world graph increases, eigenvector centrality becomes more accurate (lower MAE) in approximating ground truth trust values. On the other hand, for small world graphs, as the size of the graph increases, eigenvector centrality becomes less accurate in estimating ground truth trust values (MAE increases, but at a lower rate than the increase in the size of the graph). When applied to  $G_{n,\rho}$  random graphs, eigenvector centrality becomes more accurate (lower MAE) in estimating ground truth trust values as the size of the network grows.

Considering code execution times, when eigenvector centrality is applied to small world graphs for estimating trust values, code execution time increases as the network size grows. However, as  $\rho$  increases we see a moderate decrease in code execution times, and as  $K$  increases we observe a large decrease in code execution times. Both of these results are expected since a higher  $\rho$  makes the graph more similar to a  $G_{n,\rho}$  graph, and a higher  $K$  increases the density of the graph. When applied to a  $G_{n,\rho}$  graph, code execution speed for eigenvector centrality also increases as the size of the network grows, at approximately the same rate as is observed in small world networks.

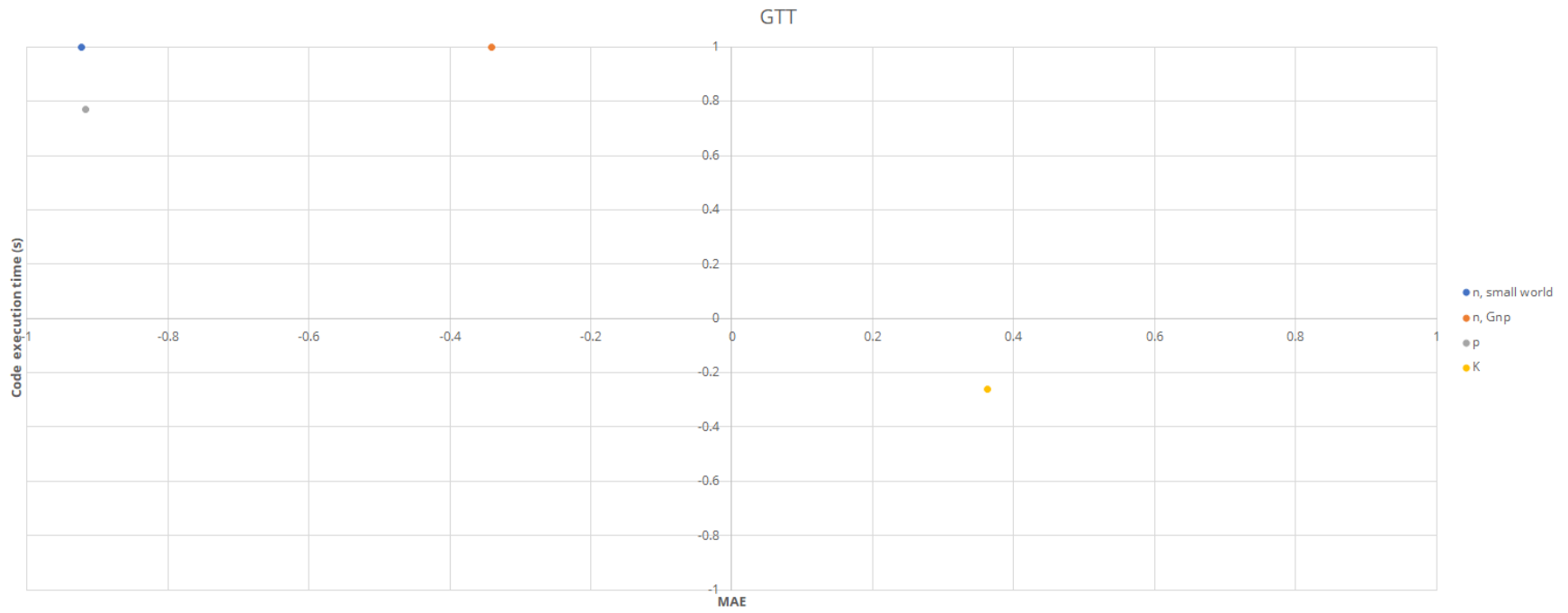


**Figure 21:** Correlation between graph structural characteristics and eigenvector centrality's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).

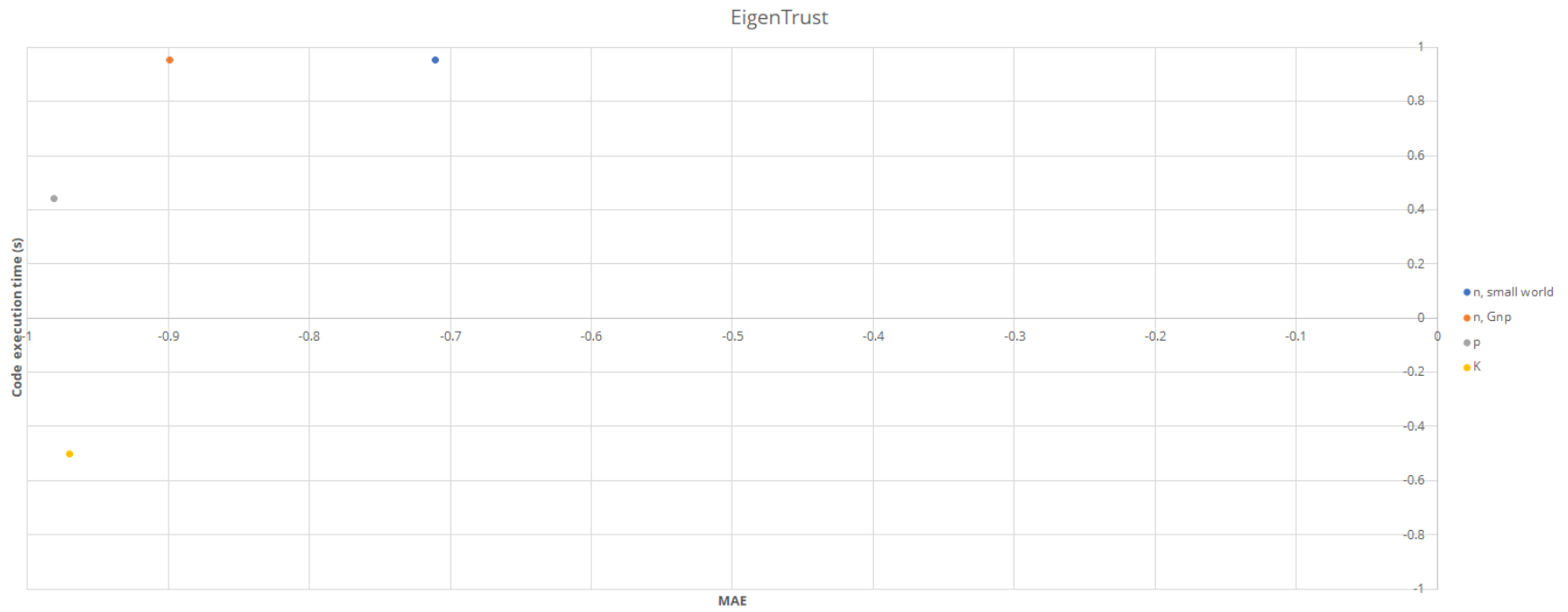




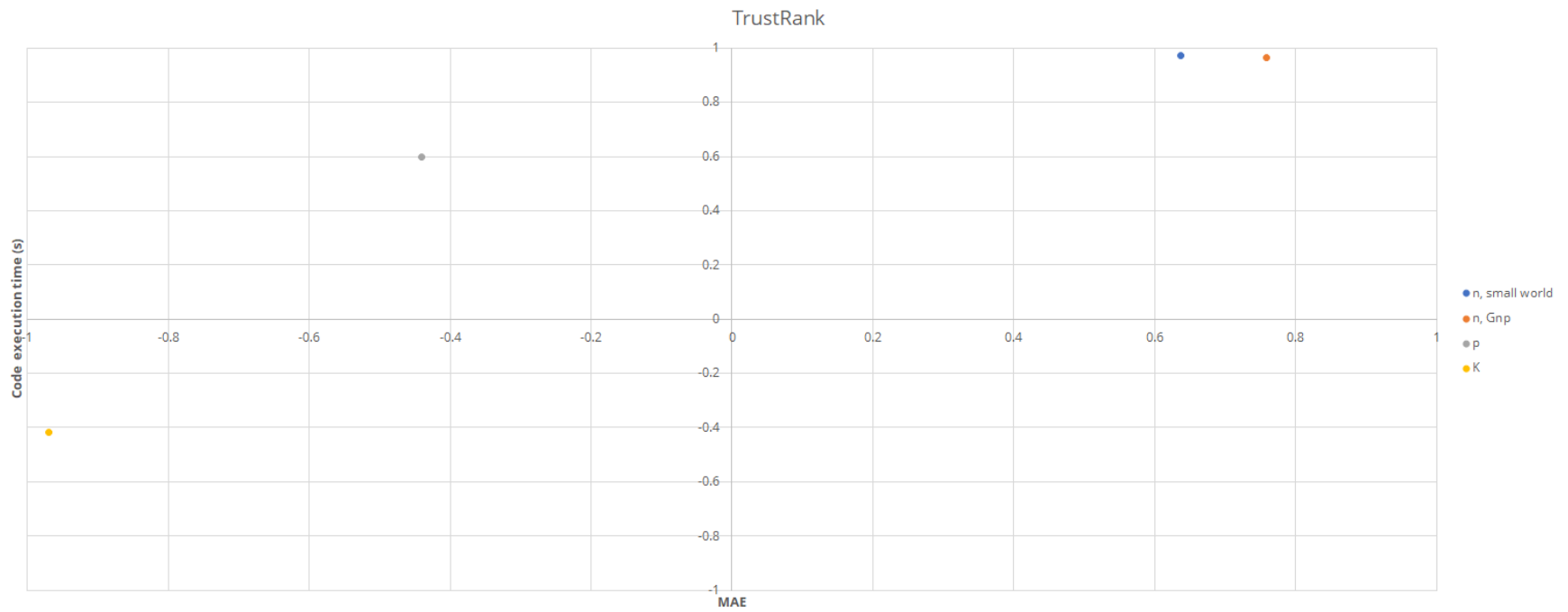
**Figure 22:** Correlation between graph structural characteristics and PageRank's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).



**Figure 23:** Correlation between graph structural characteristics and GTT's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).



**Figure 24:** Correlation between graph structural characteristics and EigenTrust's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).



**Figure 25:** Correlation between graph structural characteristics and TrustRank's performance. X axis values correspond to MAE, Y axis values correspond to code execution time (in seconds).

### *PageRank*

When PageRank is applied to small world graphs for estimating trust values, MAE varies directly with an increase in the size of the network; that is, as the network grows, PageRank becomes progressively less accurate in estimating the ground truth trusts in the network. One key reason for this is because there is recursive small- and mid-size world behavior; in other words, higher-order clustering than "classic" small world. For increasing  $\rho$  and  $K$  values, however, accuracy of PageRank in estimating ground truth trust values increases (MAE decreases). When applied to a  $\mathbf{G}_{n,\rho}$  graph, PageRank becomes more accurate (lower MAE) as the size of the network grows.

When evaluating code execution speeds, there is a direct positive relationship between increasing  $n$ ,  $\rho$ , and  $K$  and code execution speeds in small world networks, and between  $n$  and code execution speeds in  $\mathbf{G}_{n,\rho}$  graphs.

### *Generic Transitive Trust (GTT)*

Different from the two direct trust measures discussed in the immediately preceding sections, for GTT (an indirect trust measure) we observe increasing accuracy in estimating ground truth trust values as network size ( $n$ ) grows for small world and  $\mathbf{G}_{n,\rho}$  graphs, and increasing accuracy as  $\rho$  increases for small world networks. However, in a reversal of the observations from the direct trust measures, we see a decrease in accuracy of GTT (increasing MAE) as  $K$  increases.

Examining code execution speeds, for GTT we see a direct relationship between an increase in code execution speed and a growing network size ( $n$ ) for both small world and  $G_{n,p}$  graphs; we also observe an increase in code execution speeds as  $p$  increases, but at a lower rate. And, as  $K$  increases, we observe an inverse relationship between code execution speed for GTT.

These results are expected, as GTT is specifically designed for application in small world networks, with the tradeoff being increased accuracy resulting in slower code execution speeds.

### *EigenTrust*

Similar to GTT, EigenTrust demonstrates an inverse relationship between increasing network size and MAE, though the improvements to accuracy as the network grows is lower for EigenTrust than for GTT. We also observe an inverse relationship between MAE and an increasing  $p$  in small world networks. Different from GTT, however, in EigenTrust we observe increasing accuracy (decreasing MAE) as  $K$  increases.

When considering code execution speeds, we see a direct positive relationship between code execution speeds and increasing network size ( $n$ ) for small world and for  $G_{n,p}$  graphs, and a positive but more modest relationship between increasing  $p$  and code execution speeds in small world graphs. Similar to GTT, there is an inverse relationship between increasing  $K$  and code execution speeds, but code execution speed decreases faster with EigenTrust with an increasing  $K$  than it does with GTT.

## *TrustRank*

Finally, for TrustRank we see an inverse relationship between accuracy and growing network size in both small world and  $\mathbf{G}_{n,p}$  graphs. As both  $p$  and  $K$  increase, TrustRank becomes more accurate. When considering code execution speeds, we see increasing code execution speeds with TrustRank for increasing network sizes of small world and  $\mathbf{G}_{n,p}$  graphs, and increasing code execution speeds for increasing  $p$ . On the other hand, as  $K$  increases, code execution speed for TrustRank decreases. TrustRank also takes a damping factor (similar to PageRank), which for this chapter's experiments we set to 0.85, which is industry standard for PageRank-like measures.

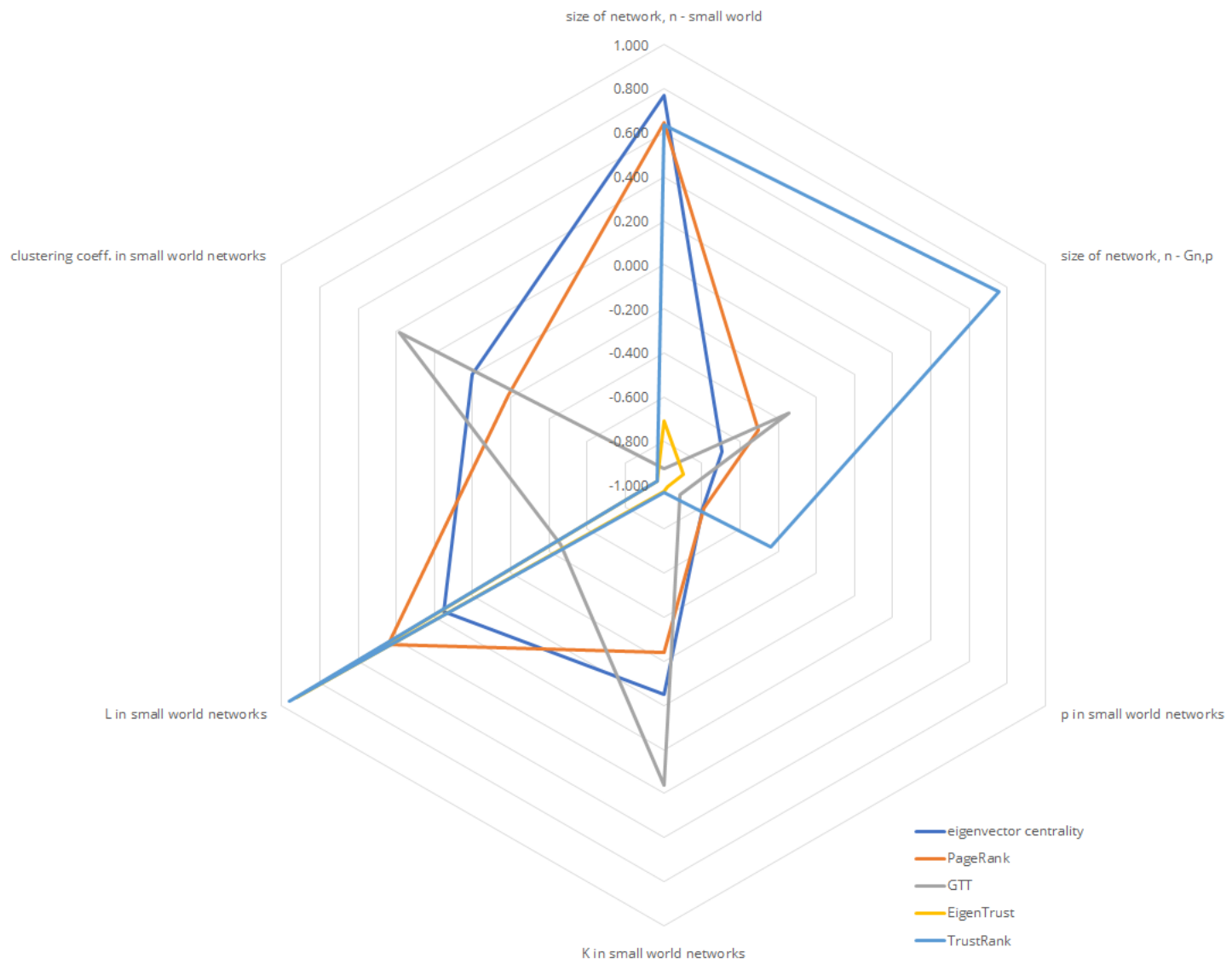
### **Framework for selecting a trust model**

Using the results and analysis from the previous section of this chapter, in this section we propose a conceptual framework to assist practitioners and researchers in selecting a trust model for use in their own applications.

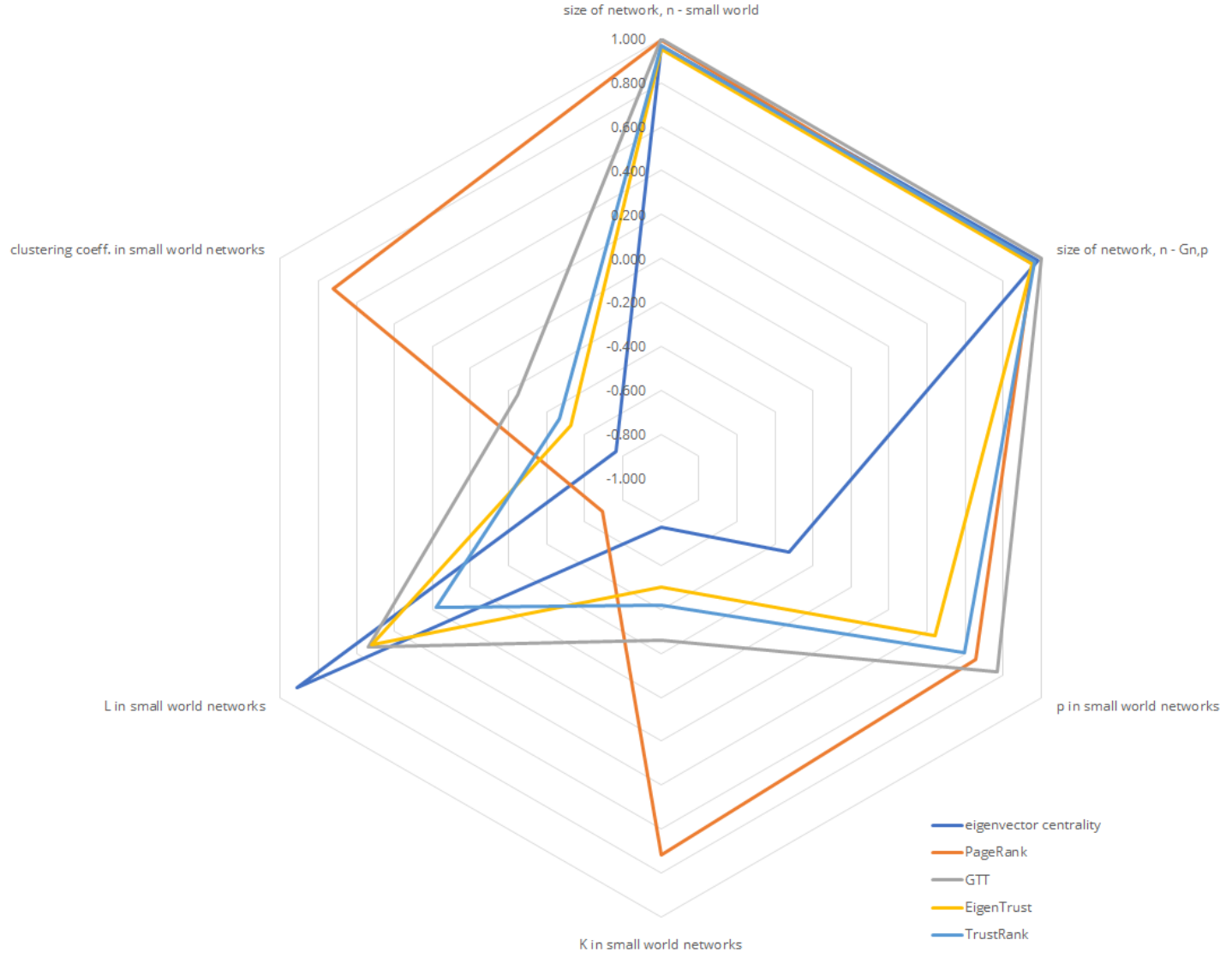
Before presenting our framework, we first present Figure 26, Figure 27, and Figure 28 to provide additional analytical background on how we derived our selection framework. Each of these figures are radar charts which plot how the performance (MAE, code execution speed, and Balanced Performance) are affected by different graph structural characteristics. From Figure 26 we see, for example, that both EigenTrust and GTT demonstrate an inverse relationship between MAE and increasing network size (that is, they become more accurate as the network grows in size). From Figure 27, among other insights, we can observe that code execution speed for eigenvector centrality is strongly

affected by the size of the network, but it is barely affected by changes in path lengths ( $L$ ) or clustering coefficients in small world networks.

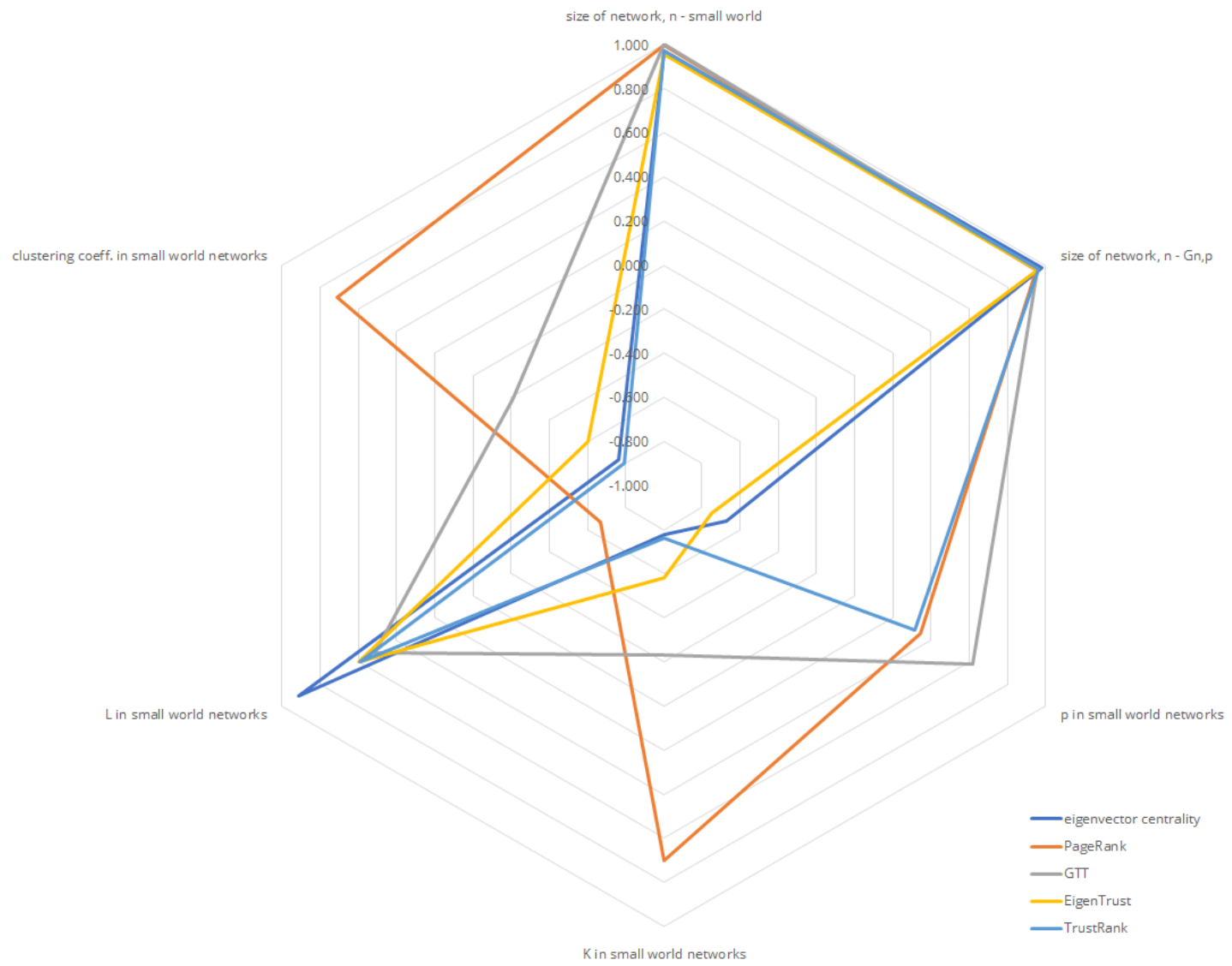




**Figure 26:** Correlation between MAE and graph structural characteristics for trust metrics.



**Figure 27:** Correlation between code execution times and graph structural characteristics for trust metrics.



**Figure 28:** Correlation between Balanced Performance and graph structural characteristics for trust metrics.

We take the specific quantitative insights from Figure 26, Figure 27, and Figure 28 and translate them into a more universal, heuristic framework which we present in Table 12 and Table 13. Given that most real world technological, sociotechnical, and social networks exhibit small worldness [4.24], we present the framework for small world networks first. Table 12 presents heuristic relationships between the expected MAE of a trust metric with respect to the ground truth trust values for the network to which it is applied, and how the MAE is affected by different graph structural characteristics (network size  $n$ ,  $p$ ,  $K$ ,  $L$  geodesic path lengths), and graph clustering coefficients. Table 13 presents the same heuristics for code execution speed.

In the tables, a sign of “+” indicates a mildly positive relationship between the two factors (between 0 and 0.5), a “++” indicates a strongly positive relationship (>0.5) between the two factors, a “-” sign indicates a mildly negative correlation (between 0 and -0.5) between the two factors, a “--” indicates a strongly negative correlation (<-0.5), and a “negligible” entry indicates a relationship that is statistically indistinguishable from 0.

To generalize our results beyond the specific trust metrics evaluated in this chapter, we collapse the trust metrics from this chapter into archetypal categories of trust metric type. Centrality-based trust metrics are those like eigenvector centrality which estimate a node’s trust value using a graph centrality measure. Other examples are degree centrality and betweenness centrality. Centrality feedback metrics are those like PageRank which consider the importance, influence or (in our case) trust of nodes that link to a node. Centrality feedback plus path uses centrality feedback while also considering paths

between nodes within a graph. Finally, path-based metrics rely primarily on the multiple paths (if any) between a sink and a source node to estimate trust. The degree of relevance of each algorithm is clearly network architecture-dependent, with the degree of recursiveness of clustering likely a key driving factor.

As with any design challenge, in examining the framework we see that there are tradeoffs in accuracy and speed when selecting a trust metric to use in an application. Our framework provides a systematic way for thinking about how to make these tradeoffs during design, based on the structural characteristics of the network to which it will be applied.

**Table 12:** Relationship between graph structural characteristics in small world networks and MAE for different categories of trust model

	MAE			
	centrality	centrality feedback	centrality feedback + path	path
network size	++	++	--	--
$p$	--	--	--	--
$K$	neglible	-	--	+
$L$	+	+	++	-
clustering coeff.	neglible	-	--	+

**Table 13:** Relationship between graph structural characteristics and code execution times for different categories of trust model

	Code execution time			
	centrality	centrality feedback	centrality feedback + path	path
network size	++	++	++	++
$p$	-	++	+	++
$K$	--	++	--	-
$L$	++	--	++	++
clustering coeff.	--	++	--	-

### Limitations

Our framework is constrained by several limitations; thus, caution, judgement and interpretation are required when using the framework. First, as was described earlier in this chapter, our experiments only consider global trust, not local trust. This is a useful measure for managers of platforms who have access to relevant proprietary data about the platform, but it is less useful for inferring trust relationships between specific nodes. Second, our framework was derived solely from graph structural characteristics, and thus when considering platforms that have access to other data types such as content or behavior this framework will be incapable of considering these measures. Additionally, our framework doesn't consider all categories of trust models (though it does encapsulate the most popular categories). Our framework is derived from a limited dataset and limited number of experiments, and thus its robustness and sensitivity should be explored further. Finally, the framework only provides heuristic guidance – not specific quantitative predictions – as to the direction and approximate intensity of performance metrics' relationships to graph structural characteristics. Despite these limitations, to the best of

our knowledge our framework represents the first systematic selection framework for a trust metric, based on a quantitative experimental analysis of the relationship between specific graph structural characteristics and corresponding performance of different categories of trust models.

### **Future Work**

To strengthen the findings of this chapter, future work should focus on robustness and sensitivity checks of the various measures investigated. Additional graph structural characteristics could be considered, such as assortativity. Critically, future work should also evaluate differences between performance of trust metrics applied to directed v. undirected graphs; this chapter considered only undirected versions, but inclusion of benchmarking on directed graphs represents an important extension for real world networks. The experiments should be run for more iterations than we did in this chapter (three iterations of each experiment) to further smooth out any differences owing to random factors that may be present in our findings, and to add statistical power to the findings. Additional trust metrics should also be benchmarked; in this chapter, although we selected trust metrics that are representative of the majority of the types of trust metrics that have been designed, there are many more that could have been included. Our experiments only consider global trust values. While these values are useful for managers of platforms who seek to minimize spread of bad information and have access to all data needed to perform these computations, they are less useful for inferring pairwise trust

from one node to another. Thus, future research should also consider how performance of trust metrics changes when considering local, pairwise trust values.



## V. Characterizing the nature of trust and misinformation on Twitter

Owing to the hyperconnected nature of modern businesses, economies, and society we've observed a high and accelerating rate of breaches of security and privacy of user data. A popular cybersecurity model is the CIA Triad. In this model, "C" stands for Confidentiality, "I" stands for Integrity, and "A" stands for Availability. When considering cybersecurity through the lens of the CIA Triad, we can see that cybersecurity incidents don't only have to involve data breaches, ransomware, identity theft, or other cybercrimes.

The online spread of misinformation and disinformation can be considered as a subcategory of cybersecurity threat, as it deals with the "I" – (data) Integrity – of the CIA Triad. Researchers have discovered and described an increasing number and severity of misinformation, disinformation, and misinformation (MDM) threats whose spread is enabled and, in many cases, encouraged by online social network platforms. CISA, the Cybersecurity and Infrastructure Security Agency (an agency of the United States Department of Homeland Security) considers misinformation to be part of the overall cybersecurity threat landscape [5.1].

As online connectivity has grown and as more of our daily interactions shift to taking place online, trust metrics and misinformation in the online realm have both been explored extensively by researchers. Less well-understood to-date is the *overlap of trust with online misinformation networks*.

In this chapter, we seek to develop a stronger understanding of the relationship between trust and misinformation in online social networks.

*Does trust lead to the network structures that we see in an online misinformation network?*

*Or, do misinformation networks lead to the types of trust relationships that we observe?*

*Or is it both, or something else?*

To explore these questions, we utilize an empirical dataset from the Twitter platform regarding a specific subset of misinformation: conspiracy theories related to the COVID-19 pandemic.

## **Introduction**

In 2013, the World Economic Forum (WEF) listed “digital wildfires in a hyperconnected world” as a global risk for policymakers and technologists to address as part of its annual Global Risks Report [5.1]. The WEF report authors stated that *“The global risk of massive digital misinformation sits at the centre of a constellation of technological and geopolitical risks ranging from terrorism to cyber attacks and the failure of global governance. This risk case examines how hyperconnectivity could enable “digital wildfires” to wreak havoc in the real world. It considers the challenge presented by the misuse of an open and easily accessible system and the greater danger of misguided attempts to prevent such outcomes.”*

[5.1] Unfortunately, the WEF report authors were justified in their worry, as we observe an increase in the frequency, severity, and ubiquity of online misinformation across geographies and platforms since 2013.

Many scholars and practitioners today are focused on how to identify misinformation in online social networks (OSNs). The most popular approaches today leverage techniques from natural language processing or deep learning to identify and classify online claims as true or false (misinformation), and to flag them for further treatment. Other approaches make use of crowdsourcing to use the community in flagging and fighting online misinformation. These methods hold great promise, and are already helping to combat the prevalence of online misinformation.

Can these machine learning (ML) misinformation detection models be augmented by layering trust metrics on top of them? All other factors being held equal, it stands to reason that a user is more likely to share, comment on, and ultimately believe a misinformation claim online if they first hear that claim from a trusted connection. If we can understand the characteristics of misinformation trust networks, this may serve as an additional tool in helping to slow spread of the same.

Computer scientists have been developing and implementing trust metrics for more than two decades, as a way to estimate trust among agents in an increasingly-connected world. In this chapter, we use the principles and concepts developed by this rich body of work and apply them to improve our understanding of the nature of trust and misinformation online.

## **Background**

Trust is an inherently subjective and social phenomenon which is difficult to precisely model and quantify computationally. In spite of these challenges, simple, coarse-

grained models of trust have been developed and demonstrated their usefulness in improving the functioning of sociotechnical systems – particularly online social networks (OSNs) (including e-commerce settings), but also distributed computing systems, wireless communications networks, Internet-of-Things (IoT) networks, and other related networks whose primary function is to facilitate transactions among agents.

For more than two decades, scholars in computer science and adjacent fields have developed numerous trust metrics (as reviewed in Chapter III of this dissertation), which are methods for quantitatively estimating trust among nodes in a network. Some trust metrics are designed for use with a specific platform or context in mind, while others are relatively speaking more general purpose. For the purposes of this chapter, we use [5.3]’s definition of trust: *“trust is the belief of one in another, based on their interactions, in the extent to which the future action to be performed by the latter will lead to an expected outcome.”*

By examining this definition, we can see that trust relationships 1) need not be mutual, and 2) when one agent trusts another, it doesn’t necessarily imply anything else about that relationship. For example, on Twitter if A follows B, it is an explicit expression of A trusting in B within the context of Twitter; A trusts that B will have interesting, entertaining, or otherwise worthwhile things to say. B need not follow A back in order for A to trust in B; in many cases it’s quite likely that B isn’t even aware of the existence of A. Similarly, even if A follows and thereby trusts (within the context of Twitter) B, it doesn’t necessarily mean that they are friends, or even know each other.

Another important property of trust that most trust metrics make use of is transitivity of trust. This concept tells us that trust, to a certain extent, can be transmitted from one node to another even when those nodes are not direct neighbors, by virtue of their shared trusted connections. This transmission is not perfect and it has a limited distance (i.e., trust diminishes the farther one moves away from the original source of the trust), but it has nonetheless been found to be a useful concept for estimating local and global trust.

Global trust scores consider overall trust levels in a network, typically registering and representing these trust values in the form of a trust matrix. Global trust measures can be useful for authorities managing complex networks of interconnected nodes, but may be less useful for an individual node wanting to know whose recommendation they should trust. A well-known example of a global trust metric is EigenTrust, originally designed for use in peer-to-peer (P2P) file sharing systems [5.4].

Local trust scores provide estimates of trust from the perspective of a given node with respect to other nodes in the network. In this case, the agent is less concerned with performance of the overall network but rather how that particular node can benefit from knowing their own trust values with respect to their neighbors (or their neighbors' neighbors). An example of a local trust metric is SUNNY, which can be used to provide trust-based recommendations for an individual user in a network [5.5].

Two of the most frequent applications of trust metrics have been seen in recommender systems and reputation systems (refer back to Chapter III for additional

detail). To be sure, these are far from the only applications of trust metrics but they are the ones that have received the most attention and research. Recommender systems provide suggestions to users for other products they may enjoy, other people they should connect with, other movies they should watch, and so on. One of the more popular techniques in recommender systems is collaborative filtering (CF). CF tends to work quite well when there is rich data, but in cases where data is sparse relying on trust metrics can prove to be a useful alternative.

Reputation systems provide a score keeping mechanism for contributors to online communities to publicly display the value they provide to the wider community. Examples include reviewers of restaurants on Yelp, reviewers of products on eBay or Amazon, and answerers of questions on Epinions or Quora.

### **Related Work**

Many researchers have investigated issues related to spread of misinformation and echo chambers in OSNs. These include [5.6], [5.7], [5.8], [5.9] and [5.10] and have investigated these issues in different platforms (Facebook, Twitter, etc.) and using different methods (analytics, AI/ML, etc.). Much work has also been conducted to automatically identify online bots, who frequently spread or amplify misinformation online (though they are not the only sources) [5.11] [5.12] [5.13] [5.14]. Related to characterizing trust in online networks, [5.15] provide a comprehensive treatment of online networks epinions.com and Ciao to understand the structure, topology, and dynamics of trust within these networks. [5.16] examines trust in online networks and find evidence for the existence of small

worldness in these networks. [5.17] examines trust on Twitter and propose a new method for estimating trust on Twitter, then ranking users based on their trust levels. To the best of our knowledge, no other researches have yet examined the intersection of trust and misinformation online.

### **Research Questions and Datasets**

In this chapter we don't seek to identify misinformation online; we seek to understand how and if *trust* influences its presence and spread, and what the topology of networks in which misinformation spreads look like.

This paper proposes three research questions as early explorations into the nature of trust and misinformation in an OSN, specifically in Twitter:

***RQ5-1: What is the topology and structure of an online misinformation trust network?***

*How is it similar to, and how does it differ from a regular information trust network?*

***RQ5-2: Does trust lead to the network structures that we see in an online misinformation network, or do misinformation networks lead to the types of trust relationships that we observe?***

***RQ5-3: How do the structures of misinformation trust networks influence their behavior?***

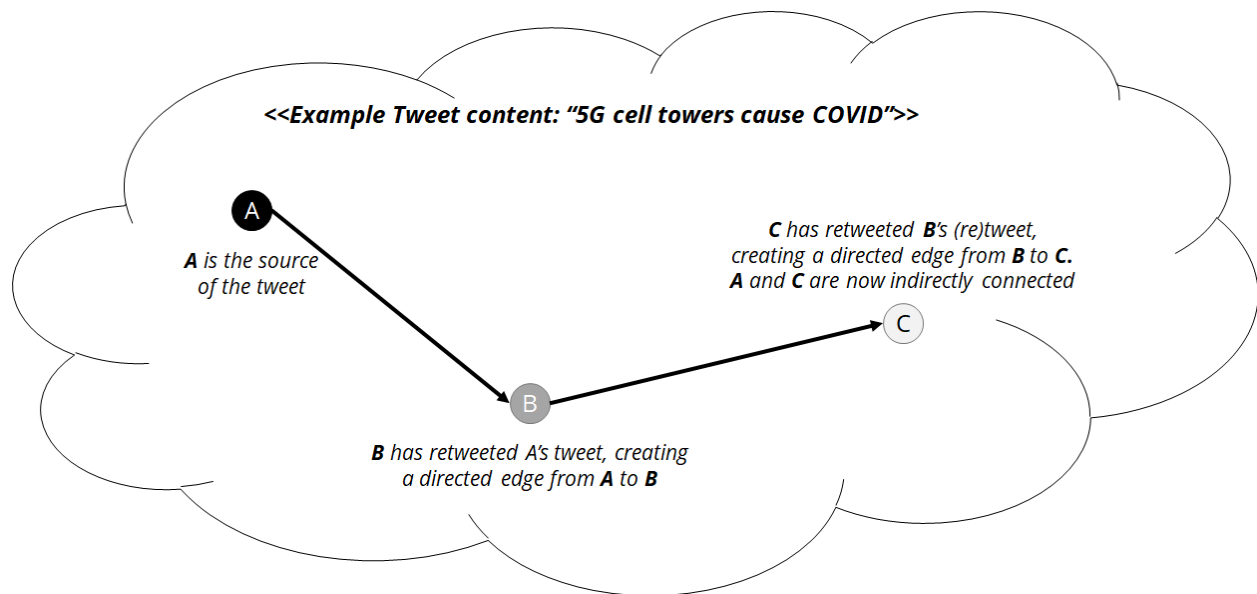
To answer these questions, we make use of a labeled COVID-19 misinformation tweets dataset from [5.18]. The authors gathered data from Twitter between 17 January 2020 and 15 May 2020, focusing on tweets that referenced 5G and the coronavirus pandemic. The authors refer to any public message on Twitter – which encapsulates tweets, retweets, replies, and quotes – as *statuses*. The authors collected more than 800,000 statuses related to COVID-19 during the aforementioned time period, then selected 10,000 of them at random to be manually classified.

Status was classified into three distinct categories. The first category was tweets that dealt with conspiracy theories conflating 5G and the coronavirus (i.e., claims that 5G networks were responsible for causing COVID-19). The authors selected this (and other COVID-related conspiracy statuses) because there was more scientific agreement about their falsity than there is for political tweets, for example. The second category was of tweets that dealt with conspiracy theories generally related to COVID-19, but not specifically focused on 5G (e.g., the virus wasn't real, or it was intentionally developed and released, etc.). The third and final category of tweets were non-conspiracy tweets related to COVID-19. For this paper, we make use only of the 5G COVID conspiracy tweets and the non-conspiracy tweets, as we reason that the specific use of one particular conspiracy theory will help to bring the results of our analysis into greater focus.

For each of these categories, the authors constructed subgraphs consisting of an edge list, a nodes list, and a subgraph plot. Each subgraph represents the structure of a *status* (tweet), not a user; that is, each subgraph traces the spread of a specific status



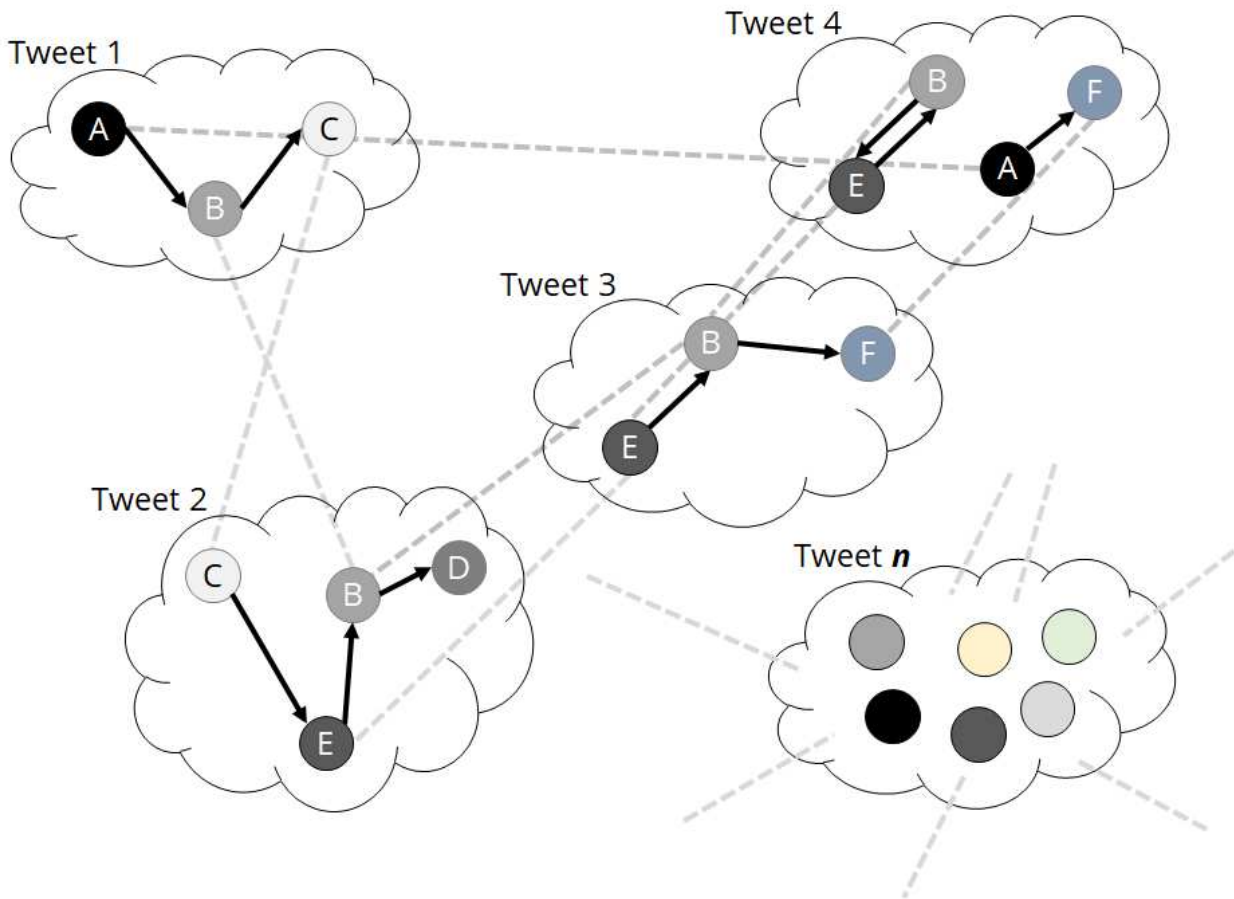
among specific users. The edge list for each subgraph represents follower relationships among users represented within that particular status, i.e. if Alice originates a status stating *"Does 5G increase your likelihood of getting COVID?"* and Bob comments favorably on that status, Bob and Alice share a directed edge with Alice as the sink. If Alice replies favorably to Bob's favorable comment, Alice and Bob share a directed edge with Bob as the sink. And, if Charlie retweets Bob's comment, Charlie and Bob share a directed edge with Bob as the sink. Figure 29 gives a graphical illustration of this construction. Additionally, in many of the subgraphs there are multiple components which may not be connected. The interpretation of this in the context of this dataset is that users may access information through other non-Twitter sources. For example, a user may be on Twitter, then leave Twitter and browse an online forum, encounter an item of misinformation, and then return to Twitter where they then tweet/retweet/comment/reply to it. One additional important note to mention related to the dataset and its construction is that, because of the how Twitter's API works, the authors of the dataset were limited to collecting 100 statuses per tweet; thus, the network sizes are each limited in the dataset, but in reality some of them may have been larger (perhaps much larger). Nonetheless, this limitation does not seriously affect our analysis, as we are less interested in the total number of users reached and more interested in the structure and dynamics of this reach.



**Figure 29:** Conceptual illustration of the construction of a tweet subgraph. Authors' own work

In the toy illustration presented in Figure 30, we can begin to see how a larger graph can be constructed based on the individual subgraphs. In Tweet 2, C is the originator of the tweet, where it is retweeted by E, then B, then D. This creates a direct path from  $C \rightarrow E \rightarrow B \rightarrow D$ , but it also creates an indirect path from  $A \rightarrow D$ , since A is connected to C in Tweet 1. In Tweet 4, we can see a case with two distinct components. In the first component with E and B, we can see that there are reciprocal edges between the two. This indicates that E originated a tweet, B retweeted it, and then E subsequently responded to B's retweet. And, in the other component in Tweet 4, we can see that A and F are connected, with A originating the tweet and F retweeting it. In the case of Tweet 4, the same subject is being discussed, but it was originated independently by E and A. Such cases may represent a range of possibilities, two of the most likely ones being 1) E and A both independently learned of the claim of the tweet from elsewhere (a different network within Twitter, or a

network outside of twitter like a forum), or 2) that there is in fact a path from E to A, but it is outside of twitter – for example, E could have texted the claim contained in the tweet to A, and A then posts it in Twitter.



**Figure 30:** Conceptual illustration of how tweet subgraphs are interconnected, forming the overall tweet graph

## **Methods**

To answer the research questions, we take a general exploratory approach to analyzing the structure and behavior of the empirical networks (and in some cases, analyzing specific nodes of interest, too), and we compare these to the structure and behavior of representative random graphs. In cases where there is no significant difference

between the random graphs and the empirical ones, we conclude that there is unlikely to be any underlying mechanisms of interest; in cases where there *are* significant differences between the empirical graphs and their corresponding random graphs, we explore the phenomena further and attempt to provide explanations for mechanisms that may contribute to the phenomena.

We first construct graphs from the misinformation dataset described in the previous section. We use Python3, and the following Python libraries: Pandas (data handling and data analysis), NumPy (mathematical and matrix operations), NetworkX (graph modeling and analysis), and Matplotlib (visualizations).

To facilitate system-level analysis, we join all of the 5G COVID conspiracy subgraphs into one overall 5G COVID conspiracy graph. We do the same with all of the non-conspiracy COVID subgraphs, creating one overall non-conspiracy COVID graph.

The networks are modeled as directed, unweighted graphs. Given that each subgraph represents a specific misinformation tweet, in this case a node represents a user (not a tweet), and when an edge exists between users it represents transmission of the tweet in question. Thus, if B retweets or comments on A's tweet, there is a directed edge from A to B. If, subsequently, A responds to B's comment, then a directed edge is also created from B to A, creating reciprocity between A and B. In this way, a subgraph is formed for the tweet in question. It should also be noted that within each tweet's subgraph there may be multiple unconnected components, as users may independently tweet a piece of misinformation after discovering from a different (non-Twitter) source. A node

(user) may also be present in multiple subgraphs because they interact with multiple different tweets.

Additionally, for part of our analysis we are interested in which users are present in both the conspiracy and non-conspiracy networks, and so we concatenate all of the subgraphs from each category (5G COVID conspiracy, and non-conspiracy COVID) of the [5.18] dataset into one graph for representing each category, which allows us to see how different users from each graph interact with one another more generally across different tweets. This results in the network sizes described in Table 14, which provides a summary of the characteristics of the two modeled empirical networks. It should be noted that the number of nodes  $n$  and the number of directed edges  $m$  from the conspiracy and non-conspiracy graphs do not sum to the same amount for the “Entire Network” column, and this is because there are a small number of nodes that are present in both graphs. Additional discussion of this point is provided later in this chapter.

**Table 14:** Number of nodes and directed edges for empirical networks analyzed

	5G COVID Conspiracies Graph	Non- conspiracy Graph	Entire network
# nodes $n$	11076	90663	98187
# arcs (directed edges) $m$	56129	322555	375087

The trust overlay network

A trust relationship, and by extension a trust graph, is different in subtle but important ways from a standard graph. A trust relationship doesn't have to be mutual, like

a friendship graph might be. A trustor (also referred to as a source) may trust in a trustee (also referred to as a sink), without that trust necessarily being reciprocated. By virtue of the interactions that occurred in our Twitter graphs, we assign a trust relationship to any interaction between two nodes. When A retweets B's status, a directed edge from A to B is drawn, and we infer that to some extent A trusts B (in this context), or he wouldn't have retweeted the status. Thus, we construct a trust overlay network as a representation of trust relationships on top of the original structural network.

Table 15 summarizes the trust overlay network, which is a virtual network modeling trust on top of the tweet statuses network. We see that in comparison with the tweet statuses network shown in Table 14, there are clear differences between the nature of trustors and trustees. In both the conspiracy and the non-conspiracy graphs, there are fewer trustors than trustees, and in the case of the non-conspiracy graph there are far fewer trustors than trustees. An implied fact of these figures is that there are some nodes who are only trustors – they only retweet statuses, but other don't retweet their statuses – and some nodes who are only trustees – their statuses are retweeted, but they don't retweet others' statuses. The nodes in the 5G COVID conspiracies graph also have a noticeably higher mean degree than those in the non-conspiracy graph. The interpretation of this is that the conspiracy nodes are more active and more likely to share or reshare statuses than the non-conspiracy nodes are. Additionally, there are important differences between a small subset of nodes that are present in both networks, which we discuss in the next section.

**Table 15:** Trust overlay network summary

	5G COVID Conspiracies Graph	Non- conspiracy Graph	Entire network
# trustors (sources)	7678	47502	52685
# trustees (sinks)	10736	88066	95378
Average degree $\langle k \rangle$	5.1	3.6	3.9

Because of the nature of the dataset used, we don't have more granular information about the detailed nature of interactions among users. For example, we don't know when an edge is formed between two users if it's because a tweet was retweeted, if it was commented on, if it was replied to, etc. – we only know that *one* of these actions happened. Additionally, with the dataset we used we don't have direct access to the *content* of the tweets, retweets, comments, etc. (though, in most cases, the tweets are still posted and could be matched with the records in this dataset). Pairing the graph structural data with textual analysis of the comments would add another layer of understanding to the findings.

Thus, we must rely on simple/naïve trust measures. This is precisely the type of scenario that graph-based trust metrics tend to perform best in, in which we know little about the characteristics of different nodes but we have fairly detailed information about the *structure* of the network. For this reason, we use a method similar to that described in [5.15], which focuses on simple trust measures based on various centralities; these measures only analyze direct trust between nodes that are directly connected.

### Defining metrics used: hubs, authority, PageRank, eigenvector centrality

To analyze our networks, we apply several different typical network analysis measures to them. In this section, we briefly discuss these metrics, how they are calculated, and what each is useful for.

#### Centrality measures

Centrality is an important concept in the study of networks, and there are many different centrality metrics that have been proposed by scholars. Regardless of the type of centrality being considered, centrality metrics generally seek to quantitatively understand the prominence of a given node based on its connectivity to surrounding nodes and to the broader network. Different centrality metrics are sensitive to different aspects of a graph, and thus require interpretation. In this chapter, we utilize two centrality metrics: eigenvector centrality, and PageRank. We select these two specifically because of the findings from [5.15], wherein the authors find that centrality measures PageRank and eigenvector centrality are most appropriate for application to computing trust metrics. This is because the number of trustors of a node isn't assured to be a reliable measure of trust in that node, and thus measures like degree centrality – which are directly reliant upon the in-degree or out-degree of a node – will suffer from this same shortcoming. The authors of [5.15] find that information in a trust network flows in simple paths (such as a direct path between a pair of nodes) and non-simple paths (such as triangles). For this reason, they conclude that measures like eigenvector centrality and PageRank can be expected to give



the most reliable measures of trust in a complex network, since trust can flow in simple paths and non-simple paths in these types of networks.

Eigenvector centrality considers the centrality of a node with respect to the centralities of its neighbor nodes. It tends to be a useful and robust centrality measure for many different types of networks because a node can achieve a high importance through having many connections, or by being connected to other important nodes – or both. The eigenvector centrality of a given node is calculated as in Equation 3 [5.19], where  $x_i$  is the node in question,  $A_{ij}$  is an element in an adjacency matrix,  $k_i$  is the largest of the eigenvalues of  $\mathbf{A}$ , and  $x_j$  are  $x_i$ 's neighbors:

$$x_i = k_i^{-1} \sum A_{ij}x_j$$

**Equation 3:** *Eigenvector centrality of a given node,  $i$*

The PageRank algorithm was used as the initial algorithm for the Google search engine. Although PageRank was first proposed for ranking importance of web pages, PageRank returns a ranked listing of nodes in a network, and as such it has been applied broadly to many other domains. PageRank makes use of the same centrality feedback mechanism in which nodes with higher prestige provide more prestige when linking to other nodes [5.19], with an important difference being that it relies on a random walk strategy which eigenvector centrality does not.

Like PageRank, the HITS algorithm (hyperlink-induced topic search) was originally designed for ranking Web pages, but it has since been adapted to many other uses too

[5.20]. HITS utilizes two important concepts, Hubs and Authorities. Hubs are nodes in a network who serve in some sense as clearinghouses, connecting many different nodes or resources, but which on their own aren't authoritative. Authorities, on the other hand, are linked to by many hubs. Said another way, hubs receive high scores for the values of its links, while authorities receive high scores for the value of the node itself (its content in the case of the Web).

### Geodesic paths (path lengths)

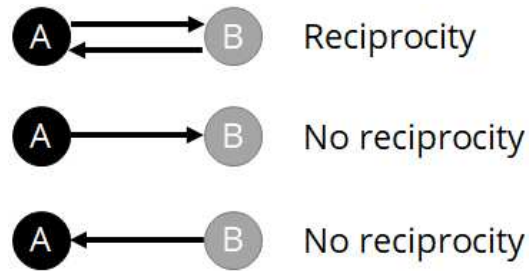
Geodesic paths (also referred to as a shortest path) give the shortest possible distance (measured in number of edges traversed) between two nodes in a graph, given a set of paths between the nodes. Given an adjacency matrix  $\mathbf{A}$ , the shortest distance between two nodes  $i$  and  $j$  is given by finding the smallest value of  $r$  in the equation below.

$$[\mathbf{A}^r]_{ij} > 0$$

**Equation 4:** *Geodesic distance/shortest distance*

### Reciprocity

Although reciprocity is not required for a trust relationship to be present, measuring reciprocity in trust networks give us a useful additional data point, and an indication as to the *level of trust* present in the network. In its simplest form, reciprocity is observed when a pair of nodes in a directed network both link to each other, as in the figure. In a trust network, when reciprocity exists between a pair of nodes, it implies some level of mutual trust within a certain context.



**Figure 31:** Graphical illustration of reciprocity among pairs of nodes

Overall reciprocity in a graph can be calculated by simply identifying the number of edges in the graph which are reciprocal, and dividing by the total number of edges [5.19].

$$r = \frac{1}{m} \text{Tr}\mathbf{A}^2$$

**Equation 5:** Equation for calculating reciprocity in a directed network

### Assortativity

Degree assortativity measures how likely it is for a node to connect with a node of another type, and can take any value from -1 to 1 [5.21]. A degree assortativity value of 1 would indicate like nodes connecting only to like nodes, a degree assortativity value of 0 would indicate no pattern to the types of nodes that any given node connects to, and an assortativity value of -1 would indicate that nodes connect only with nodes different from themselves.

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j^{\text{in}} q_k^{\text{out}})}{\sigma_{\text{in}} \sigma_{\text{out}}}$$

**Equation 6:** Calculating assortativity in a network. Adapted from [5.21] *Mixing patterns in networks*, Newman.

### Generating random networks

To provide us with a useful null model for comparing the empirical networks to, we generate a set of random networks based on the empirical networks, using the configuration model. The configuration model generates a random graph based on a given degree sequence [5.18], which we take to be the actual degree sequence from the empirical networks. In this type of configuration model, the degree of each node is fixed, which leads to the direct result that the number of edges in the network is also fixed. We randomly generate 100 random networks using the configuration model for the directed 5G COVID conspiracies graph, and the non-conspiracies COVID graph. The results of the 100 random networks are averaged when compared against the empirical networks. Where appropriate, comparisons of the empirical networks with the random networks are provided in the following sections.

### **Results and Discussion**

In this section, we present the results of our analyses, and interpret their meaning.

#### Key findings

We find that eigenvector centrality is a useful and simple estimate of direct trust for both online misinformation networks and online information trust networks, demonstrating lower time runtime than other more complex trust metrics.

We find that in misinformation trust networks the *sources* of (mis)information enjoy higher trust, while in the non misinformation network the *consumers* of information are the ones who enjoy relatively higher trust. Using eigenvector centrality as a measure of direct

trust, when considering this metric from a global level we find that the misinformation trust network has an order of magnitude (approx. 10x) more trust than the non misinformation network. We hypothesize that this is one of the mechanisms that causes misinformation to be more prevalent in these networks, but also suspect that it is an iterative loop where the sharing of misinformation also causes trust to increase.

We find that a strong influence on the trust levels in the networks – and by extension the likelihood that misinformation will spread – comes from a relatively small number of nodes who can be classified as *brokers* (discussed in greater detail below). The brokers make up less than 4% of the total network analyzed, yet they represent a magnitude or order more of the transmission of information and misinformation in the network.

With these same brokers, we find that in the context of misinformation trust networks brokers may be less skeptical of claims from others and more likely to trust their peers, compared to the general population. Based on our analysis, we find that in online misinformation trust networks broker nodes are more trusting and more trusted, helping to reinforce misinformation beliefs within a community and increasing the likelihood that misinformation will spread to other groups.

We find evidence for small worldness in the online misinformation trust network. To the best of our knowledge, this is the first publication that specifically identifies evidence for small worldness when considering the *intersection* of trust and misinformation online.

That is, small worldness in a trust network in an online context appears to either influence or be influenced by – or both – the ease with which misinformation spreads.

We find that misinformation nodes are more active and more likely to share or reshare statuses than the non-misinformation nodes are.

We find that there are many smaller communities that share similar ideas but which aren't directly connected, indicating that transitive trust has real-world limits on how far it can be propagated and that this propagation distance should be directly related to the network's structure.

We find that in misinformation trust networks the component sizes are smaller than in information networks, allowing trust to more easily form. These smaller component sizes may lead to more frequent and deeper contact among nodes than would be observed in larger components.

We find that misinformation nodes demonstrate mildly positive assortative mixing. In positive assortative mixing, nodes that are similar to one another by some measure tend to connect mostly with other similar nodes (“birds of a feather”, or homophily). In negative assortative mixing, nodes that are different from one another by some measure tend to connect more with one another (“opposites attract”) [5.21]. In our case, this finding indicates a preference by the conspiracy-oriented nodes to trust in and seek information from other similar (conspiracy-oriented) nodes, while the non misinformation nodes exhibit slightly negative assortative mixing, leading them to seek out more diverse information sources.

We find that there are noticeably shorter mean path lengths in the misinformation trust network than in the information trust network, enabling misinformation to spread easier and faster.

Summary statistics

First, we present summary statistics to orient us as to the nature of the networks that were analyzed. In Table 16, we sketch out the general characteristics of the networks in terms of number of nodes, number of (directed) edges, the size of the weakly connected component for each, the number of components, the densities, and the assortativity.

**Table 16:** Summary Statistics for Directed Network Models

	<b>5G COVID Conspiracies Graph</b>	<b>Non- conspiracy Graph</b>	<b>Entire network</b>
# nodes $n$	11076	90663	98187
# arcs (directed edges) $m$	56129	322555	375087
size of weakly connected component $n_{cc}$	9383	83713	91015
# components	276	1634	1774
Density	0.000458	0.0000392	0.0000389
Degree assortativity	0.053	-0.0603	-0.0518

In examining Table 16 we see that the size in terms of both nodes and directed edges of the non-conspiracy graph is much larger than the 5G COVID conspiracies graph, a fact that is not surprising given the insular nature of most conspiracy theory networks. As was mentioned previously, we also see that the sum of nodes and edges from each network is greater than it is for the entire network, and this is because of the presence of

some nodes (3522 of them) in both networks; this will be discussed in greater detail later in the paper.

The weakly connected component of a directed graph is the size of the connected component of the underlying undirected graph, ignoring edge directions. If ignoring edge direction, all nodes within the connected component can be reached by all other nodes. We can see from Table 16 that for both networks, most nodes are included in the weakly connected component.

We also see the number of components in each network is relatively large, implying that although most nodes are part of the weakly connected component, there are nonetheless a significant number of nodes that are part of their own small independent components. Keeping in mind that each subgraph of the conspiracy and non-conspiracy networks represents a specific tweet around which activity occurs, the real world interpretation of this for these components is that 1) there are a large number of tweets which, relatively speaking, don't gain as much traction as more popular ones, and/or 2) there are many smaller communities that share similar ideas but which aren't necessarily directly connected.

Network density is a measure of how many edges are present relative to the total number of possible edges (which is a function of the total number of nodes in the network). For directed graphs, network density is given by Equation 7, with the denominator representing the total possible number of edges; a network with density of 1



means that every node is connected to every other node, and a network with density of 0 means that there are no edges present.

$$d = \frac{m}{n(n-1)}$$

**Equation 7:** Equation for calculating network density

From Table 16, we see that density in both networks is low (close to 0), which is typical for real-world networks with many possible connections. It is interesting to note, however, that the density of the 5G COVID conspiracies network is an order of magnitude higher than that of the non-conspiracy network, a likely indicator of higher levels of trust within the conspiracy network.

Finally, in Table 16 we also presented the degree assortativity of each network. A high degree of assortativity indicates homophily, which is frequently observed in social networks. We see that for both networks the degree assortativity is close to 0; nonetheless, there is a small preference within the 5G COVID conspiracies network to connect with other nodes of similar degree, and in the non-conspiracies network we see a slight preference for nodes to connect with nodes of different degree. One possible interpretation of these values is that the conspiracy minded nodes have a slight preference for trusting in and seeking information from other nodes that are similar to them, while the non conspiracy nodes have a slight preference for trusting nodes that are different from them, seeking out more diverse information sources.

## Visualizations

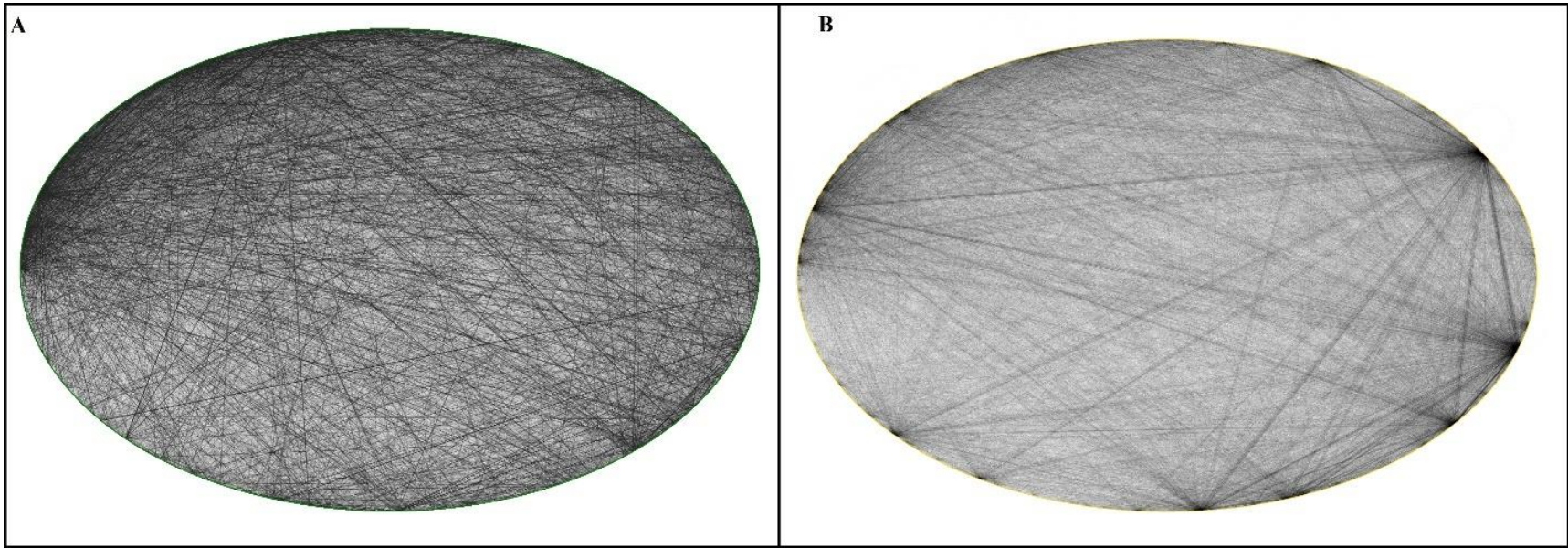
On its own, visualizing a network doesn't necessarily provide actionable information at a detailed level. Nonetheless, by visualizing the networks we can start to get a sense for their structures, giving us clues as to what elements of them – if any – are promising for further investigation. Below, we provide visualizations of the connected components of the 5G COVID conspiracies network, and the non-conspiracies COVID networks.

In Figure 32, we see the weakly connected component of the empirical non-conspiracies network next to the empirical 5G COVID conspiracies network. From this figure, it's apparent that proportionally there are a small number of nodes (relative to the total network size) that occupy an important place in terms of connectivity, while in the 5G COVID conspiracies network we see a relatively higher proportion (relative to the total network size) of highly-connected nodes.

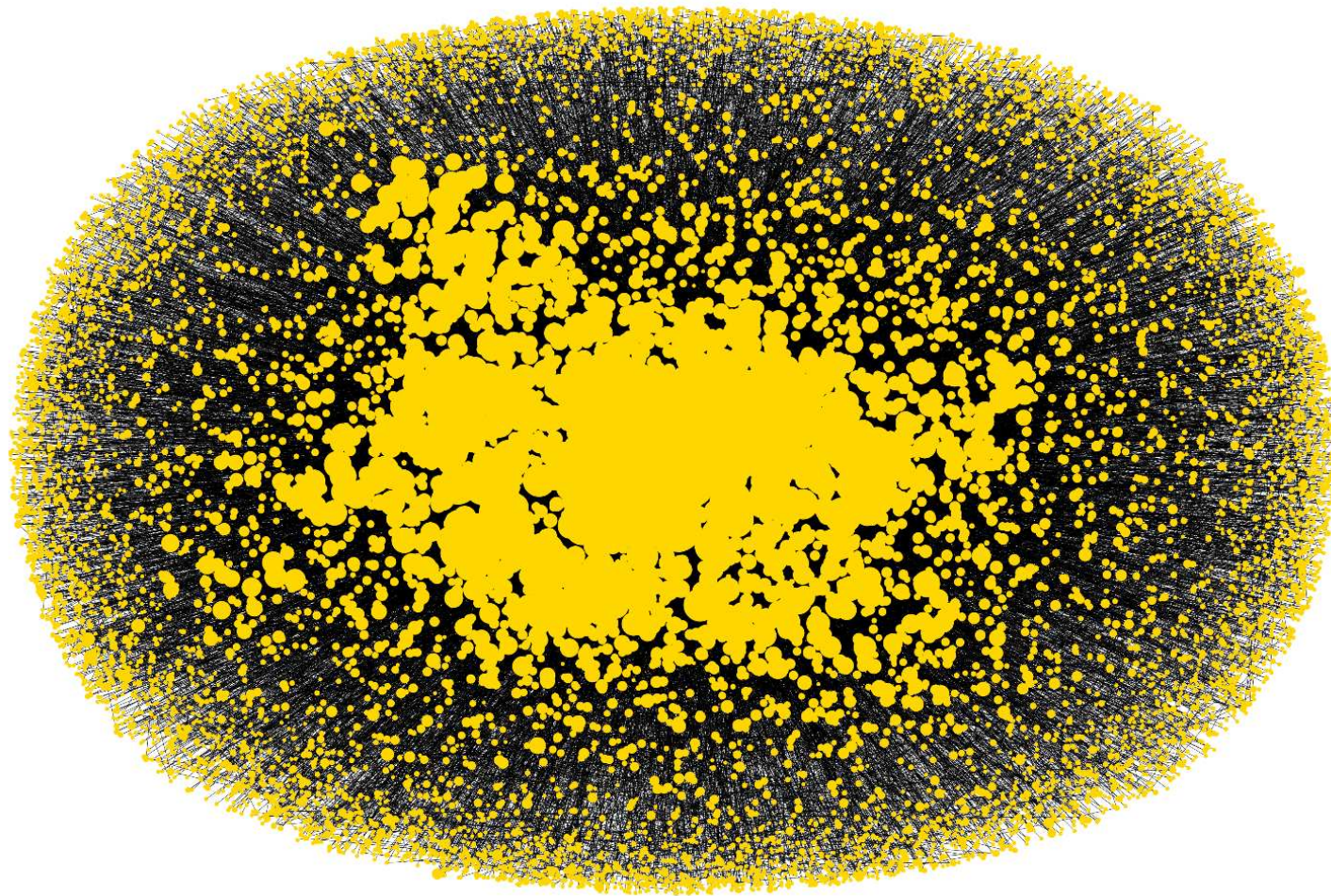
In Figure 33, we see a visualization of the 5G COVID conspiracies network, with the nodes sized according to their degree. Through this visualization, we get a clearer idea for the importance (as measured by connectivity) of different nodes within the network, where it is clear that the network is composed of a relatively small number of high-degree (highly-connected and – based on our definitions for purposes of this chapter – highly trusted) nodes at the core surrounded by many more relatively low-degree nodes on the periphery.

In Figure 34, we see the weakly connected component of the *empirical* 5G COVID conspiracy network (panel A) and the *random* version of the same network (panel B). There is a clear difference between the two networks, which indicates that there may be

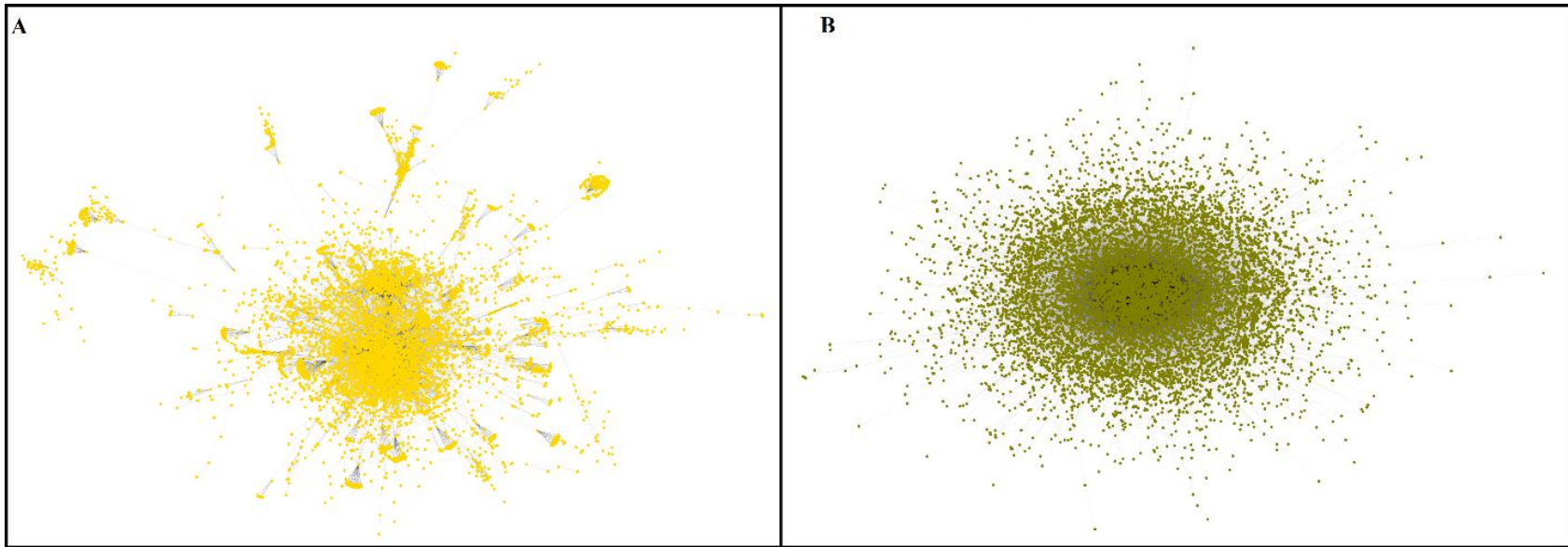
mechanisms of interest at play. Since we are interested in exploring the relationship between trust and the structure and behavior of misinformation networks, we can start to see that trust is likely to be high in these networks, as they are well-connected and generally have short path lengths between nodes, particularly in the cores.



**Figure 32:** Weakly connected components of empirical networks analyzed in this chapter, visualized using circular layout; panel (A) depicts the empirical non-conspiracies network, and panel (B) depicts the empirical 5G COVID conspiracies network



**Figure 33:** 5G COVID conspiracies network, visualized using NetworkX and matplotlib in Python language, with spring layout. In this figure, nodes' sizes are adjusted based on their degree, with higher degree being illustrated by a larger node size.

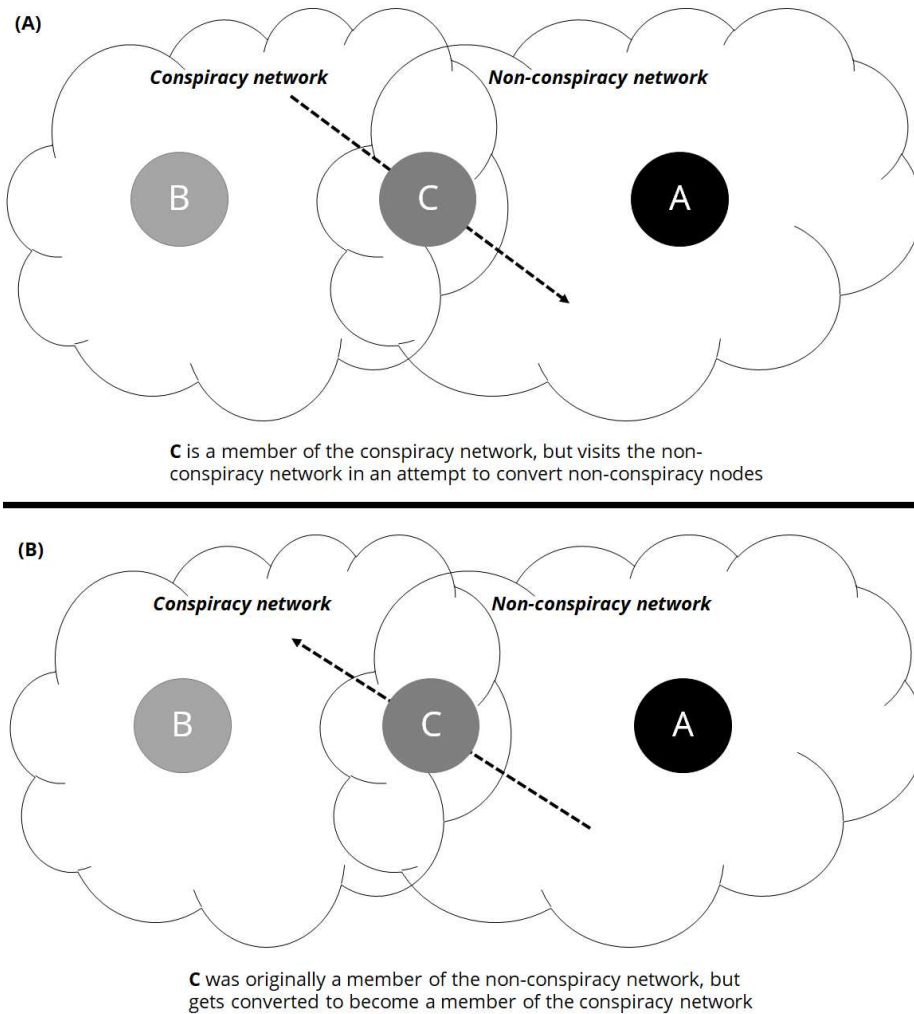


**Figure 34:** 5G COVID conspiracies networks weakly connected components, visualized using NetworkX and matplotlib in Python language, with spring layout. Panel (A) depicts the empirical network with nodes in gold, and panel (B) depicts the random version of the same network in olive.

### Who is in both networks?

Before proceeding further with the analysis of the networks' structure and behavior, we first examine an important question which we hypothesized would have a large influence on the same: namely, are there nodes that were present in both the 5G COVID conspiracies network and the non-conspiracies network and, if so, how did they differ in their attributes and their behavior from nodes that were found in only one or the other of the two networks?

What is the interpretation be of a node being found in both networks? Through reasoning we identify at least three likely possibilities, but we are unable to say with certainty which of these possibilities (if any) represent the reality because we don't have more granular information from the dataset about these nodes. The first possibility is that a node found in both networks believes in a conspiracy theory, and enters the non-conspiracy network to try and convince others of their belief. A second possibility is that a node found in both networks previously did not believe in a conspiracy, but later comes to believe in one. Finally, a third possibility is that *both* of these scenarios may happen for some nodes; the node may *first* have been a member of the non-conspiracy network, then gets converted to being a member of the conspiracy network, and then *returns* to the conspiracy network to attempt to convert more nodes.



**Figure 35:** Possible scenarios for explaining presence of nodes in both the conspiracies and the non-conspiracies networks. In panel (A), C is a member of the conspiracies network, and visits the non-conspiracy network to attempt to convert non-conspiracy nodes. In panel (B), C was originally a non-conspiracy member, but is converted and becomes a member of the conspiracy network.

We find a striking difference in the connectivity of the nodes that are found in both networks compared to nodes that are found only in one or the other of the two networks. From this point forward we refer to the nodes present in both networks as *brokers*. [5.22] says that “*brokers play an integral role in connecting different communities of actors, moving knowledge and information.*”



We find 3522 brokers – nodes that are present in both the 5G COVID conspiracies network and the non-conspiracies network. In Table 17, we present a summary of the trust network when considering the presence of brokers.

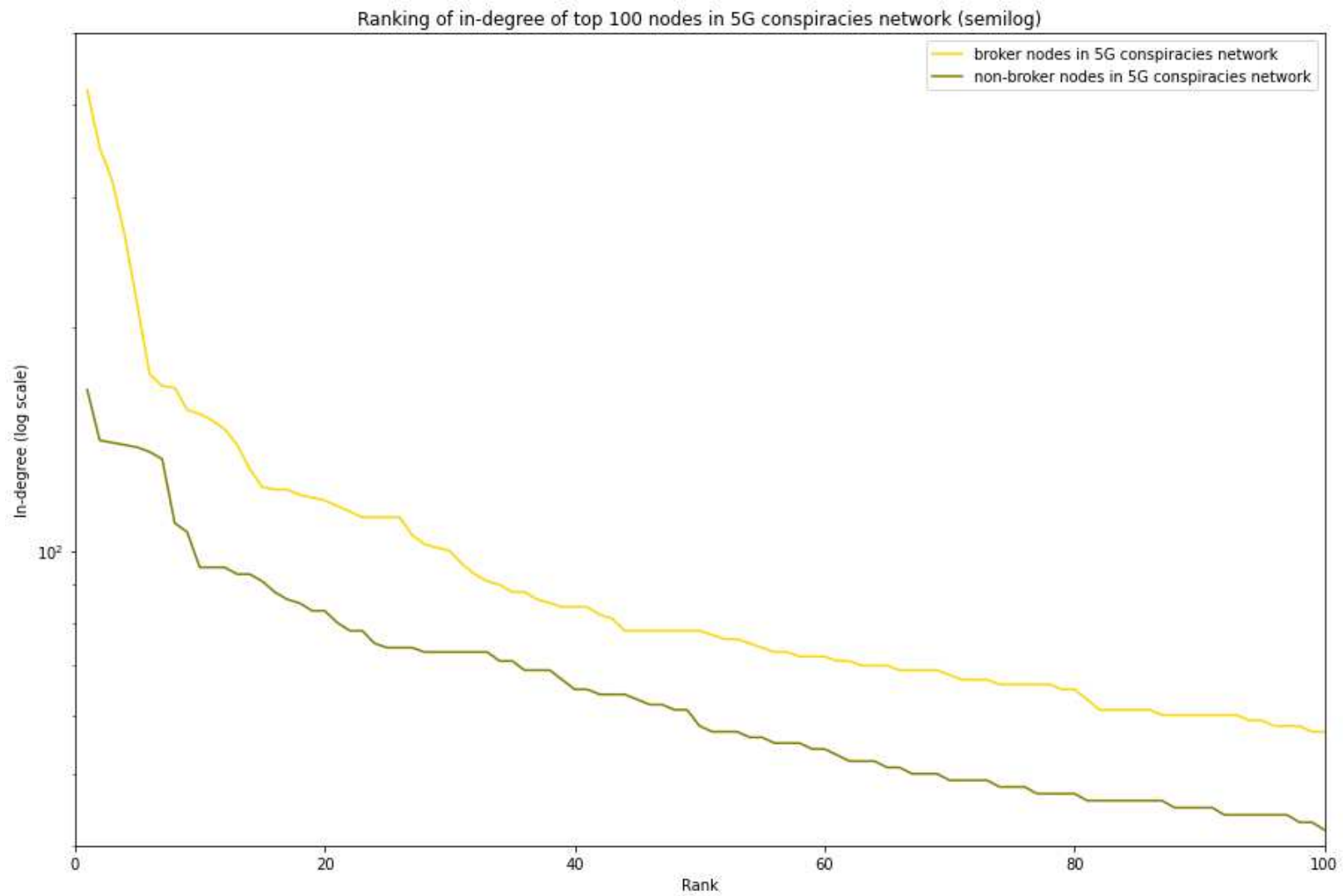
It's clear from Table 17 that the brokers have significantly higher connectivity than their non-broker peers. When comparing broker to non-broker nodes, we see a nearly threefold increase in degree connectivity (2.6x for the 5G COVID conspiracies network and nearly 2.9x for the non-conspiracies network). Additionally, for each subgraph we consider the ratio of trustors (sources of trust) to trustees (sinks or receivers of trust). Again, we find a significantly higher ratio of trustors : trustees when considering only brokers. A real-world interpretation may be that brokers in the context of misinformation trust networks (not generally) are less critical of claims from others and more likely to trust their peers, whether those peers are brokers or non-brokers. The trustor : trustee ratio is significantly lower in the non-conspiracy graph with brokers removed, close to a 1:2 ratio of trustors to trustees.

**Table 17:** Trust network with brokers

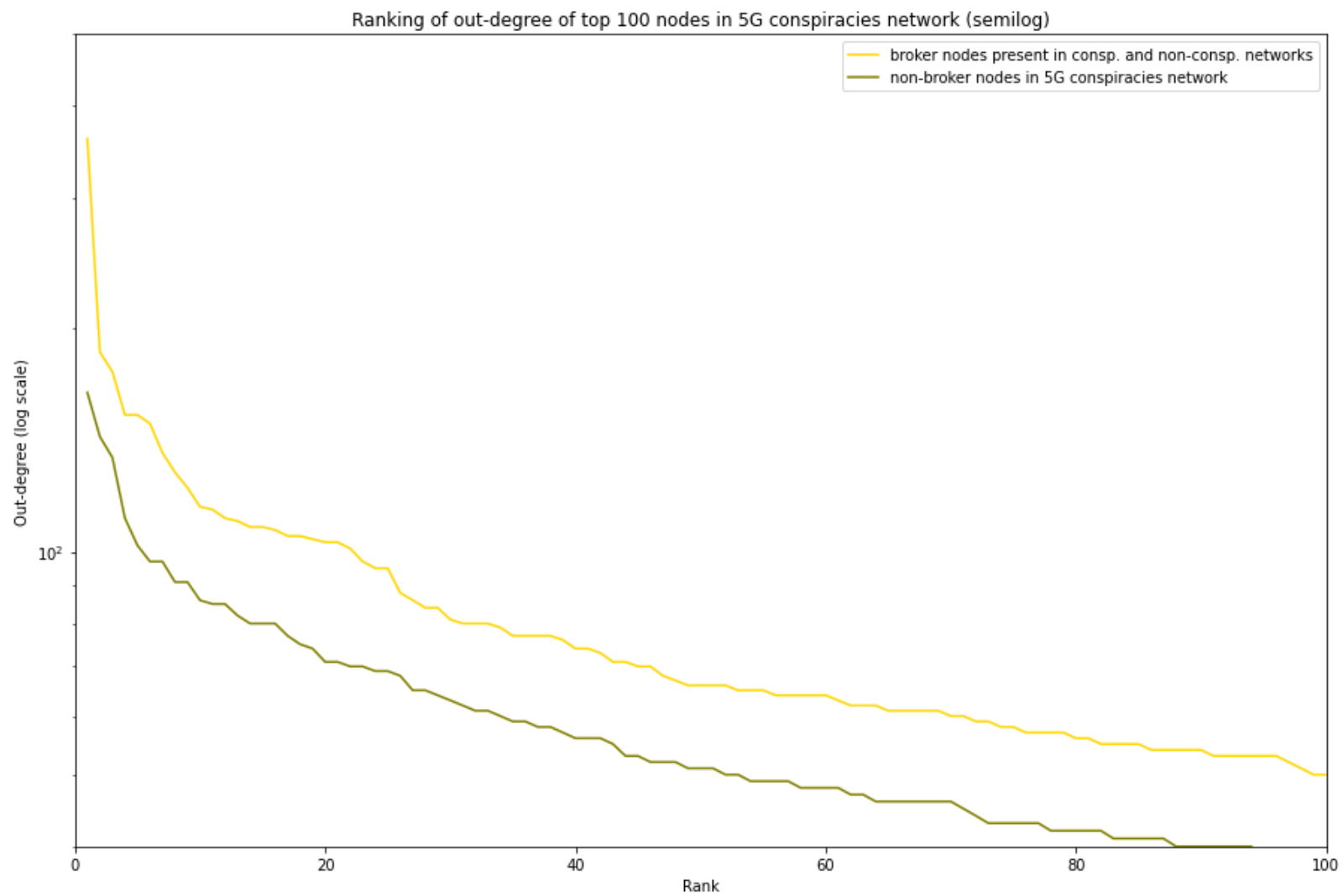
	5G COVID Conspiracies Graph	Non-conspiracy Graph	Entire network	Brokers only	5G COVID Conspiracies Graph w/o Brokers	Non-conspiracy Graph w/o Brokers
# trustors (sources)	7678	47502	52685	2803	4875	44699
# trustees (sinks)	10736	88066	95378	3462	7274	84604
ratio of trustors:trustees	0.72	0.54	0.55	0.81	0.67	0.53
Average degree $\langle k \rangle$	5.1	3.6	3.9	9.5	3.6	3.3

## Brokers

To better understand the nature of the brokers in this dataset, we consider rankings of broker nodes in the 5G COVID conspiracies network. In Figure 36, we see the ranking of the top 100 broker nodes compared with the ranking of the top 100 non-broker nodes from the 5G COVID conspiracies network based on in-degree (trustors). In Figure 37 we see the same rankings based on out-degree (trustees). For both trustors and trustees, we see a clear and consistent difference between the two groups: the broker nodes demonstrate significantly higher in- and out-degree than the non-broker nodes; the trend holds for the rest of the network beyond the top 100.



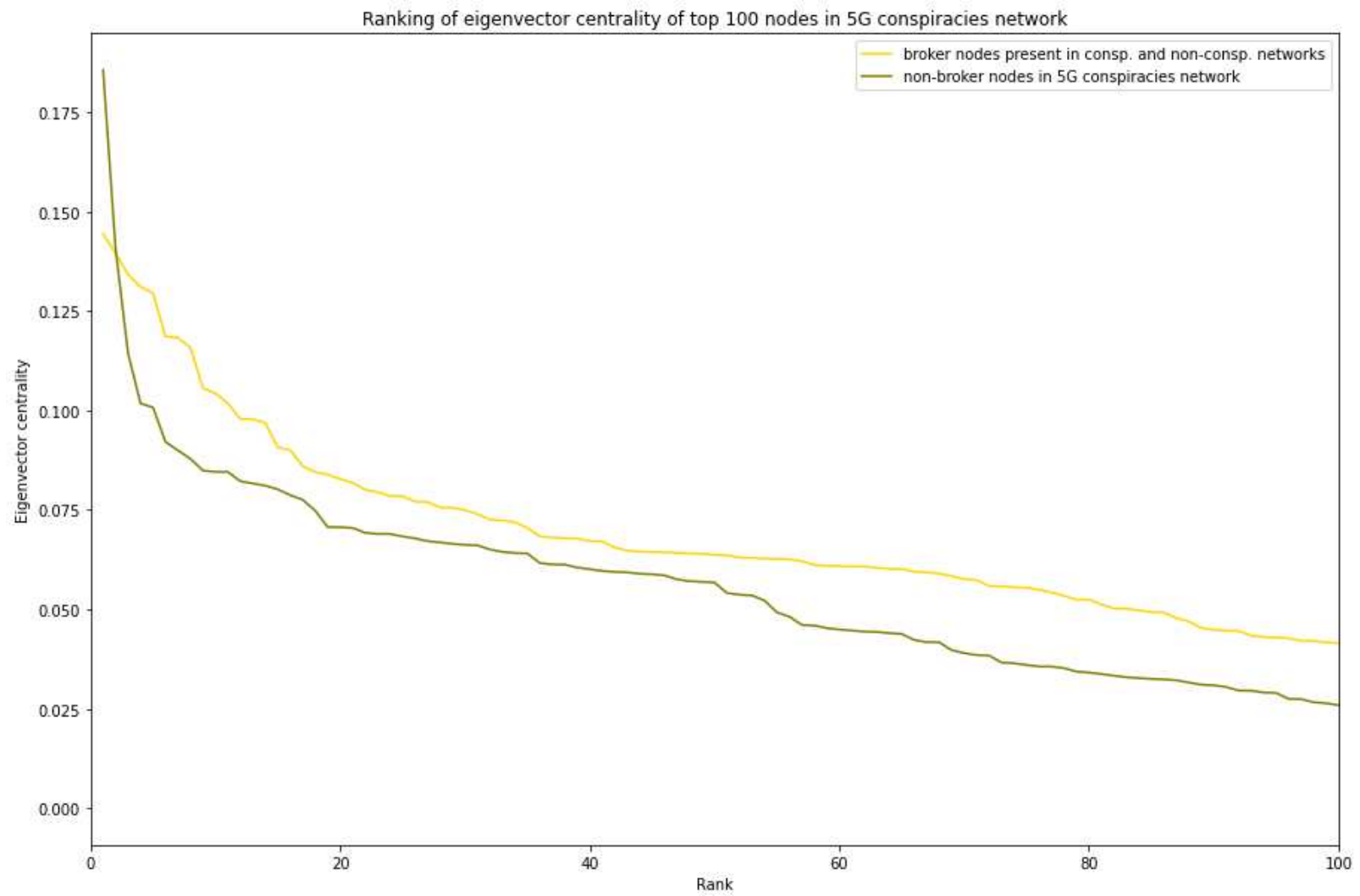
**Figure 36:** Ranking by in-degree of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale.



**Figure 37:** Ranking by out-degree of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale.

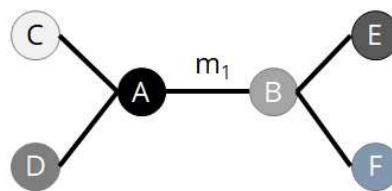
In Figure 38 we see a similar trend when considering the eigenvector centralities for the top 100 broker nodes and the top 100 non-broker nodes. With the exception of just a handful of nodes, the broker nodes once again have significantly higher eigenvector centrality values than their non-broker counterparts.

We interpret this as additional evidence that broker nodes in the context of a misinformation trust network are both more trusting and more trusted, leading to stronger within-group reinforcement of misinformation beliefs, and an increased likelihood of spread of misinformation beliefs to other groups.



**Figure 38:** Ranking by eigenvector centrality of top 100 broker nodes and non-broker nodes in the 5G conspiracies network. Broker nodes are depicted in gold, and non-broker nodes are depicted in olive. Semilog scale.

Broker nodes generally occupy a space on a bridge. In networks, bridges are edges that connect two otherwise unconnected components of the network; if the bridge (edge) were to be removed, the two components in question would no longer be connected. As such, bridges are another useful measure for understanding the criticality of a node in the overall performance of a network, even if that node would not otherwise be considered important based on connectivity measures. Figure 39 provides a visual illustration of a simple bridge, where  $m_1$  in the figure represents an edge which is also a bridge from **A** to **B**; if the edge  $m_1$  were to be removed the two components of the network would no longer be connected.



**Figure 39:** Illustration of a bridge in an example network

### Components

A component in a network is a group of connected nodes that together form their own subgraph within the broader graph. As was presented previously in Table 16, we find 276, 1634, and 1774 components in the 5G COVID conspiracies network, the non-conspiracies network, and the full network, respectively. We analyze the rankings of the components by size, comparing the 5G COVID conspiracies network to the non-conspiracies network. Figure 40 presents the results of this analysis for the 100 largest components in each network.

We see a clear difference between the two networks in component size, with the 5G COVID conspiracies component sizes being generally smaller than the comparable ones from the non-conspiracies network.

An interpretation of this related to trust may be that trust can more readily form and strengthen in the 5G COVID conspiracies network *because* the component sizes are comparatively smaller. The smaller component sizes could lead to more frequent and deeper contact among nodes than would happen in larger components.

### Degree distribution

To gain a better understanding of the structure of the networks being analyzed, we construct several plots depicting different views of degree distributions of the 5G COVID conspiracies network and the non-conspiracies network.

First, as was noted previously, there is a significant difference in mean degree between the two networks; the node with highest outdegree in the non conspiracies network is less than twice that of the node with highest outdegree in the conspiracies network, despite the fact that the non conspiracies network is several times larger than the conspiracies network. This indicates a relatively higher degree of activity, engagement, and connectivity in the conspiracies trust network than in the non conspiracies trust network.

To gain an understanding of nodes' trust, we use the same method as in [5.15] where prestige is used as a proxy for trust, with prestige being measured by the centrality values of nodes. In Figures 41 and 42 we present normalized plots of the out-degree against hubs, authority, PageRank, and eigenvector centrality for the 5G COVID



conspiracies network and the non-conspiracies network, respectively. With the way we've constructed our graphs, outdegree represents a trust relationship with the source of the outdegree being the sink of the trust (the trustee). In examining the figures, we notice several interesting points.

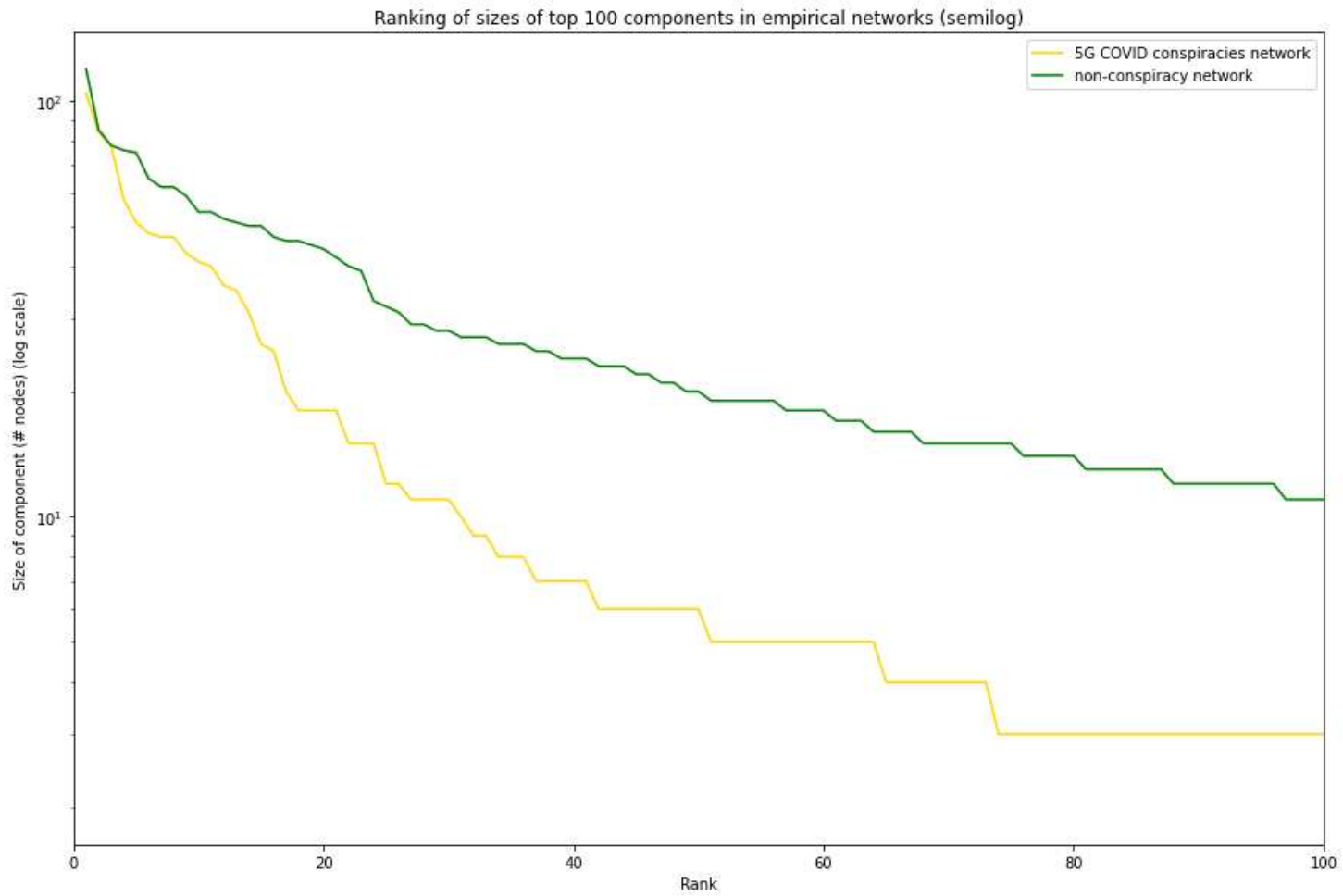
First, when comparing the conspiracies network (Figure 41) with the non conspiracies network (Figure 42), we notice a marked difference in the distribution of centrality measures against outdegree (trustees). In the conspiracies network, nodes with high outdegree (sources of information) generally have relatively high eigenvector centrality values, too. In Table 18, we see a correlation coefficient between outdegree and eigenvector centrality of 0.6836. On the other hand, in the non conspiracies network nodes with high outdegree (trustees) generally have *low* eigenvector centrality values, and nodes with high eigenvector centrality values generally have low outdegree. Referring to Table 18, we see a correlation coefficient of 0.0178 – barely different from zero. However, from Table 18 we also see a significantly higher correlation between indegree (trustor status) and eigenvector centrality for the non-conspiracies network (0.2688). The real world interpretation for this phenomenon is that in the conspiracies network, *sources* of (conspiracy) information – who have higher outdegree values than do their followers – enjoy higher trust, while in the non-conspiracies network the *consumers* of information are the ones who enjoy relatively higher trust. Additionally, when considering the mean eigenvector centrality values in both networks, we observe that the trust values (as measured by eigenvector centrality) are an order of magnitude higher in the conspiracies

network than the non conspiracies network. We interpret this as meaning that the overall levels of trust in the misinformation trust network are high than in the non conspiracies network, and we hypothesize that these higher levels of trust are a significant contributor to the ability of the misinformation to spread and reinforce beliefs.

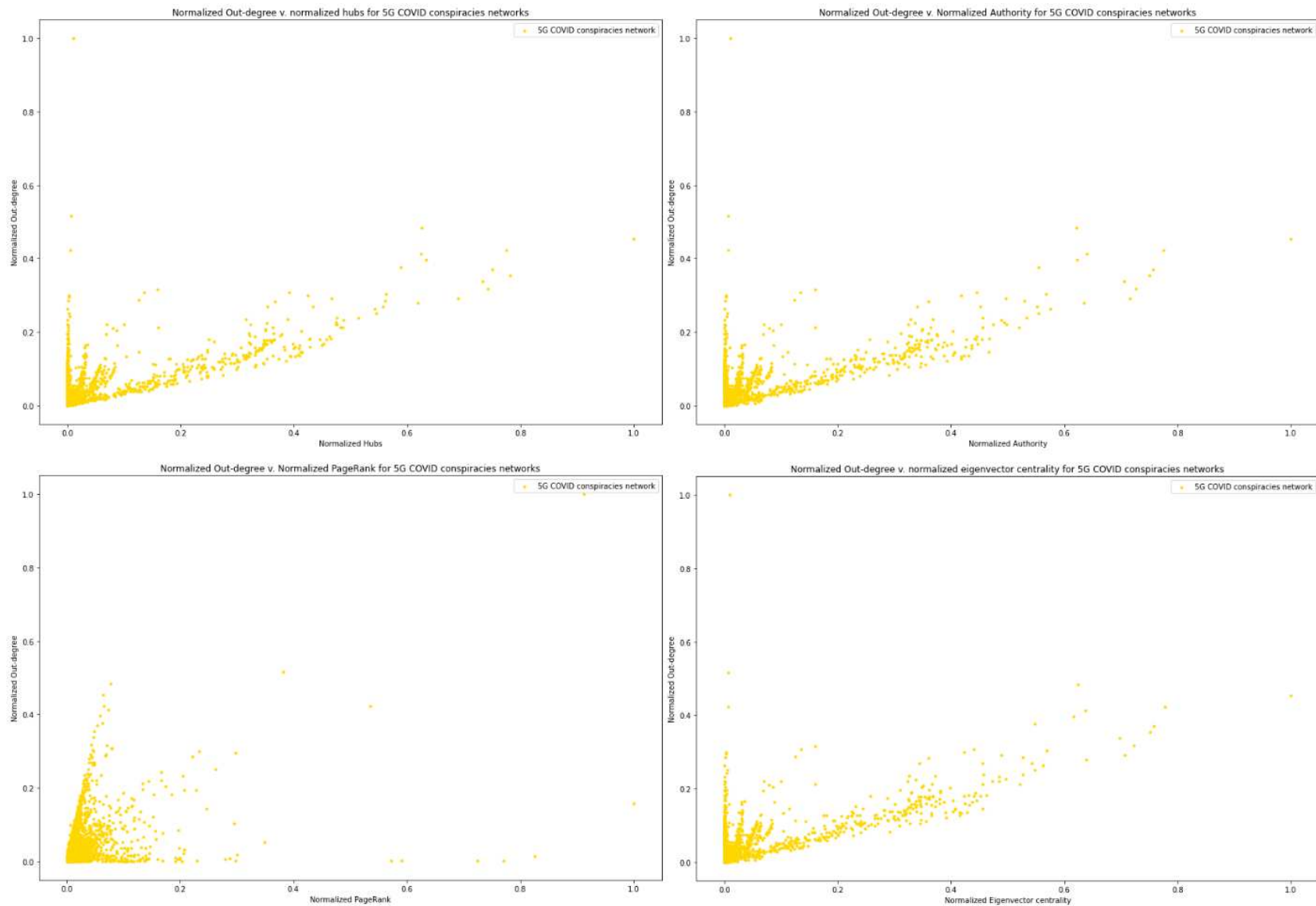
Second, for the conspiracies network we observe nonlinear relationships for all of the measures except for outdegree v. PageRank, which is roughly linear. In the non-conspiracies network, all of the measures appear nonlinear.

Next, for both networks we see very similar distributions in outdegree v. hub, outdegree v. authority, and outdegree v. eigenvector centrality, but a noticeable difference in outdegree v. PageRank. This is consistent with [5.15] findings that examined trust in the epinions.com and Ciao online networks. We confirm [5.15]'s findings and adopt eigenvector centrality as the most straightforward measure of direct trust for these types of networks.

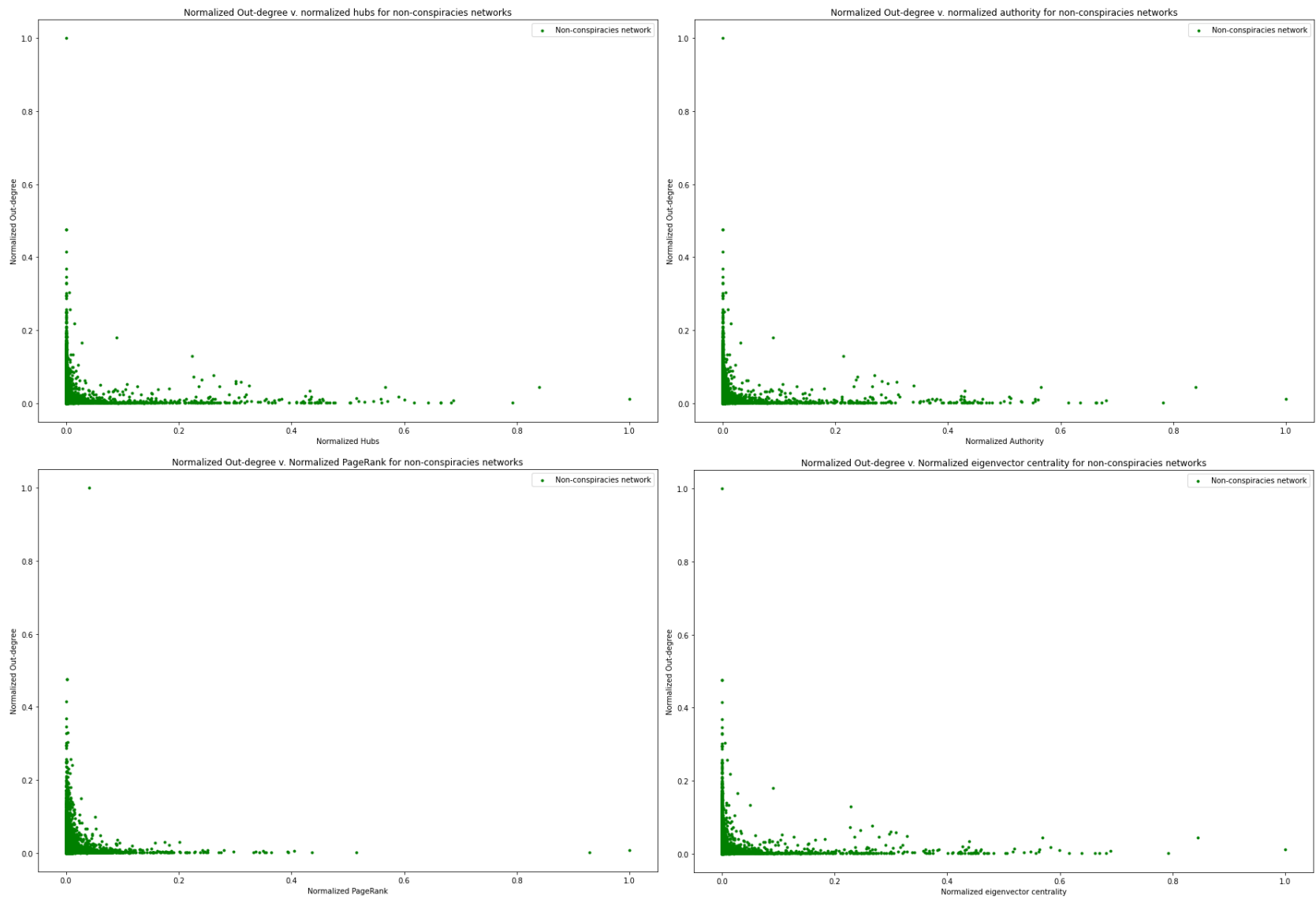
Finally, interestingly in the conspiracies network outdegree v. PageRank and outdegree v. eigenvector centrality plots are noticeably different from one another, but in the non-conspiracies network these two measures appear quite similar. We propose that this differential may be useful as a test for identifying networks that may be experiencing greater spread of misinformation. We also propose that eigenvector centrality functions as a more useful direct trust measure for misinformation trust networks.



**Figure 40:** Top 100 components by size for the empirical networks; the non-conspiracies network is depicted in green, and the 5G COVID conspiracies network is depicted in gold



**Figure 41:** Normalized plots of out-degree v. (clockwise from top left) hub, authority, eigenvector centrality, and PageRank for the 5G COVID conspiracies network



**Figure 42:** Normalized plots of out-degree v. (clockwise from top left) hub, authority, eigenvector centrality, and PageRank for the non-conspiracies network

In Table 18, we present correlations among the different measures studied for both networks. One interesting difference between the two networks is that there is a strong correlation (0.8410) between indegree and outdegree for the conspiracies network, but very little correlation (0.0221) between the same measures for the non conspiracies network. This indicates that in the conspiracies network there is a much higher degree of reciprocity (which we will discuss specifically in a later section) – a node is almost as likely to be a trustor as it is to be a trustee – while in the non conspiracies network there is little reciprocity. In the conspiracies network, we see a strong relationship between outdegree and eigenvector centrality, while in the non conspiracies network the strongest relationship is between PageRank and indegree.

**Table 18:** Correlations of network structural measures in conspiracies and non-conspiracies network

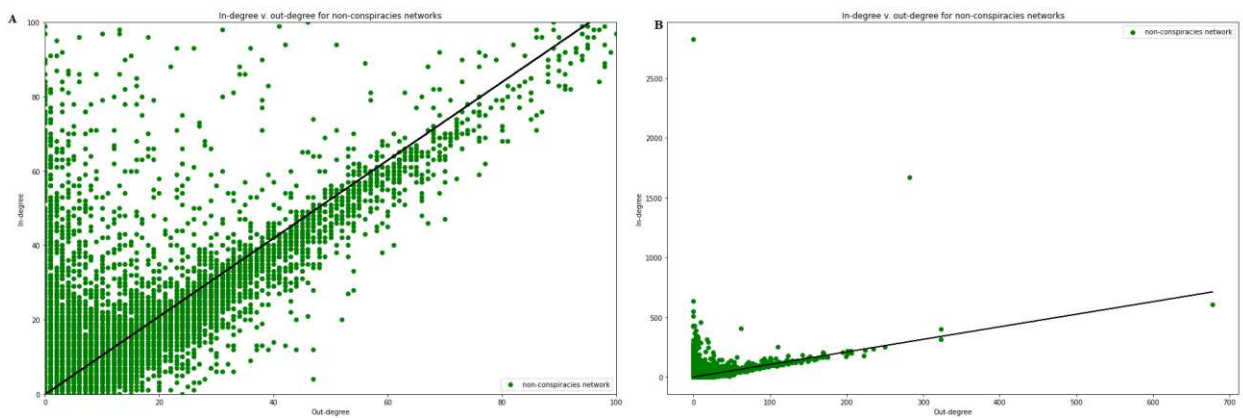
5G COVID conspiracies network

	in-degree	out-degree	eigenvector centrality	PageRank	hub	authority
in-degree						
out-degree	0.8410					
eigenvector centrality	0.5636	0.6836				
PageRank	0.7374	0.4370	0.0861			
hub	0.5594	0.6839	0.9975	0.0837		
authority	0.5640	0.6837	0.9999	0.0864	0.9974	

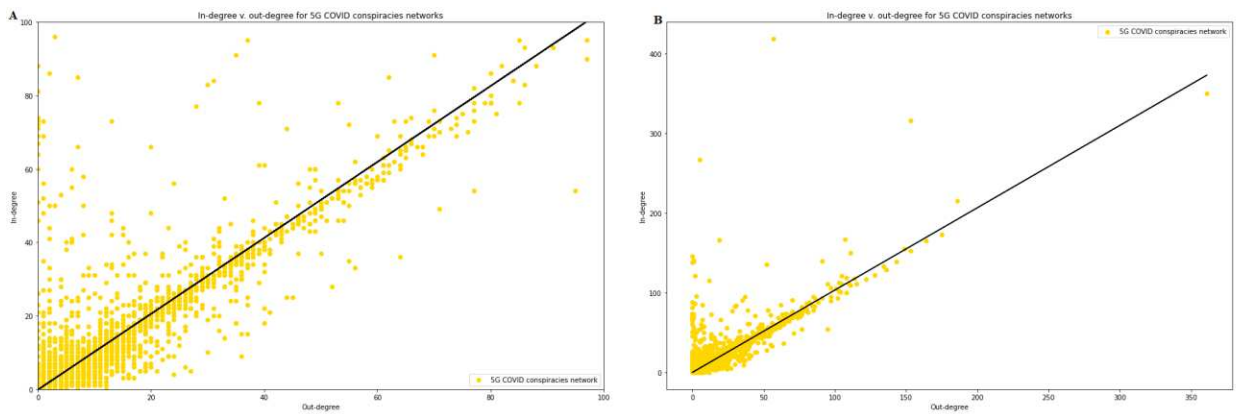
Non-conspiracies network

	in-degree	out-degree	eigenvector centrality	PageRank	hub	authority
in-degree						
out-degree	0.0221					
eigenvector centrality	0.2688	0.0178				
PageRank	0.7490	0.0114	0.0631			
hub	0.2845	0.0176	0.9969	0.0750		
authority	0.2503	0.0176	0.9932	0.0549	0.9943	

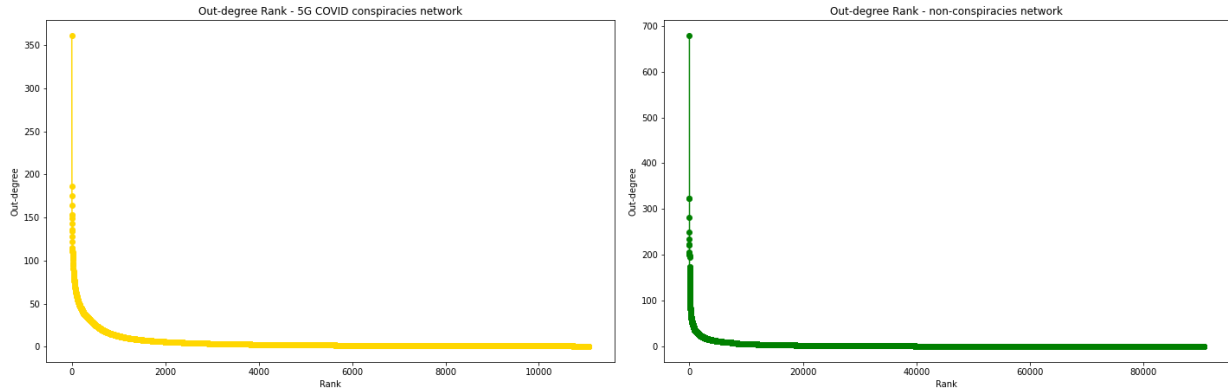
In Figures 43 and 44 we plot trustors v. trustees for the two networks. For each figure, the left panel is a zoomed in view of the lower end of the distribution where data points are denser, and the right panel is the overall distribution. In these figures, we see a close relationship between indegree and outdegree in the conspiracies network (roughly speaking, for each in-link there is an out-link) while in the non conspiracies network we observe that a node is much more likely to have a high indegree than a high outdegree.



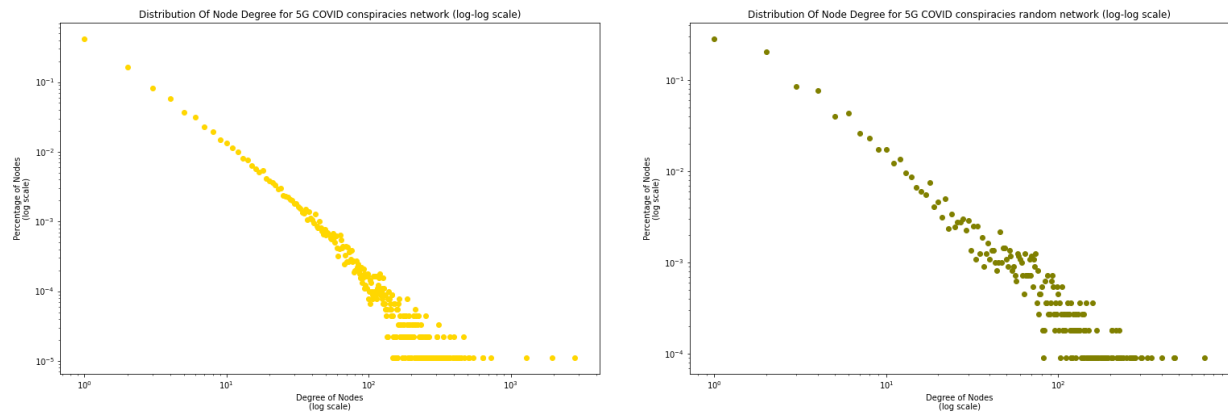
**Figure 43:** In-degree v. out-degree for the non-conspiracies network; panel (A) focuses on degrees below 100 where data is richer, and (B) illustrates the entire degree distribution



**Figure 44:** In-degree v. out-degree for the 5G COVID conspiracies network; panel (A) focuses on degrees below 100 where data is richer, and (B) illustrates the entire degree distribution



**Figure 45:** Out-degree ranking plot for 5G COVID conspiracies network (left, in gold) and non-conspiracies network (right, in green)

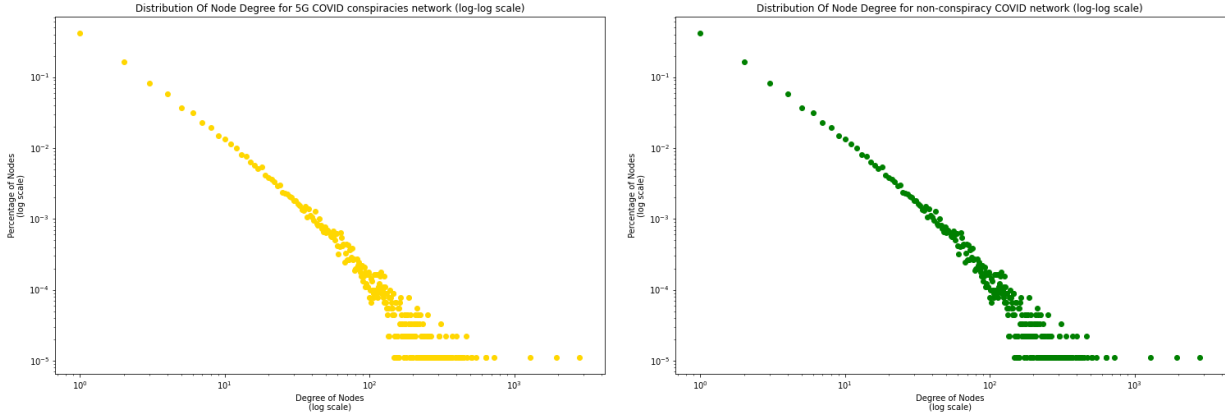


**Figure 46:** Degree distributions plotted with log-log scale; on the left (in gold) the empirical 5G conspiracies network is depicted, and on the right (in olive) the random 5G conspiracies network is depicted

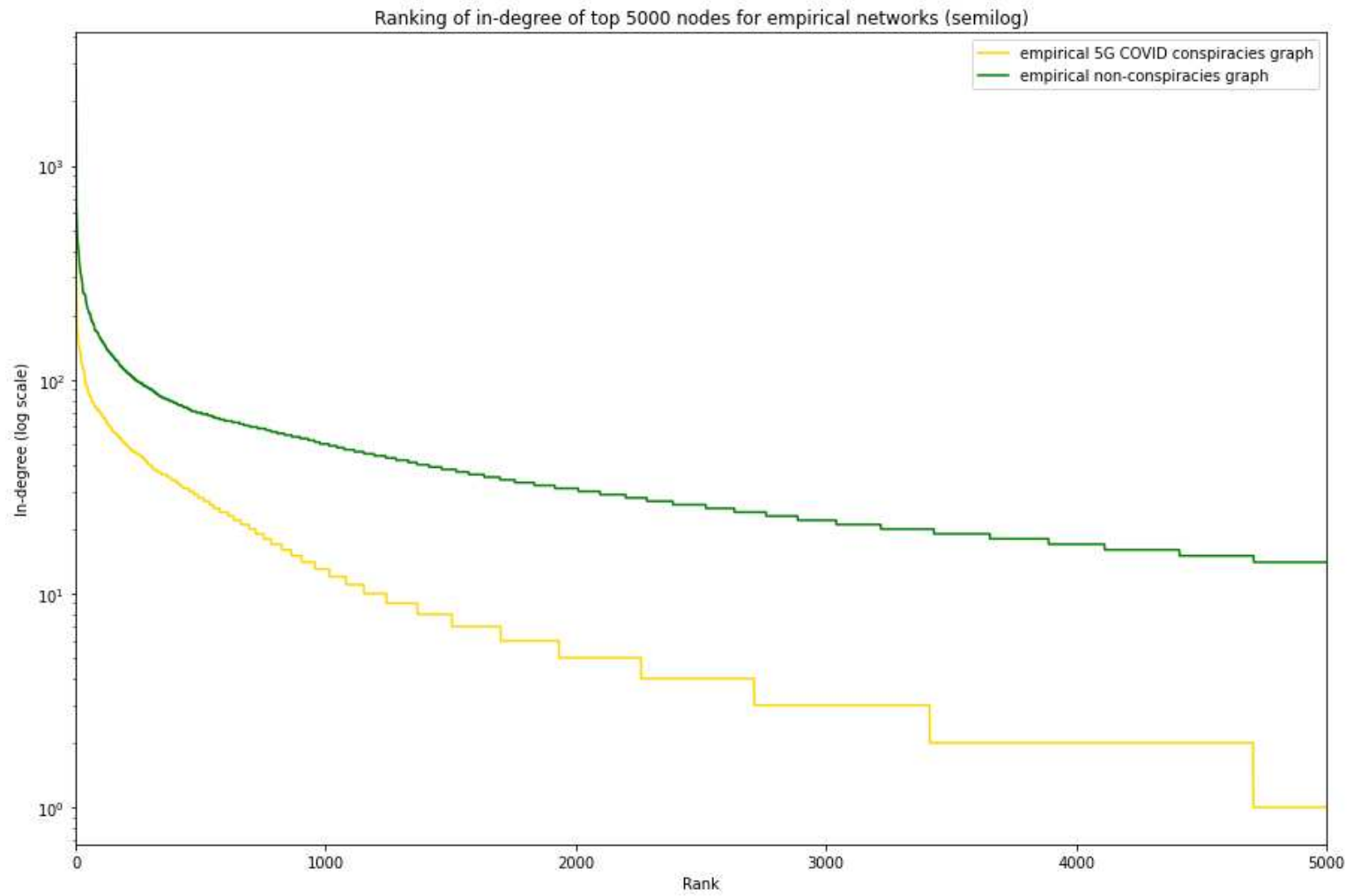
In Figure 47, we see the distribution of node degree for the 5G COVID conspiracies network and the non conspiracies network, plotted with a log-log scale. We observe stark differences between these distributions and a Poisson distribution. Both networks exhibit the characteristic attributes of a power law distribution, wherein there are a small number of very high degree nodes, together with a very large number of relatively low degree



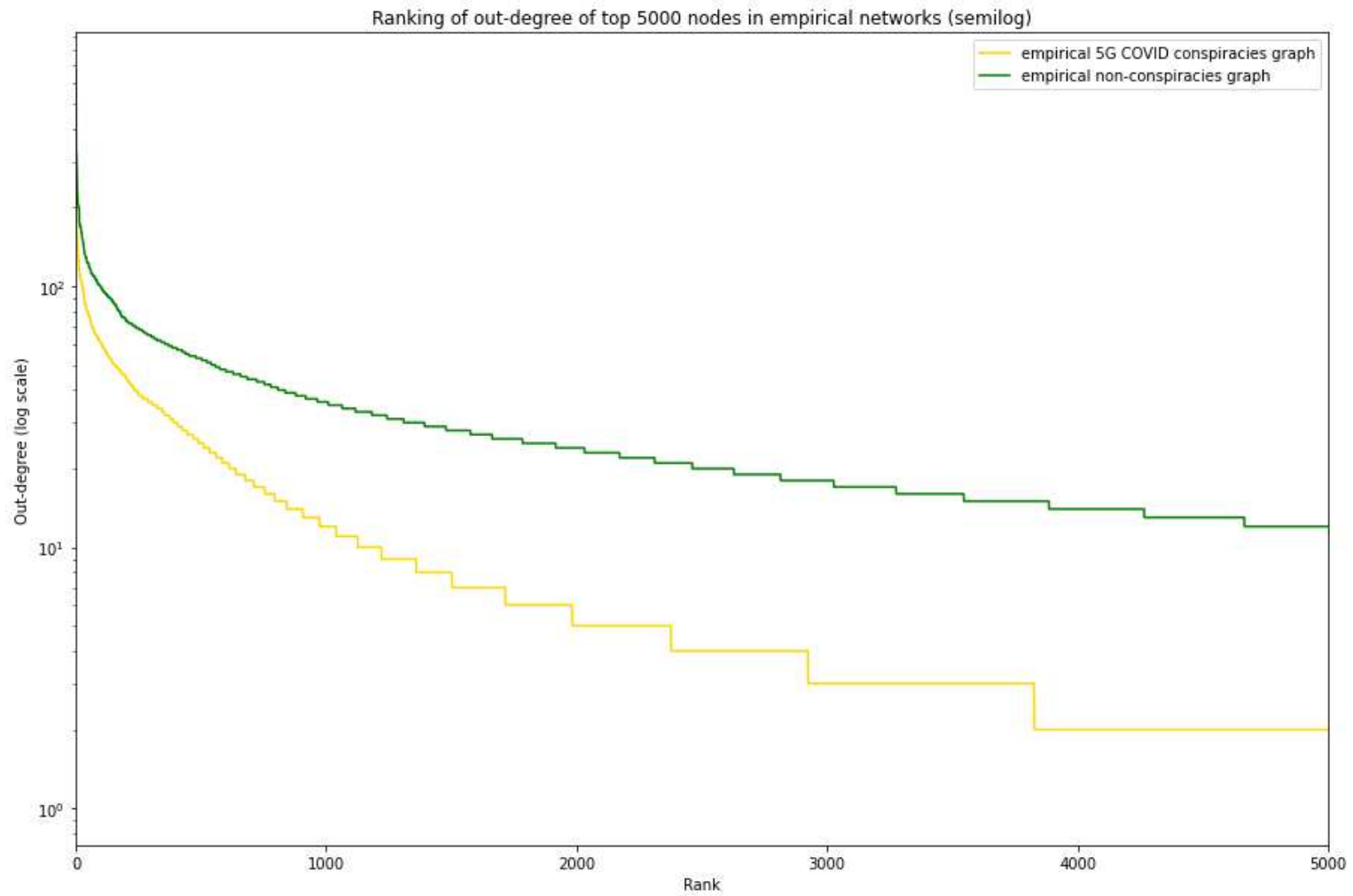
nodes, leading to a long tail in the distribution. [5.23] take this – together with a high clustering coefficient and small worldness – as an indicator of a complex network.



**Figure 47:** Degree distributions plotted with log-log scale; on the left (in gold) the 5G conspiracies network is depicted, and on the right (in green) the non-conspiracies network is depicted



**Figure 48:** Ranking of indegree of top 5000 nodes for empirical networks



**Figure 49:** Ranking of outdegree of top 5000 nodes for empirical networks

### Reciprocity

We compute reciprocity for the networks analyzed in this chapter, and find that the reciprocity in the 5G COVID conspiracies network is noticeably higher than that of the non-conspiracies network. This finding is separate from the networks' density and various centrality measures, discussed previously, and is another indication of higher trust among nodes in the misinformation trust network than in the information trust network. The interpretation of this is that total trust levels (as measured by reciprocity) are higher in misinformation trust networks than information trust networks.

**Table 19:** Reciprocity in the 5G COVID conspiracies graph and the non-conspiracies graph

	5G COVID Conspiracies Graph	Non- conspiracy Graph
Reciprocity	0.78	0.64

### Mean shortest paths

Because of the size of the networks being analyzed, it is computationally prohibitive to calculate all shortest paths. The time complexity of calculating all shortest paths in a graph is on the order of  $O(n^3)$ , where  $n$  equals the number of nodes in the graph. For large graphs such as the ones analyzed in this work (>90,000 nodes) we would need to calculate more than 730 trillion possible paths. Instead, we randomly sample 10,000 pairs of nodes and compute the shortest paths between these 10,000 pairs. Additionally, to find path lengths there must exist a path between a pair of nodes, so we consider only the

connected component of our graphs to ensure that there always exists a path between the pair of randomly selected nodes.

In Table 20 we present the mean shortest path lengths for the empirical and random graphs for the 5G COVID conspiracies network, and the empirical and random graphs for the non conspiracy network. One of the first things we notice is relatively short path lengths in both networks. Next, we notice the marked difference in shortest path lengths between the empirical and the random graphs, indicating that path length for both information trust networks and misinformation trust networks are likely to be an important driving factor of their behavior. Finally, we also make note of the shorter path lengths in the conspiracies network than in the non conspiracies network – more than one full link shorter, on average – with the interpretation being that all other things equal it is easier for misinformation to spread in the conspiracies network than in the non conspiracies network.

**Table 20:** Mean shortest path lengths for empirical and random graphs of the 5G COVID conspiracies network and the non-conspiracies network, respectively

	5G COVID Conspiracies Graph		Non-conspiracy Graph	
	<i>Empirical</i>	<i>Random</i>	<i>Empirical</i>	<i>Random</i>
Average shortest path length	5.57	4.10	6.80	4.69

## Centrality measures

### PageRank

We analyze the networks (empirical and random) using PageRank. Figures 50 and 51 both show a similar power law-like distribution for PageRank, with a small number of nodes having relatively high PageRank values, and a large number of nodes having relatively low PageRank values. Additionally, we see that there is close agreement between the empirical and the random graphs for both the 5G conspiracies network and the non-conspiracies network. Because of the similarities in the empirical and the random networks for PageRank, we conclude that PageRank is a less useful direct trust measure for understanding the nature of trust in a misinformation trust network.

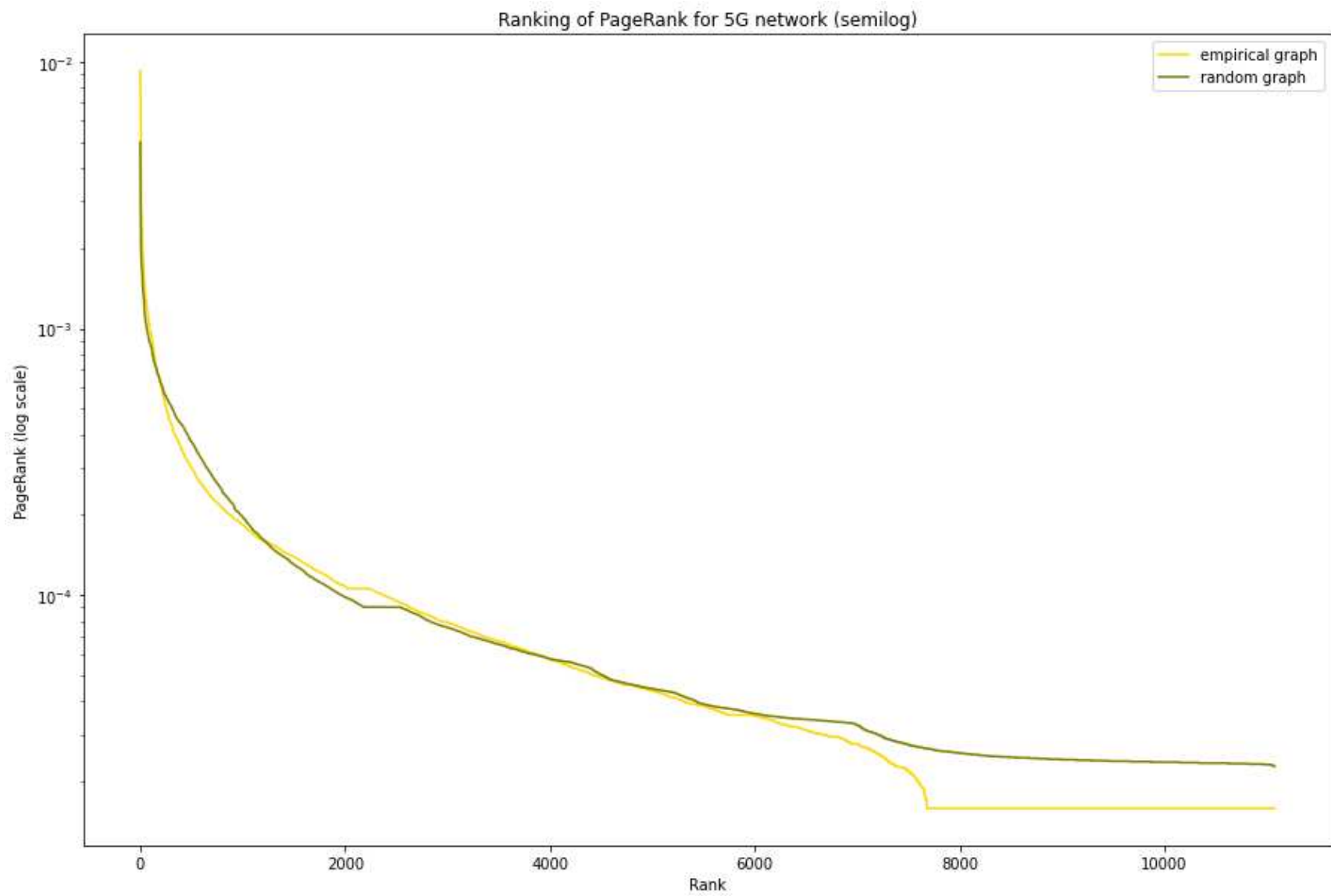
In Figure 52, we compare PageRank values for both empirical networks. We observe the interesting result that PageRank values are noticeably higher for the 5G COVID conspiracy nodes compared to the non-conspiracy nodes at almost all parts of the distribution curve. Within the context of how we have defined trust in this chapter, we interpret this to mean there are higher levels of trust in the misinformation trust network than in the standard information trust network.

### Eigenvector Centrality

Figures 53 and 54 present the eigenvector centrality rankings for the empirical and random versions of the 5G COVID conspiracies and the non-conspiracy networks, respectively. Unlike PageRank, we find a marked difference between measurements of eigenvector centrality for our empirical networks compared to the random networks. We

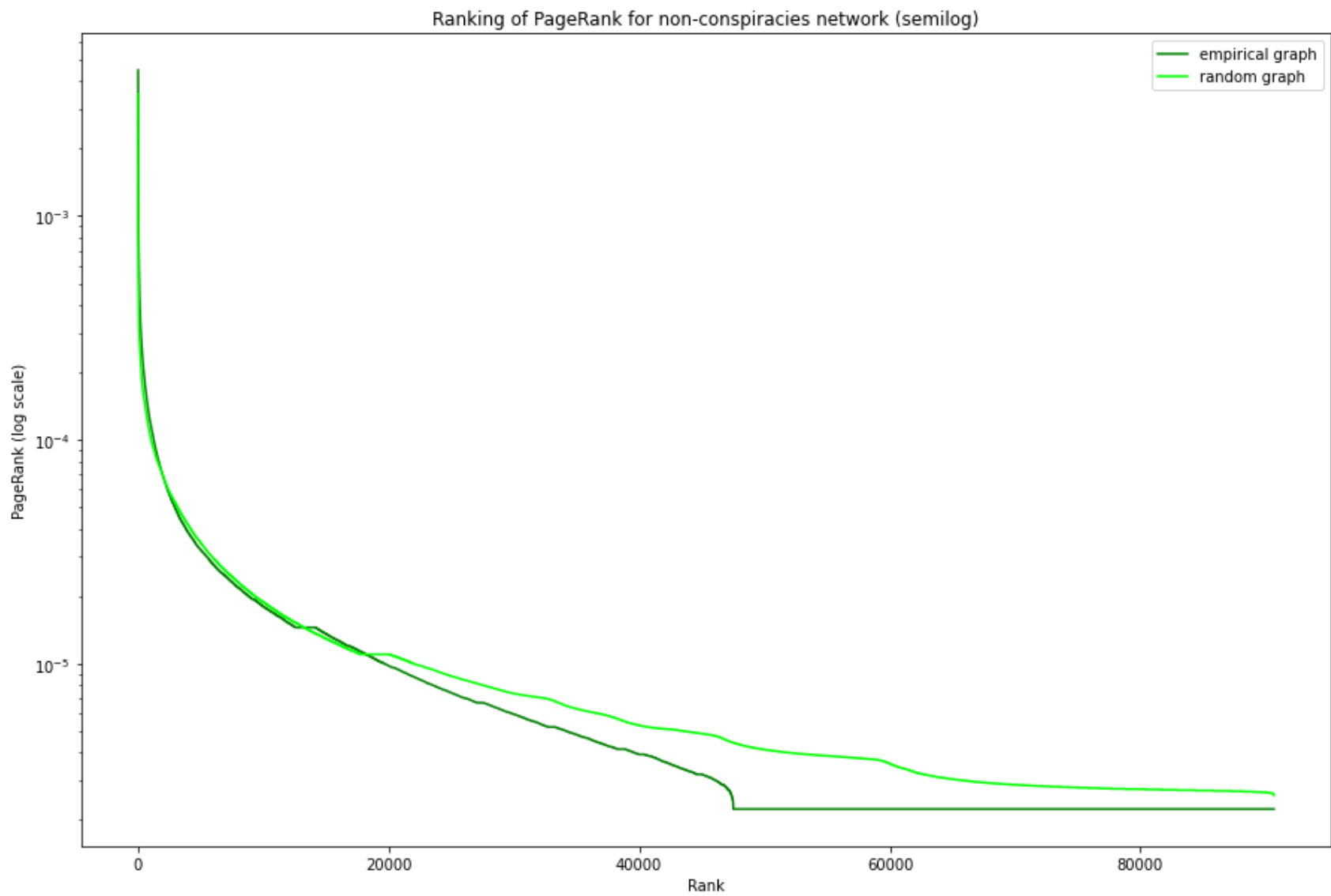
take this as another indication that eigenvector centrality serves as a useful measure of direct trust for these types of networks.

In Figure 55, we present a comparison of the two empirical networks' eigenvector centrality rankings to one another. In both networks, we see a similar maximum eigenvector centrality value (maximum value of 0.186 in the 5G COVID conspiracies network, and 0.197 in the non-conspiracies network). As discussed previously, we find that the trust in the misinformation network (as measured by eigenvector centrality) is an order of magnitude higher than that of the information (non conspiracies) network, enabling faster spreading of misinformation and a higher likelihood of these beliefs taking root.

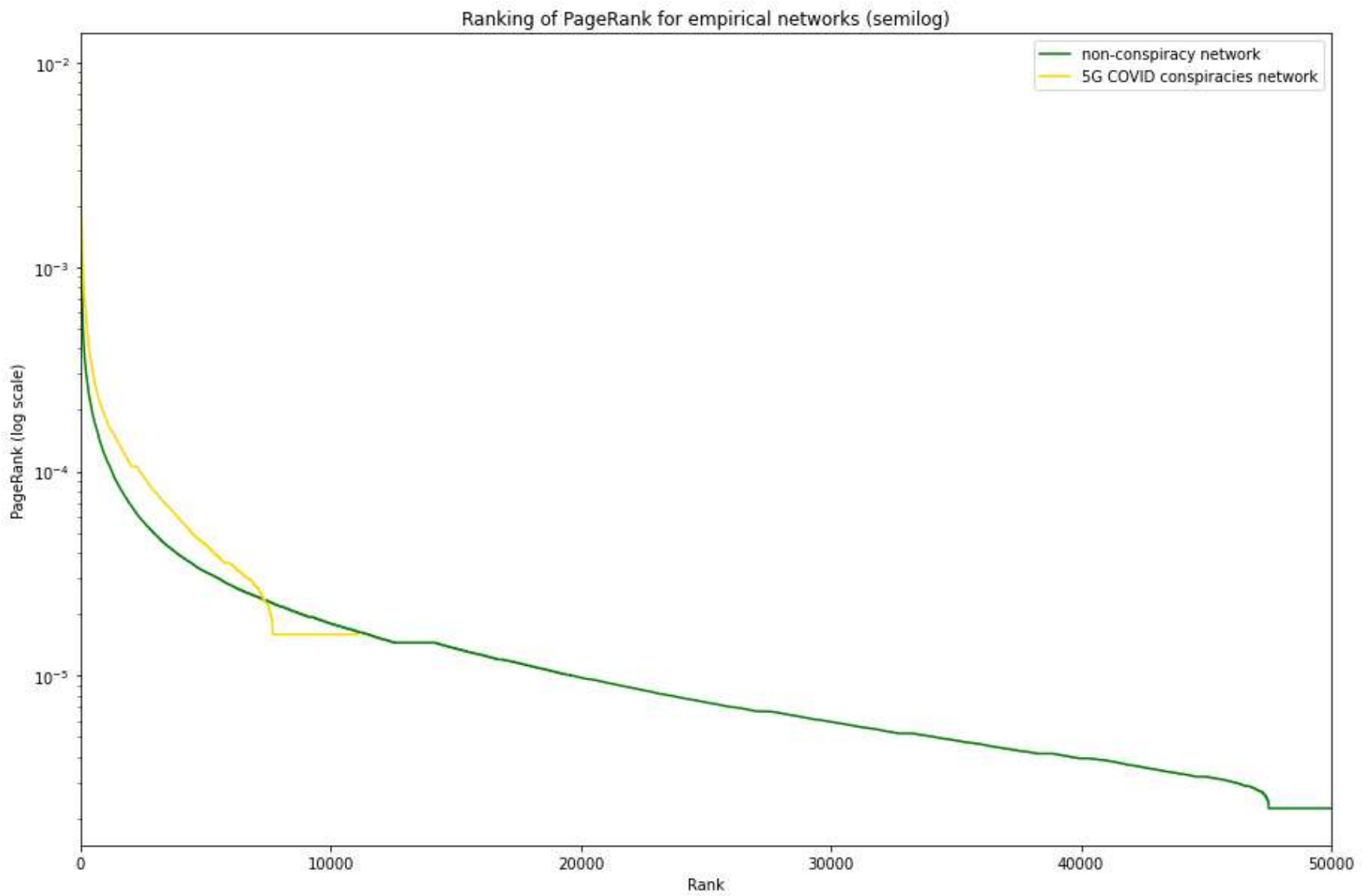


**Figure 50:** Ranking of PageRank for empirical and random versions of 5G COVID conspiracies network

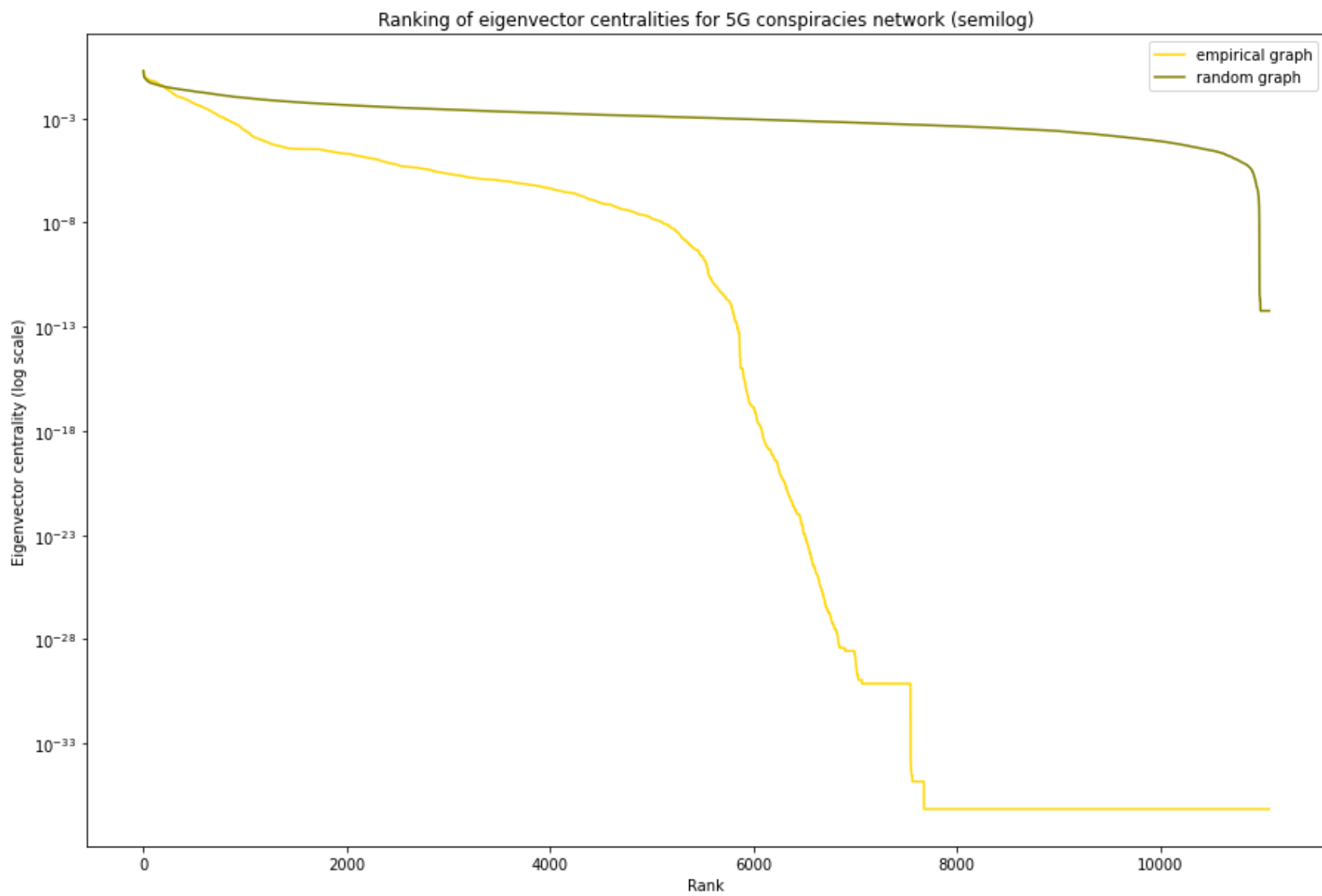




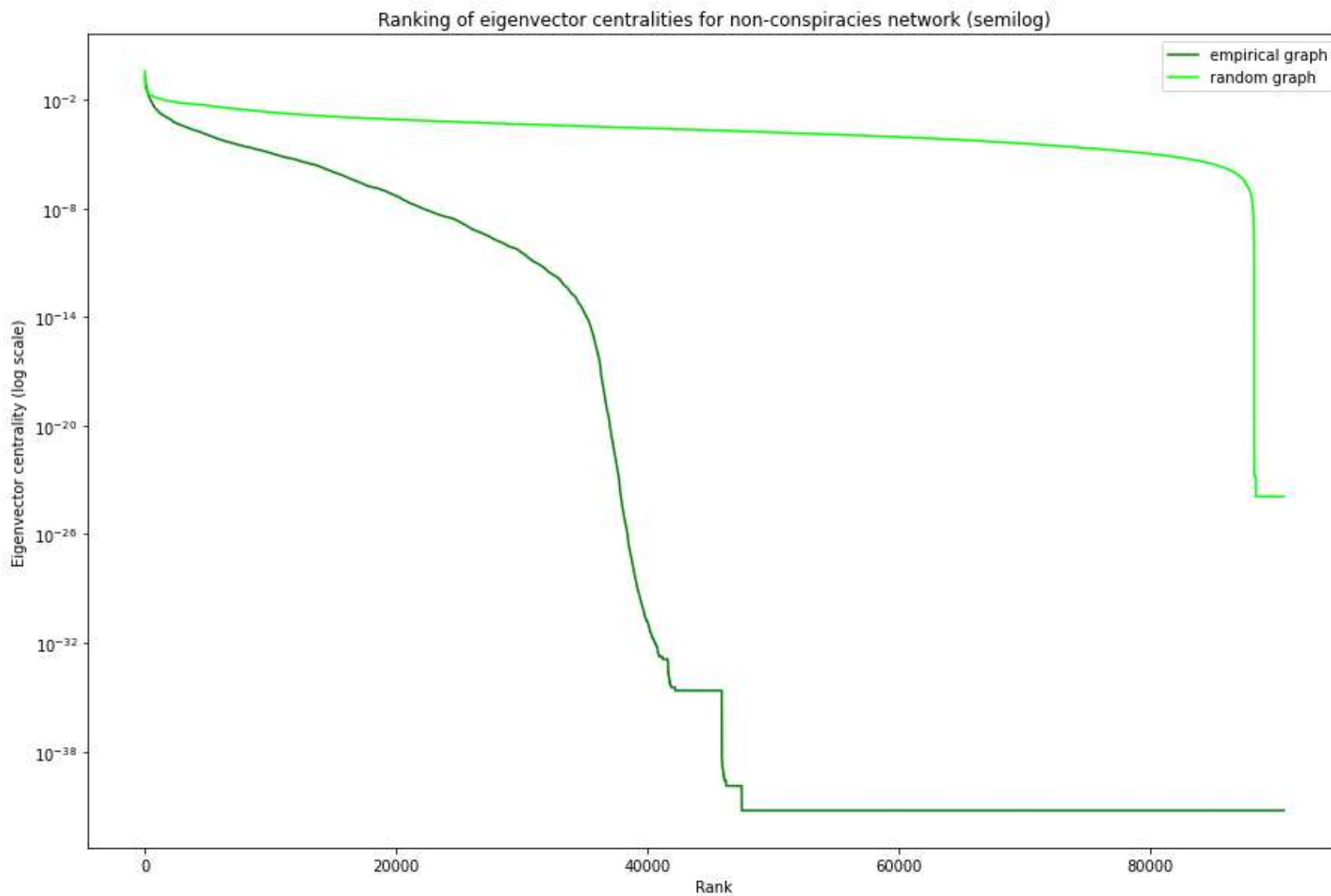
**Figure 51:** Rankings of PageRank for empirical and random versions of non conspiracies network. Semilog scale.



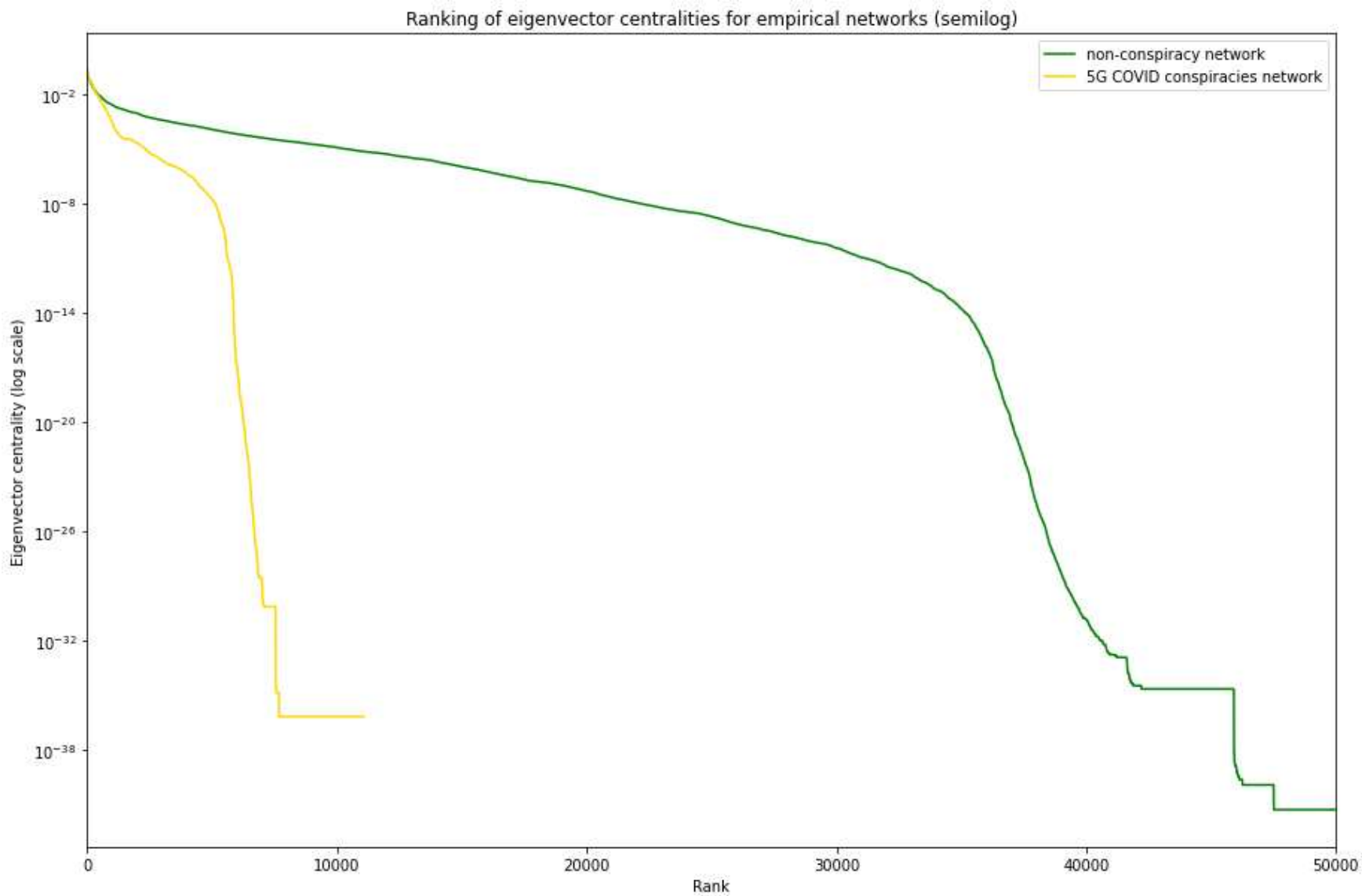
**Figure 52:** Rankings of PageRank for empirical 5G COVID conspiracies and non conspiracies networks. Semilog scale.



**Figure 53:** Rankings of eigenvector centralities in empirical and random versions of the 5G COVID conspiracies network. Semilog scale.



**Figure 54:** Rankings of eigenvector centralities in empirical and random versions of the non conspiracies network. Semilog scale.



**Figure 55:** Rankings of eigenvector centralities for empirical 5G COVID conspiracies and non-conspiracies networks. Semilog scale.

### Small worldness

Small world networks are characterized by short mean path lengths (compared to the regular network), high clustering coefficient (compared to the random network), and degree distributions that are significantly different from the Poisson distribution.

As described in earlier sections, we find a power law distribution of node degrees in both the misinformation trust network and the information trust network. Networks that exhibit the power law distribution are also referred to as scale free networks [5.24]. Scale free networks exhibit interesting characteristics: they are robust to random failures even when the percentage of failed nodes or links is high, yet they are fragile to intentional attack wherein if the right nodes (highly connected nodes) are targeted, the network can be severed by removing only a small number of nodes or links. An implication of this concept for a misinformation trust network is that the diffusion of conspiracy theories or other misinformation could be curtailed relatively easily by deplatforming the most trusted sources of misinformation.

Consistent with findings that are typical for real world networks from other researchers, we find evidence for small worldness and scale freeness in both the misinformation trust network and the information trust network.

### **Conclusions and Future Work**

In this chapter we have presented an exhaustive examination of the structure and topology of online misinformation trust networks, comparing them to non misinformation online trust networks, and to random equivalent networks. Our analysis presents evidence

in support of the existence of small worldness in online misinformation trust networks. This finding has important implications for how to measure trust in these types of networks, and for designing trust management systems that rely on the transitivity property of trust.

We confirm [5.15]'s findings that eigenvector centrality serves as a simple yet useful and robust metric for measuring direct trust in online settings.

We find a relationship between higher levels of trust in an online network and the ability of misinformation to more readily spread, but we are unable to draw a conclusion from this as to causality and, if there is a causal relationship, what the direction of the causality is. Further investigation will be required to be able to answer these questions.

Although this work utilizes a dataset that specifically focused on COVID-19 misinformation on a specific platform (Twitter), we believe the findings are applicable and extensible to online misinformation generally. Future research should examine misinformation trust networks in other online settings, such as other social media platforms, online messaging platforms (like WhatsApp), in the comments sections of news websites, or simply in other domains on Twitter. In this study, we did not examine temporal effects of these misinformation trust networks. Future research will explore the temporal aspect of misinformation and trust. For example, it will be insightful to consider if and how different levels of trust affect the speed of spread of misinformation, or to investigate if there is a relationship between differing trust levels and the speed with which links form

(i.e., if the amount of time that lapses between engagements with a given tweet affects the ease with which trust relationships form).

Future work should also consider models for epidemic spreading on networks, which include temporal aspects as referenced in the previous paragraph. For example, a traditional SIS model could be adapted and prove useful for modeling epidemic spread of misinformation in online networks. In the classical SIS model, there are two compartments of the population: the “S” compartment represents nodes in the network that are susceptible to the contagion in question, but not currently infected, and the “I” compartment representing nodes that are currently infected with the contagion in question. In the SIS model, nodes can alternate between being susceptible (S) and infected (I), and back to susceptible. The SIS model takes as inputs the number of nodes (N), the number of initially infected nodes (I), the probability of contact with an infected node ( $\beta$ ) and the probability of recovery ( $\gamma$ , i.e., transitioning from I back to S) [5.19]. We posit that beta can be estimated using the proportion of misinformation nodes in the overall network, while gamma can be approximated using the rate of contact of broker nodes (discussed in depth in an earlier section of this chapter) with the rest of the population. We also posit that the level of reciprocity in a misinformation trust network should have a direct effect on the recovery rate, gamma. In [5.25] we see evidence supporting the idea that higher reciprocity leads to echo chambers. We hypothesize that within the context of this chapter higher levels of trust should lead to a higher beta (infection rate) because nodes are more likely to come into contact with the infection (in this case, an artifact of



misinformation – a tweet), while simultaneously causing a lower gamma (recovery rate) because nodes, once infected, are more likely to remain connected with similar nodes (the echo chamber effect).

Finally, future research should also explore generative network models for reproducing online misinformation networks. If existing generative network models are unable to adequately reproduce the generative process for online misinformation networks, efforts should be made to develop a generative model that mimics the process for generation of online misinformation trust networks. Doing so will provide scholars and practitioners with additional insights into how these networks grow and evolve, and more importantly they may be useful in helping to slow the spread of online misinformation when paired with other methods like AI/ML models that identify misinformation.

## **VI. The relationship between trust and security in open source development**

### **Introduction**

To improve our understanding of the relationship between trust and security in open source development projects online, we turn to GitHub. GitHub is an online service for hosting software development repositories, with nearly 70 million users and tens of millions of code repositories [6.1]. GitHub hosts both private repositories (used internally for organizations who don't want to open source their software) and public repositories (used for open source software development and control). In an open source project, if a developer wants to propose a change (via a commit) to existing software code, they can initiate this process by creating a pull request. Different projects have different requirements, rules, or expectations for how to submit a pull request to their project, but in general pull requests describe the change being proposed and give their reasoning for the proposed change. From here, a conversation (in the form of a threaded discussion, as is typical in many online forums) can ensue between the developer who opened the pull request and the developers who administer the project in question and have authority to merge commits. Typically, these conversations are a back and forth between the originator of the pull request and one or more project administrators. If the originator of the pull request presents a convincing argument for why the code base should change, the

administrators of the project will approve the commit. With this process in mind, we can extract data from GitHub that are useful for modeling the trust networks of these online open source software development projects. Later in this chapter, we detail how we construct a dataset and graphs from these online interactions in GitHub's pull request system.

## **Background**

Free and open-source software (FOSS) projects provide critical technological infrastructure for organizations of all sizes, in countless industries, around the world. Examples of widely used FOSS include the Apache HTTP server, the nginx server, the Python programming language, the Linux kernel, and all the related Linux operating systems. A primary working method for producing and distributing FOSS is through creation, modification, and tracking of documents. Typically, software code begins its life as a document – text, in the form of a high-level programming language that human engineers can readily understand, share, modify and save in a document. Additionally, FOSS projects often feature not just a single document of code, but rather multiple code documents that reference one another – for example, a “main.py” and a “config.txt” document – in addition to ancillary documents such as licensing, readme, admin, logs, and others. Thus, FOSS projects can be modeled as sociotechnical systems or networks of developers and documents, organizing and giving structure to the work of FOSS projects.

Developers and contributors to FOSS projects are the primary intended users of the resulting document collections. Nonetheless, because these documents are public and

easily accessible, they also open themselves to scrutiny and attempted attacks by attackers (blackhat hackers). Blackhat hackers seek to identify and exploit vulnerabilities in software (or hardware). This software may be proprietary, or it may be open source. Attackers' goals are varied, ranging from criminal to political to bragging rights. One advantage attackers have when seeking vulnerabilities in FOSS is they generally have full access to source code, with full understanding of the structure and logic of the software, and knowledge of how data flows through and is transformed by the code. A 2021 report by Synopsys found 84% of FOSS code bases had at least one reported security vulnerability, and the average number of vulnerabilities was 158 per codebase [6.2]. This issue is compounded by the fact that FOSS projects typically are not standalone entities, but instead rely on other FOSS components to complete tasks; according to the same Synopsys report [6.2], an average software application today relies on 528 open-source components.

FOSS projects are by their nature highly collaborative. This collaborative creation and modifying of documents in a public forum allow for data to be collected and analyzed related to the dynamics of the development project over time. Collaborators, the documents they produce, and their interactions in any given FOSS project form a graph, or network. Naturally, these graphs feature varying degrees of formality and complexity, depending on the specific FOSS project. If we think about these interactions as a graph or network, we can then apply a variety of network analysis techniques to uncover insights about their behavior.

One such network analysis method that computer scientists have developed for application in information and communication systems (principally the Internet and the Web, though there are others) is *trust metrics and trust management systems*. Trust is a concept that is difficult to objectively measure, but which nonetheless can be critical for realizing improved performance of an engineered system. Trust affects the speed, quality, efficiency, and/or success of transactions between people, among groups of people, from person to machine, from machine to machine, or any combination of these. Therefore, having ways to estimate or model different types and levels of trust becomes useful.

### **Related Work**

[6.3] considers how trust affects contributors to open source software projects. The authors identify two primary types of motivations for open source contributors: intrinsic motivations and extrinsic motivations. Intrinsic motivations can be enjoyment based or obligation based. Based on these two broad categories (intrinsic and extrinsic) the authors propose five types of open source contributors based on their motivations: 1) commercial service providers (such as Red Hat), which are extrinsically-motivated, 2) software customizers, who are also extrinsically motivated, 3) reputation investors (people who seek to become well-known for their contributions), also extrinsically-motivated, 4) *Homo Ludens* “just for fun” contributors, who are intrinsically-motivated, and 5) those who are members of the tribe “helping others”, “giving back”, also intrinsically-motivated. Intrinsically-motivated contributors are particularly important in the starting phase of a new project, because they are the ones most likely to expend the time, effort, and attention needed to

create something new. The authors state that trust is a critical ingredient in enhancing cooperation and success of open source development projects, and that trust in this context tends to be more institutional than personal. Because many open source projects consist of many different contributors who can enter and exit whenever they wish, the development of trust in open source projects doesn't rely as much on repeated personal interactions as it does in other contexts. The authors posit that trust in open source software development instead takes the form of swift trust, which is a type of trust observed in teams that work together for a limited amount of time. Within swift trust, the authors identify two types: encapsulated interests, and cognitive trust. Encapsulated interests takes a rational approach, described by "Alice trusts Bob because Alice believes it is in Bob's best interest to trust Alice." Cognitive trust is based on an estimation of the characteristics of the people that are working together on a project. The authors state that development and presence of trust is critical in attracting new members because new members will be concerned about the ability of the group to resolve conflicts suitably. Because of the nature of open source software development, potential new members can easily observe the behavior of the group, and thereby make their own estimates of how much to trust (or not) that group. On the side of the trustor, the authors state that "extrinsically motivated trust based on encapsulated interests is sufficient". On the side of the trustee (the sink of the trust), there needs to be a certain minimum number of intrinsically-motivated contributors to enable development of swift trust from trustors. The authors recognize that institutions must be in place to encourage participation of

intrinsically-motivated contributors, and ensuring that the contribution costs are low enough to ensure that even intrinsically-motivated contributors don't leave.

Researchers have successfully derived valuable insights into the behavior of complex systems, including information and communication systems, by modeling them as graphs or networks. A network consists of nodes, which are “participants” in a network, and edges which are “interactions” among the participants (nodes) of the network. With this as the foundation for analysis, we can apply analytical techniques to a network to draw insights about its structure and behavior.

Scholars have developed numerous methods for estimating trust in diverse information and communication systems, such as online social networks, peer-to-peer (P2P) file sharing networks, encryption schemas (i.e., PGP), and others. Trust is inherently an amorphous and subjective concept, and thus it takes on different meanings in different contexts and is difficult to measure precisely. Nonetheless, as a developed area of research, trust scholars have proposed and defined several different types of trust. For our purposes, we take trust to mean the ability of one party to rely on the accuracy and integrity of another party's work – namely, the accuracy and integrity of pull requests in an online collaborative software development environment, GitHub.

Much research has been made into questions of cybersecurity in open source software projects. Approaches have considered if and how the complexity of code affects its security, if and how the number of contributors to an open source project affects its

security, the application of the software in question, where the software in question is deployed (i.e., Web-based versus embedded or desktop) [6.4].

Comparatively little work has considered the intersection of trust metrics and online FOSS projects. [6.5] proposed a system that incorporated a “karma” value of developers on open source projects which was derived from the developer’s upvotes and activity. [6.6] present an extensive system that combines machine learning and graph techniques to estimate trust among developers in GitHub. They first gather data from pull requests on GitHub and then apply a natural language processing model to the comments contained in the pull requests to perform sentiment analysis. With the results of the sentiment analysis, the authors translate positive sentiments to high trust values and negative sentiments to low (or zero) trust values. They then use these direct trust values (direct trust between a pair of developers) together with graph based techniques to estimate indirect trust among developers who have never interacted.

### **Research Questions**

To gain a clearer understanding of the role played by trust in online open source development projects, we propose the following research questions, which we investigate throughout this chapter:

***RQ6-1: What is the structure and topology of a trust network in free and open source software (FOSS) projects?***



**RQ6-2:** *Does trust lead to the network structures that we see in online FOSS networks, or do FOSS networks by their nature give rise to the types of trust relationships that we observe?*

**RQ6-3:** *What is the relationship (if any) between the trust network and security incidents in FOSS projects?*

To answer these research questions, we first construct and then utilize a dataset of interactions in two different FOSS projects hosted on GitHub, both of which are aimed at Internet security (cryptographic functions and encryption). In the following section, we first describe our construction of a dataset from GitHub, then describe the analysis methods we apply to this dataset to improve our understanding of the role of trust in online open source software development.

## **Methods and Data**

Different from Chapter V where we utilized an existing dataset, for Chapter VI we collect our own raw data and transform it into a usable dataset. In this section, we describe our methods for creating the dataset, as well as our methods used to then analyze the dataset in pursuit of our research questions.

### Dataset: rationale, data collection and dataset construction

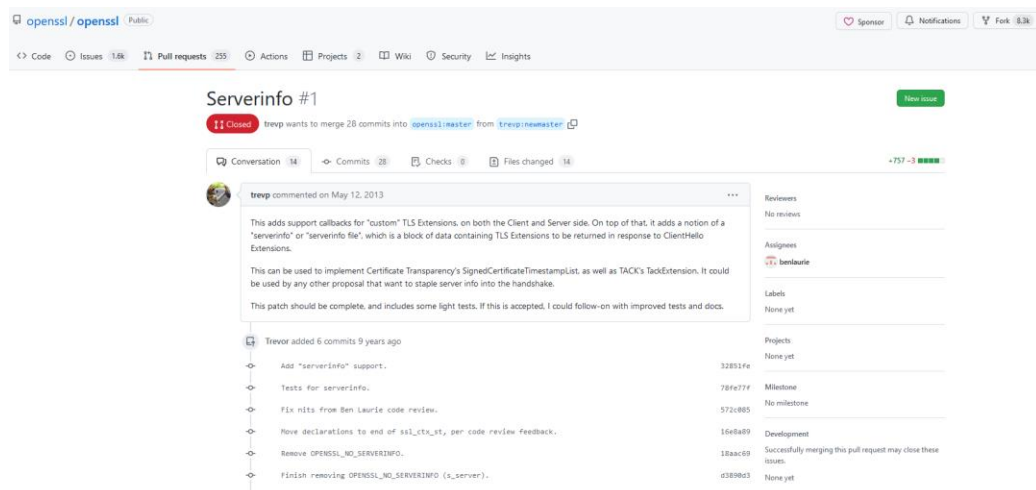
In this section, we describe our methods for creating the dataset, as well as our methods used to then construct graphs from the dataset and to analyze the dataset via these graphs.

#### Rationale

Before collecting any data, we needed to select useful sample projects to be able to answer our research questions. To do so, we begin our process by reviewing the list of disclosed CVEs (Common Vulnerability and Exposures, operated by The Mitre Corporation and the US National Cybersecurity FFRDC) for open source software projects that are available for public review in GitHub. We begin our exploratory analysis by reviewing the “MSR 20 Code Vulnerability Dataset” (hereafter referred to as the MSR dataset) from [6.7]. The MSR dataset includes 122,774 CVE entries from 2002 to 2019, each of which includes 21 features. Features included commit ID (in a specific code repository), CWE (common weakness enumeration) ID, CVE ID, a link to the CVE description Web page, CVSS (common vulnerability scoring system), the (primary) programming language used in the project, what the project was, and several other features. While the full MSR dataset includes more than 100,000 records, the subset of data that represents CVEs associated with a known FOSS project on GitHub includes 4246 records.

We selected 2015 as our sample year, as it is far enough in the past that new developments related to CVEs disclosed in that year should be finished, but recent enough to still be able to draw useful parallels to FOSS projects of today. In 2015, there are 374

CVEs represented in the MSR dataset, and of these there were seven that were specifically associated with OpenSSL, which is a project whose repository is publicly hosted on GitHub. OpenSSL's contributors describe it as "a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library" [6.8]. OpenSSL helps to secure billions of Internet based transactions every day, around the world. Because OpenSSL is an open source project and is hosted on GitHub, we can observe revisions of documents associated with the project over time, and interactions among collaborators and documents. Moreover, because OpenSSL is software that is specifically designed for use on the Internet (as opposed to embedded software, enterprise software, etc.) – the focus of this dissertation – we identify it as a good candidate for our analysis.



**Figure 56:** An example of a pull request on GitHub. This pull request is the first public pull request for the OpenSSL project, from 2013.

To provide a useful comparison project, we seek a project that is also available for observation in GitHub, primarily intended for Internet-based use, and has comparable

reach or impact to that of OpenSSL. With these criteria in mind, we identify Let's Encrypt as our comparison project. Let's Encrypt is "a free, automated, and open certificate authority" [6.9] started in 2012 and is today the world's largest certificate authority. Let's Encrypt was later brought under the umbrella of the Internet Security Research Group (ISRG), and is an official collaborator with the Linux Foundation. In 2015, Let's Encrypt's ACME protocol (which verifies that an actor who claims to control a domain name actually controls that domain) was submitted to the IETF (Internet Engineering Task Force) for inclusion as a standard; it was subsequently rolled into an RFC (request for comment) as a proposed IETF standard. As of the writing of this dissertation, Let's Encrypt issues more than two million certificates each day, and more than 290 million active certificates in total [6.10]. Let's Encrypt's code base is, like that of OpenSSL, available publicly on GitHub and thus, we can observe changes and developer interactions that happen in it. In 2015, there were no known CVEs associated with the Let's Encrypt project, though there were subsequent vulnerabilities discovered. For example, in 2020 a vulnerability was announced which required Let's Encrypt to revoke outstanding security certificates generated with the vulnerable code, leading to millions of certificates being revoked [6.11]. Nonetheless, as there were no known CVEs associated with Let's Encrypt in 2015, we confirm it as a useful comparison project.

### Data collection

With our projects selected, we move forward with collecting data associated with the projects. Frequently in a FOSS project, an author of a pull request (PR) will propose and

describe his change, and one or more reviewers will evaluate the proposed change. Often there will be one or more rounds of discussion between the initiator of the PR and the reviewer(s). If a PR is accepted, the change is adopted into the codebase with a commit. Thus, we collected data on collaborators who proposed, reviewed, and accepted changes to code in these documents, and each time an interaction resulted in a commit to the codebase, we count this as a trusted transaction, resulting in a trust (sub)graph connecting all participants in that particular PR. Of note, because of the way we define what constitutes a satisfactory interaction, there are no “negative” interactions in our case – only positive or neutral ones.

GitHub provides an archive accessible to the public of all activity associated with public repositories hosted on the platform, with data provided from more than 20 event types represented in the data. The data are housed at [6.12] and can, among other methods, be accessed using Google BigQuery (Google’s cloud data warehouse). We identify features of interest for our analysis: 1) all pull requests for each project within the analysis timeframe, 2) all users who interact within each pull request, 3) the originator and the closer of each pull request, 4) the outcome of the pull request (if the pull request results in a commit or not), which we will use to infer trust resulting from each interaction, and 5) the documents that were modified in each successful pull request (for example, “challenges.py”), and 6) only considering closed pull requests (most of the pull requests created in 2015 for both projects have been closed, but in both cases there are a smaller

number of pull requests that are still open, and since we rely on the outcome of the pull request to infer trust, we can't do so from open pull requests).

### Dataset construction

From the raw data we next aim to construct a dataset that will be usable for network modeling and analysis purposes. We semi-automatically sort and organize the raw data to put it in the order and structure that will result in a meaningful dataset for our purposes using a combination of Python code and manual methods.

We first order the data in descending order of chronological occurrence (i.e., when was a comment posted), and sort the ordered data by pull request. This results in groups of pull requests in which comments and actions that take place within a given pull request are sorted chronologically. Next, both to simplify analysis and also to provide a basic level of privacy safeguard for the GitHub users and documents represented in our dataset, we generate random numbers and assign these numbers to a *user\_id* field to developers, and a *doc\_id* field to documents. Figure 57 provides a sample of what the formatted dataset looks like before transforming it into a format that is more amenable to graph analysis. In Figure 57, each row represents a linked list [6.13]. In our case, these linked lists are chains of interactions that took place within a given pull request, with the leftmost node being the originator of the pull request, and the rightmost node being the closer of the pull request.

```
19483 12706 19483 13096 19483 12706 19483 16009
```

**Figure 57:** Sample linked list constructed from a thread of comments in one pull request

## Inference of trust from dataset

Before continuing our dataset formatting work, we need to ensure we capture approximations of trust in the network. To a limited extent, we can make the assumption that comments (and therefore links/edges) between developers represented some very basic levels of trust. However, beyond this simple measure we reason that if a pull request results in a successful commit (i.e., the changes being proposed by the originator of the pull request are accepted by the project administrators) we can infer there is some level of trust among the developers represented in the comments thread of that pull request. Conversely, in pull requests where the proposed commit is rejected, we assume a lack of trust among the developers represented in that thread; importantly, with the data that we consider, lack of trust is not necessarily the same as mistrust or distrust [6.14], [6.15]. Thus, when a pull request results in a successful commit, we assign a trust value of "+1" to the edges in the subgraph represented by that pull request. When a pull request results in a rejection of a proposed commit, we assign a trust value of 0 to the edges in the subgraph represented by that pull request.

Next, to be able to construct graphs from our dataset (described in the following section) it will simplify our analysis if we first format our data into a structure that can be readily manipulated by our network analysis software (NetworkX). For our purposes, we identify the edge list data format as the most fitting. Doing so makes our own analysis easier to perform, but it will also result in a more useful dataset to share with other researchers, as an edge list is one industry standard data format for network analysis. Over

a series of rows, an edge list makes note of pairs of nodes that are connected, with the first node listed typically being the source node and the second node typically being the target node (in the directed case), and the two separated by a delimiter (a tab, in our case). To properly create the edge list, we can't simply separate the rows at every second tab, because doing so would miss half of the connections represented in each row. Instead, we must iteratively read through each row, writing to a list the first node as a source node and the second node as a target node, then advancing rightward by one node with what was the target node now becoming the source node, and the node to its right becoming the target node. Figure 58 provides the iterative loop that was written in Python to generate the edge list from the linked chain of pull request interactions.

```
for i in openssl_list: #build an edges list from a linked list
    index_i = 0
    for j in i[1:]:
        openssl_edges.append([i[index_i],j])
        index_i += 1
```

**Figure 58:** "For" loop written to transform linked lists of pull request interactions into edge lists useful for network analysis

After applying the iterative loop presented in Figure 58, we obtain an edge list with each row of the list containing two nodes: the first node a source node, and the second node a target node. Figure 59 illustrates what the resulting edge list looks like after transforming the linked list shown in Figure 57.



```
19483 12706
12706 19483
19483 13096
13096 19483
19483 12706
12706 19483
19483 16009
```

**Figure 59:** Sample edge list, transformed from the linked list in Figure 57

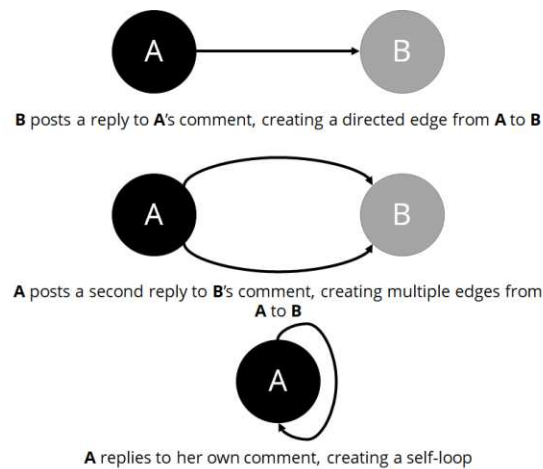
Once we have constructed edge lists from the dataset, we are now ready to begin analysis, beginning first with construction of graphs which we describe in the next section.

#### Trust graph construction: developer-developer network

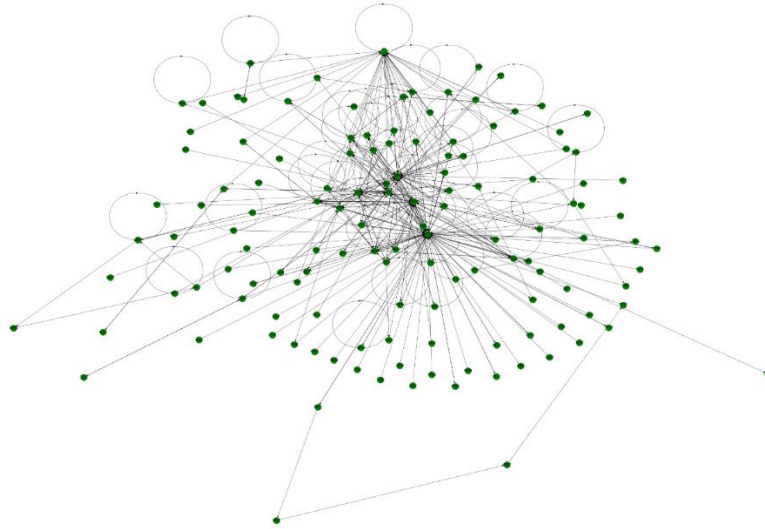
With our dataset ready for use, we begin by constructing graphs from the dataset. We construct multiple graphs with different purposes in mind for each. Graphs are constructed using the NetworkX Python library (versions 2.6 – 2.8.x, as multiple updates were released over the course of drafting of this chapter) [6.16]. An important difference between the methodology described in this chapter and that of Chapter V is that these graphs are by definition and construction trust networks because of the data we collected and how we collected it; in Chapter V, we had to extract a trust overlay network from the basic structural network. Thus, the following descriptions of the graphs are all of trust networks.

First, from the dataset we construct multigraphs from the OpenSSL project and the Let's Encrypt project (referred to hereafter as LE). A multigraph is a graph in which parallel edges (multiple edges) are allowed between pairs of nodes, and self-loops (edges where a node connects to itself) are also included. In the context of this chapter, more than one

edge between a pair of nodes indicates multiple interactions between them in the comments section of a pull request, while a self-loop indicates a developer responded to his or her own comment before someone else did (often to give additional context or details about their previous comment). In Figure 61, we provide an example illustration of a real network that includes self-loops. Because self-loops (replies to one's own comments) don't indicate any changes in trust, nor do they indicate any presence or lack of trust between pairs of nodes, we remove self-loops from our empirical graphs.

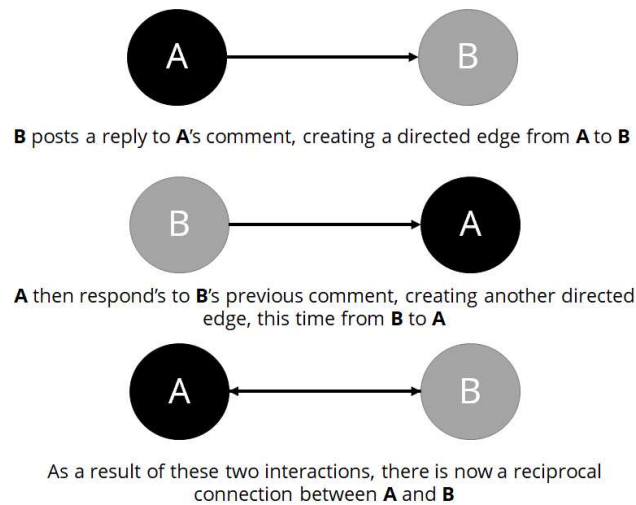


**Figure 60:** Conceptual illustration of the construction of directed versions of the graphs



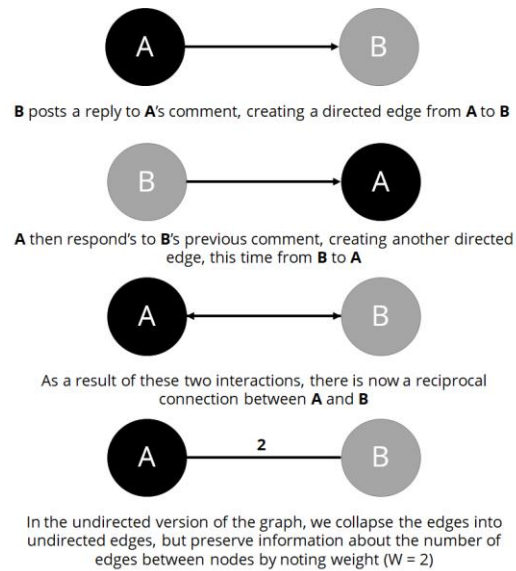
**Figure 61:** Example graph with self-loops. Generated in the Python language using the NetworkX and matplotlib libraries

We also construct directed graphs of the same networks. In the directed version of the graphs, edges have direction. Based on how we collected our data, when a developer posts a comment or reply to another developer, the developer who is making the comment receives an inbound edge while the developer who is being replied to receives an outbound edge. Construction of the directed graphs will be useful for analyzing reciprocity in the networks, and for constructing our trust overlay network (a virtual, trust-based network that sits on top of the basic, structural network).



**Figure 62:** Conceptual illustration of the construction of directed versions of the graphs

Finally, we construct undirected versions of the graphs for both projects. In the undirected versions, edges have no directionality and are collapsed into simple edges. However, we tabulate information about edge weights to preserve this information for later analysis, creating a “weights” attribute in our graphs. Generating the undirected versions will be useful for other types of analysis in which directed edges are either not allowed or which will result in intractable calculations, for example, in computing network diameters.

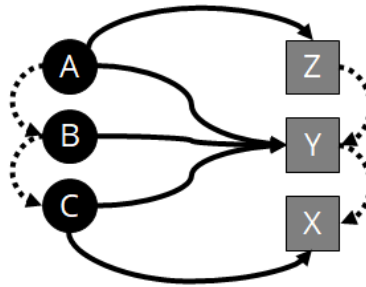


**Figure 63:** Illustration of representation of reciprocity and weighted edges in our dataset construction

Trust graph construction: bipartite network of developers and documents

In addition to the graphs representing developer-developer interactions, we also construct bipartite networks in which developers are indirectly connected to one another as a result of the documents that they modify (and vice-versa), rather than being connected through their threads of discussion in a pull request. As in the previous descriptions of the constructed graphs, we infer trust levels by examining the result of the pull request (approved or rejected commit) [6.14], [6.15]. Constructing bipartite networks allows us to consider the levels of trust placed in documents – and by extension, the code and resulting software itself – arising *indirectly* from the trust levels of the developers who modify them. Figure 64 illustrates a simple example of such a bipartite developer-document network. In this example, **A**, **B** and **C** (all developers) are connected to one another indirectly as a result of the changes they each make to **Z**, **Y** and **X** (all documents).

Developer **A** edits document **Z** and **Y**  
Developer **B** edits document **Y**  
Developer **C** edits document **X** and **Y**



**Figure 64:** Illustration of bipartite network representation; round nodes are developers, square nodes are documents.

### Random graph generation (configuration model)

To gain a better understanding of whether the structure and behavior we observe in the networks are caused by random processes or if they are caused by a specific mechanism or mechanisms, we construct random equivalents of the empirical graphs discussed in the previous sections. To provide a useful model for comparison, we generate a set of random networks based on the empirical networks, using the configuration model. The configuration model creates a random graph using a given degree sequence as input [6.17]. In this chapter, we take this degree sequence to be that of the empirical networks. In this type of configuration model, the degree of each node is fixed, which directly leads to the number of edges in the network being fixed, too. We generate 100 random networks using the configuration model for each of the graphs described previously for both the OpenSSL project and the LE project. The results of the 100 random networks are averaged when compared against the empirical networks. In Table 21, we provide a summary of all

of the graphs that are generated for use in this chapter, each of which is useful for a different type of analysis.

**Table 21:** Summary of graphs generated from the OpenSSL and LE projects for analysis in this chapter

<b>Summary of graphs constructed for analysis</b>	
G_openssl_multi	Empirical OpenSSL multigraph
G_openssl_dir	Empirical OpenSSL directed graph
G_openssl_undir	Empirical OpenSSL undirected graph
R_openssl_multi	Random OpenSSL multigraph
R_openssl_dir	Random OpenSSL directed graph
R_openssl_undir	Random OpenSSL undirected graph
G_certbot_multi	Empirical Let's Encrypt multigraph
G_certbot_dir	Empirical Let's Encrypt directed graph
G_certbot_undir	Empirical Let's Encrypt undirected graph
R_certbot_multi	Random Let's Encrypt multigraph
R_certbot_dir	Random Let's Encrypt directed graph
R_certbot_undir	Random Let's Encrypt undirected graph

### Analysis measures

As in Chapter V, we consider reciprocity, various centrality measures, assortativity, and standard summary statistics measures for this network. We refer the reader back to Chapter V for additional detail on methodology related to these measures. In addition to those measures already used and described in Chapter V, we apply other measures in this chapter which we describe in this section. These measures include modularity, community detection, and a transitive trust calculation.

### Modularity and community detection (clustering)

In this chapter, we apply an additional technique that wasn't used in Chapter V, community detection. Community detection is useful to identify what community structures, if any, exist within a graph [6.17]. In the context of this chapter, this could mean identification of communities with higher or lower values of trust within them as compared

to the graph as a whole. Myriad community detection algorithms have been proposed by researchers, and they are similar to graph partitioning methods with the difference that in community detection algorithms there is no set number of partitions that is being sought. Most community detection algorithms work in an iterative fashion, proceeding until a maximum threshold is reached. For our purposes, we utilize the greedy modularity maximization proposed by [6.18]. Greedy modularity maximization community detection starts with each node in the network as its own standalone community with successive joins of communities in an iterative fashion until the modularity ceases to increase. Modularity measures the prevalence of like nodes that connect to other like nodes in the network, while greedy algorithms focus on locally optimum (as opposed to globally optimum) decisions. In [6.18] the authors define modularity as in Equation 8, and its resulting maximization as in Equation 9.

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

**Equation 8**, from [6.18], where **Q** is modularity, **m** is the number of edges in the network being analyzed, **A** is an element of the adjacency matrix representing the network in question, **k** is the degree of a node **v** or **w**, and **c** are the communities detected in the network

$$\sum_i (e_{ii} - a_i^2)$$

**Equation 9**, from [6.18]



## Trust metrics

In this section, we discuss the measures we utilize in this chapter to estimate trust values. We discuss measures for direct trust estimation and indirect trust inference.

Direct trust is estimated using the same methods as in Chapter V (eigenvector centrality, PageRank, and degree); we refer the reader back to Chapter V for additional detail on these measures.

We estimate indirect trust (transitive trust) using our own transitive trust measure, which we construct by modifying the methods proposed in [6.19] to fit our networks. We refer to this method simply as generic transitive trust (GTT). Our method is as follows. As was discussed in Chapter III, most transitive trust metrics include a trust propagation function, a trust aggregation function, and a trust decay function. Our method includes each of these elements. For the trust propagation function, we adapt the method in [6.19] wherein the authors, upon verification of small worldness of their networks, set the maximum trust propagation distance (MTPD) as the ceiling of the mean shortest path length. In [6.19] the authors verify that the small decrease in accuracy of their trust metric resulting from setting MTPD equal to the mean shortest path length is negligible, and that the gains in terms of speed of computation are significant. Once we compute the mean shortest path lengths (discussed in a subsequent section), we round this value up to the nearest whole number and set this as the MTPD. In both trust networks, the MTPD equals 3. Thus, trust in the networks analyzed in this chapter is propagated along all paths between every pair of nodes that are equal to or shorter than the MTPD. To define our

trust decay function, we adapt a method described in [6.6]. In our case, we set the trust decay as equal to the reciprocal of the distance between source and sink such that a distance of 1 produces a trust value of 1 (1/1), a distance of 2 produces a trust value of 1/2 (reciprocal of 2), a distance of three produces a trust value of 1/3 (reciprocal of 3), and so on. Finally, for our trust aggregation function, we aggregate all trust values for all nodes in each network into a raw (non-normalized) trust value, which is simply the sum of all of the individual transitive trust values placed in each sink node. As is typical for many other trust measures, such as in EigenTrust, we then normalize our trust estimates to make them more readily comparable across networks. We normalize the trust measures by dividing the raw trust score (the sum total of all trusts given to a particular sink node) by the total possible trust for that node. For example, if a node (based on its number of connections and paths within the network) has a maximum possible raw trust score of 10 and its actual estimated raw trust score is 8, we divide 8/10 to obtain a normalized trust score of 0.8. As in many other trust metrics, trust values can only take values from 0.0 to 1.0, and a value of 0.0 does not necessarily signify *distrust* but rather *lack* of trust; a value of 1.0 would signify complete trust; and values falling between these two extremes indicate an intermediate level of trusting or untrusting.

## **Results and Discussion**

In this section, we present the results of our analyses and give interpretation of their meaning. We first present a summary of the key findings from our analyses, and then

discuss these results and their interpretations in greater depth with respect to this chapter's research questions.

We find evidence for small worldness in both the OpenSSL trust network and the LE trust network. This finding allows us to apply our simplifying assumption of setting MTPD equal to the ceiling of the mean shortest path length in each network when computing transitive trust.

We present evidence that the traditional centrality measure of eigenvector centrality is fast and effective for measuring direct trust in nodes, confirming the findings from [6.20]. At the same time, considering only direct trust leaves out a large piece of the overall picture, and thus, a transitive trust measure is needed as well.

Similar to in Chapter V, we find an important difference in the distribution of eigenvector centrality measures in the OpenSSL trust network compared to the LE trust network, and propose that this difference could serve as a useful proxy indicator for higher or lower levels of trust in a network.

We find generally higher levels of trust in the OpenSSL network compared to the LE network when using several different trust measures. With several caveats (discussed later in this chapter), we propose that contrary to our original suspicion higher trust levels in a FOSS project may be correlated with higher frequencies of security issues as measured by CVEs.

### Summary statistics

In Table 22, we present basic summary statistics of the trust networks as modeled in this chapter. Compared to the networks analyzed in Chapter V, the networks are relatively small; the OpenSSL trust network features 132 nodes with 1043 (undirected) edges and a mean degree of 7.9, while the LE trust network features 263 nodes with 3255 (undirected) edges and a mean degree of 12.4.

**Table 22:** Summary Statistics for OpenSSL and LE trust networks

	OpenSSL trust network	LE trust network
# nodes $n$	132	263
# edges $m$	1043	3255
Average degree $\langle k \rangle$	7.9	12.4

Both networks are fully-connected, which means that all nodes can be reached by all other nodes if ignoring directionality, and thus, the connected component size for both networks is simply equal to the number of nodes  $n$ .

In Table 23, we present key structural measures of the empirical and the random versions of both the OpenSSL and the LE trust networks; we interpret these measures in greater detail in the following sections.

**Table 23:** Key structural measures for empirical and random versions of OpenSSL and LE trust networks

	OpenSSL trust network		LE trust network	
	Empirical	Random	Empirical	Random
Density	0.121	0.268	0.095	0.219
Diameter	5	4	7	4
Degree assortativity	-0.289	-0.890	-0.193	-0.876
Reciprocity	0.561	0.331	0.468	0.334

### Degree assortativity

In Table 23 we see the degree assortativity for the empirical and random versions of the OpenSSL and the LE trust networks. Comparing the empirical networks, we see that in both cases the empirical networks have negative degree assortativity, with the OpenSSL trust network showing a more negative degree assortativity than the LE trust network. The interpretation of this is that in both networks nodes have a slight preference for connecting with nodes that are different from them in terms of degree, with the OpenSSL nodes having a stronger preference than the LE nodes. When comparing the empirical and the random equivalent graphs for both the OpenSSL and the LE trust networks, we see a significant difference in degree assortativity between empirical graphs and ones generated by random processes. This serves as an indication that non-random mechanisms led to the rise of these networks.

### Density

Referring back to Table 23, we see the densities of each of the empirical and random networks. Comparing the OpenSSL and the LE empirical networks, we see similar densities in both networks, with the OpenSSL trust network featuring slightly higher density than the LE trust network. When comparing each empirical network to its random

equivalent, we see that the empirical trust networks' density is approximately half that of the random equivalent networks. Once again, we take this as evidence that non-random processes gave rise to both empirical networks.

### Reciprocity

Table 23 presents reciprocities for the directed versions of our empirical trust networks and their random equivalents. From this table, we first compare the two empirical networks to one another. We find a higher level of reciprocity in the OpenSSL network than in the LE network, a possible indicator of higher levels of trust in the OpenSSL network as compared to the LE network. When considering the empirical networks compared to their random equivalents, in both cases (OpenSSL and LE) we also see a marked difference, indicating that it is likely that reciprocity plays a role in the behavior of both trust networks.

### Geodesic paths

In Table 24 we present the mean shortest path lengths (geodesic paths) for the empirical and random equivalent versions of the OpenSSL trust network and the LE trust network. Comparing the two empirical networks, we see very similar mean shortest path lengths of approximately 2.5. When comparing each empirical network to its random equivalent, we see a marked difference, once again providing evidence that a non-random processes have generated the trust networks. We interpret the difference between the random and the empirical networks' path distances as additional evidence that this type of trust network structure is not the result of random processes. Additionally, taking the

mean shortest path lengths presented in Table 24 we round them up to the nearest whole number (in both cases, 3) and take these to be the ceilings of the maximum trust propagation distance for use in the trust propagation function of our transitive trust metric described in an earlier section.

**Table 24:** Mean shortest path lengths for the empirical and random versions of the OpenSSL trust network and the LE trust network

	OpenSSL trust network		LE trust network	
	<i>Empirical</i>	<i>Random</i>	<i>Empirical</i>	<i>Random</i>
Average shortest path length	2.50	2.05	2.64	2.03

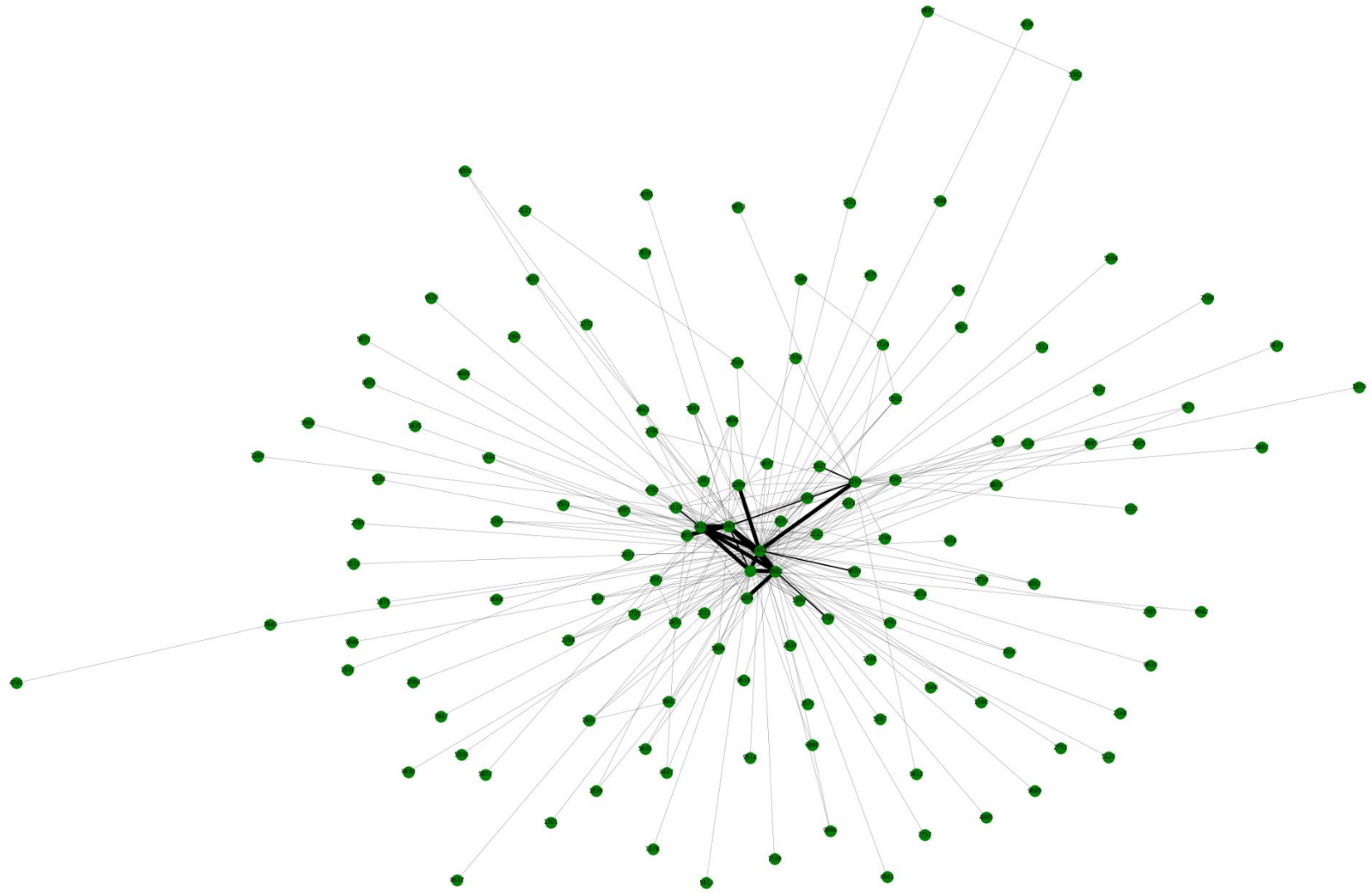
### Network visualizations

In Figures 65 and 66, we present visualizations of the empirical and random versions, respectively, of the OpenSSL undirected trust network. Figures 67 and 68 present visualizations of the empirical and random versions, respectively, of the Let’s Encrypt (LE) trust network. And, to gain a clearer sense of which nodes are in Figures 69 and 70 we present visualizations of the same networks with only those edges that have weight > 15, representing those nodes which based on their number of interactions may be the most influential nodes. Interestingly, despite their differences in scale (the LE network has twice the number of nodes as the OpenSSL network and more than three times the number of edges) in both networks we observe a very similar core size of highly-connected nodes, indicating a small group of trusted nodes (key contributors to the development project).

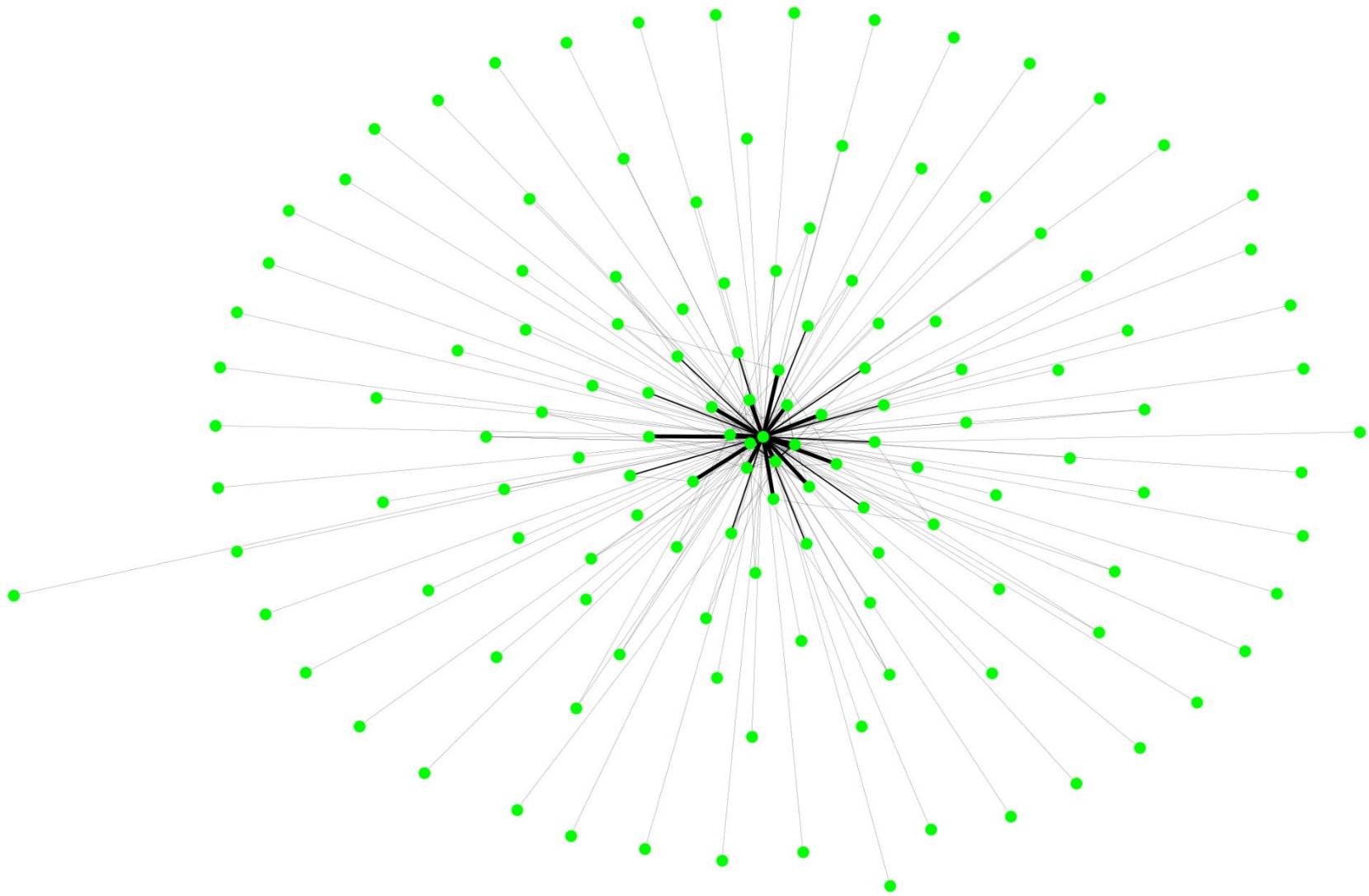
The visualizations were produced using the NetworkX and Matplotlib libraries in the Python language. Spring layout was used for both with  $k=0.33$  in all cases ( $k$  in the spring layout implementation of NetworkX is the optimal distance between nodes when visualizing them), and the width of edges is sized according to their weights, with the thicker edges representing a higher edge weight. The weights of edges represent multiple connections between the same nodes, i.e. multiple comments between the same developers will result in a higher edge weight between the two.

In both networks, we observe a core of a few highly connected nodes, with many more nodes that have few connections (small degree). Most interestingly, for both the OpenSSL and the LE trust networks we observe clear differences between their empirical and random versions. This indicates that the structures in these networks are not generated by random mechanisms, leading us to continue our consideration of trust as one possible factor that influences the structure and topology of these networks.

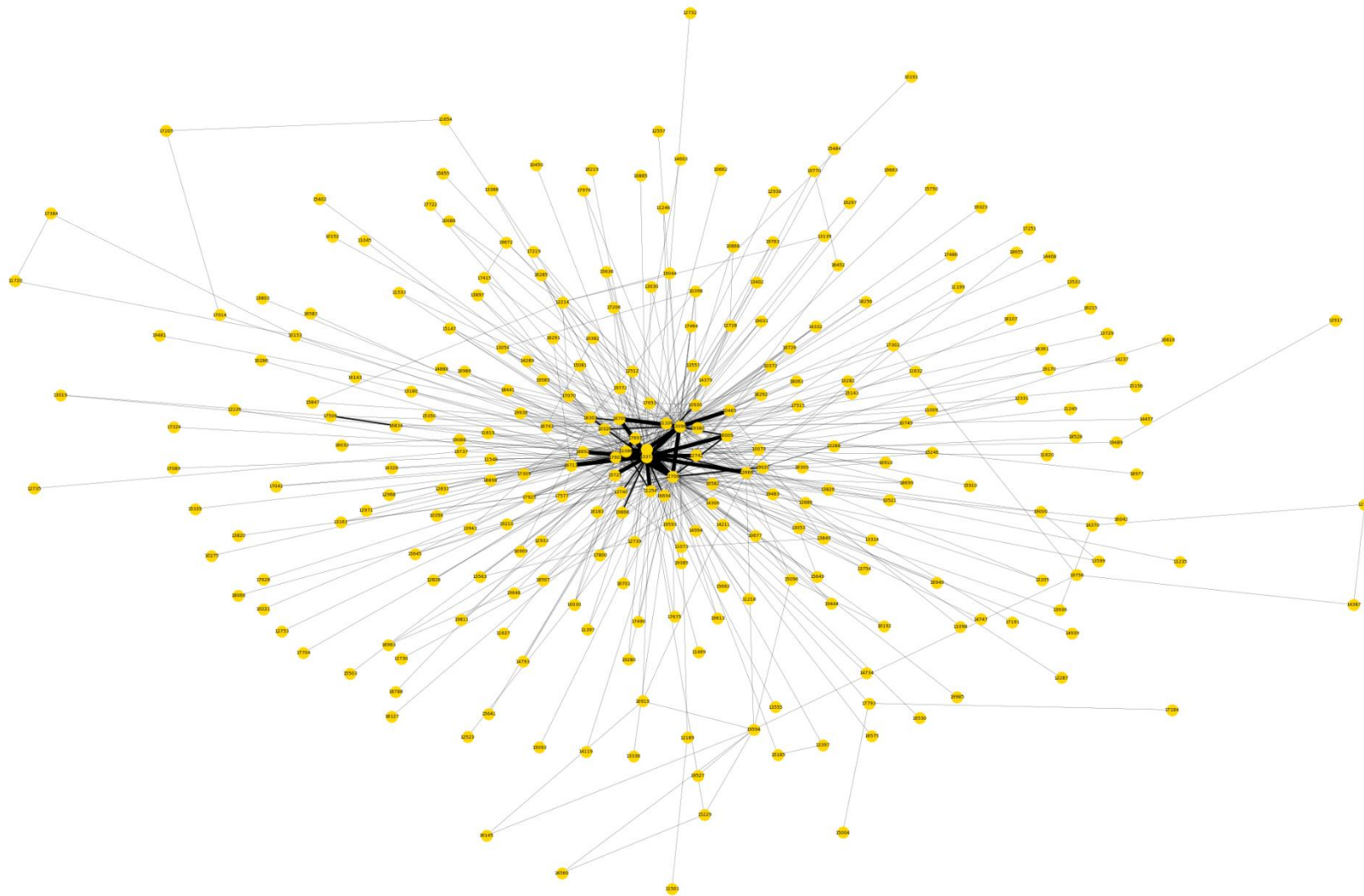




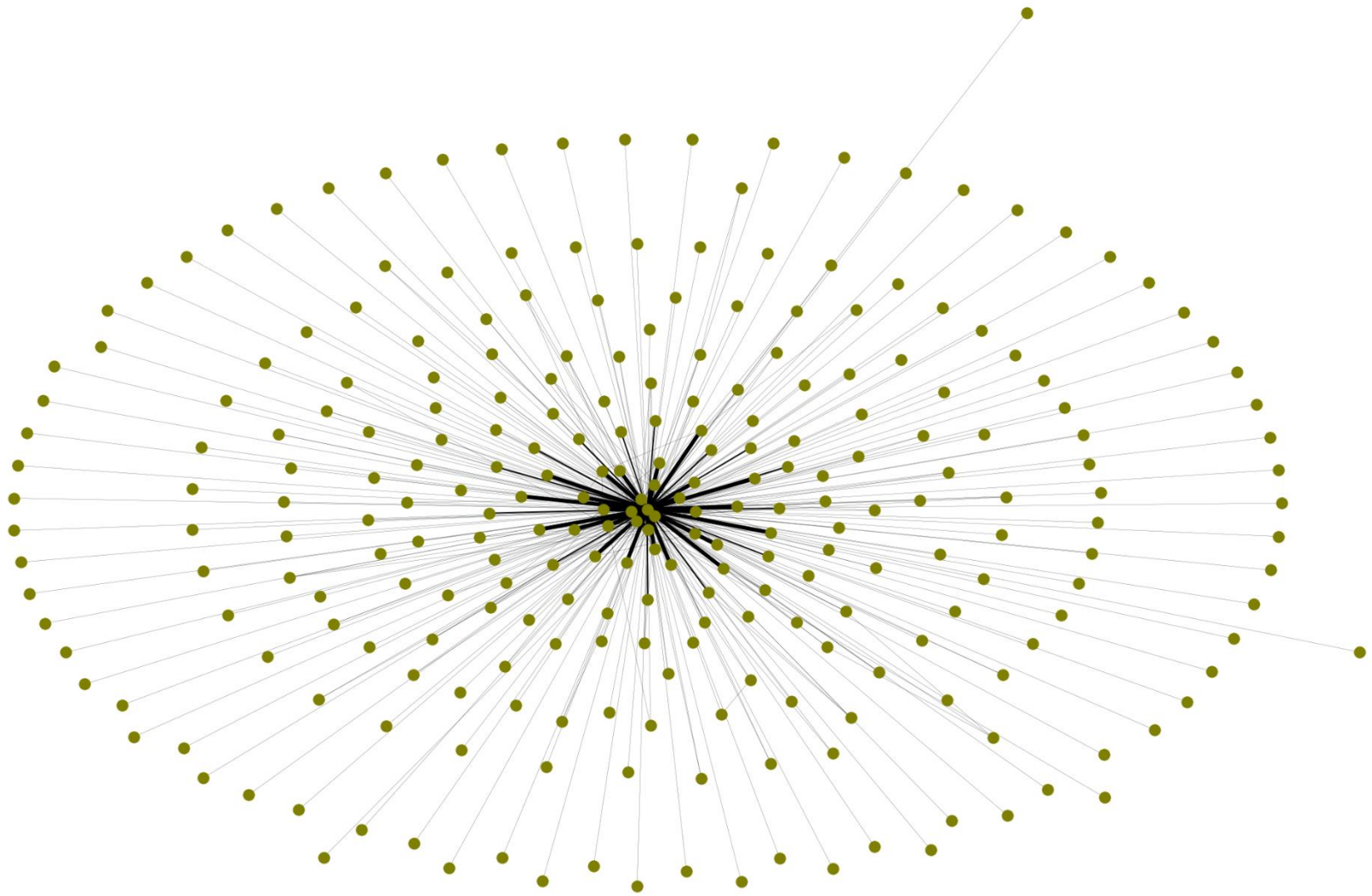
**Figure 65:** Empirical version of OpenSSL trust network, generated using spring layout in NetworkX and Matplotlib in the Python language



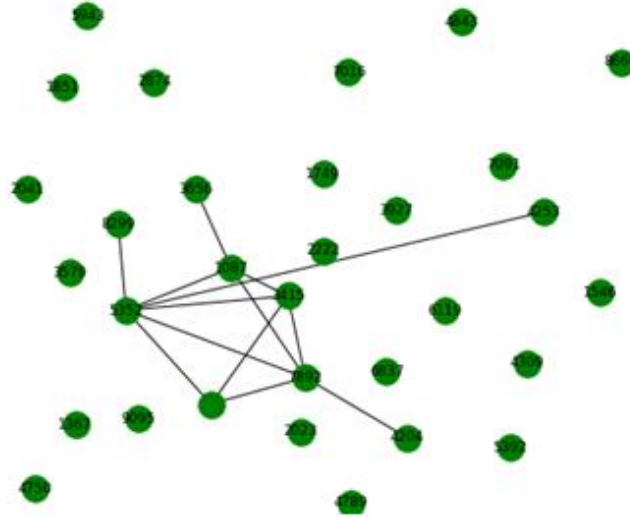
**Figure 66:** Random version of OpenSSL trust network, generated using spring layout in NetworkX and Matplotlib in the Python language



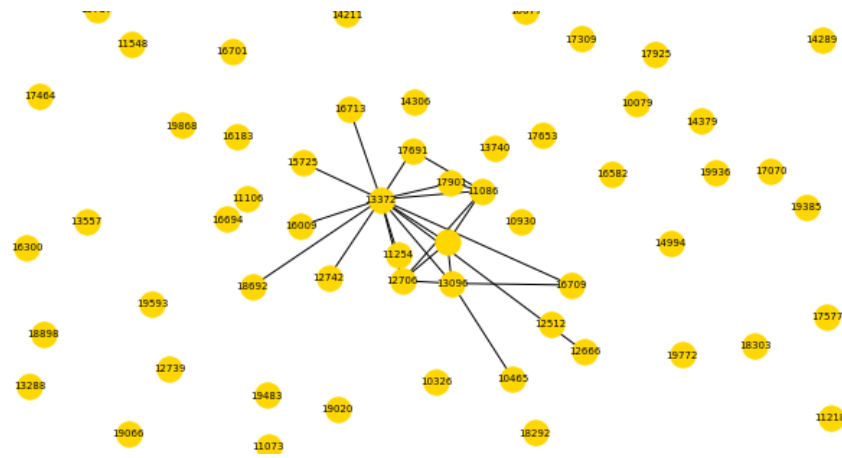
**Figure 67:** Empirical version of LE trust network, generated using spring layout in NetworkX and Matplotlib in the Python language



**Figure 68:** Random version of LE trust network, generated using spring layout in NetworkX and Matplotlib in the Python language



*Figure 69: Empirical OpenSSL trust network core*



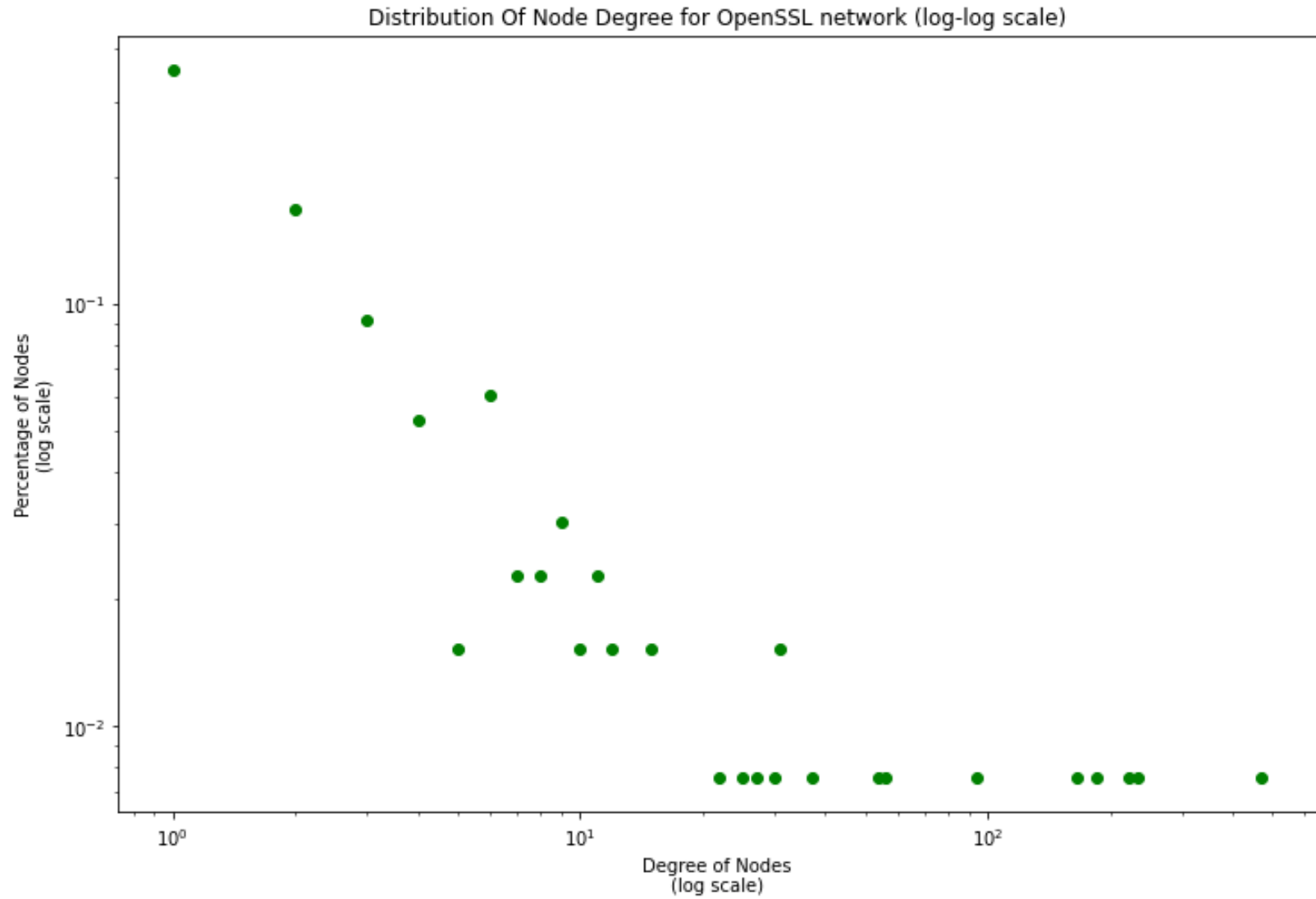
*Figure 70: Empirical LE trust network core*

### Node degree distribution

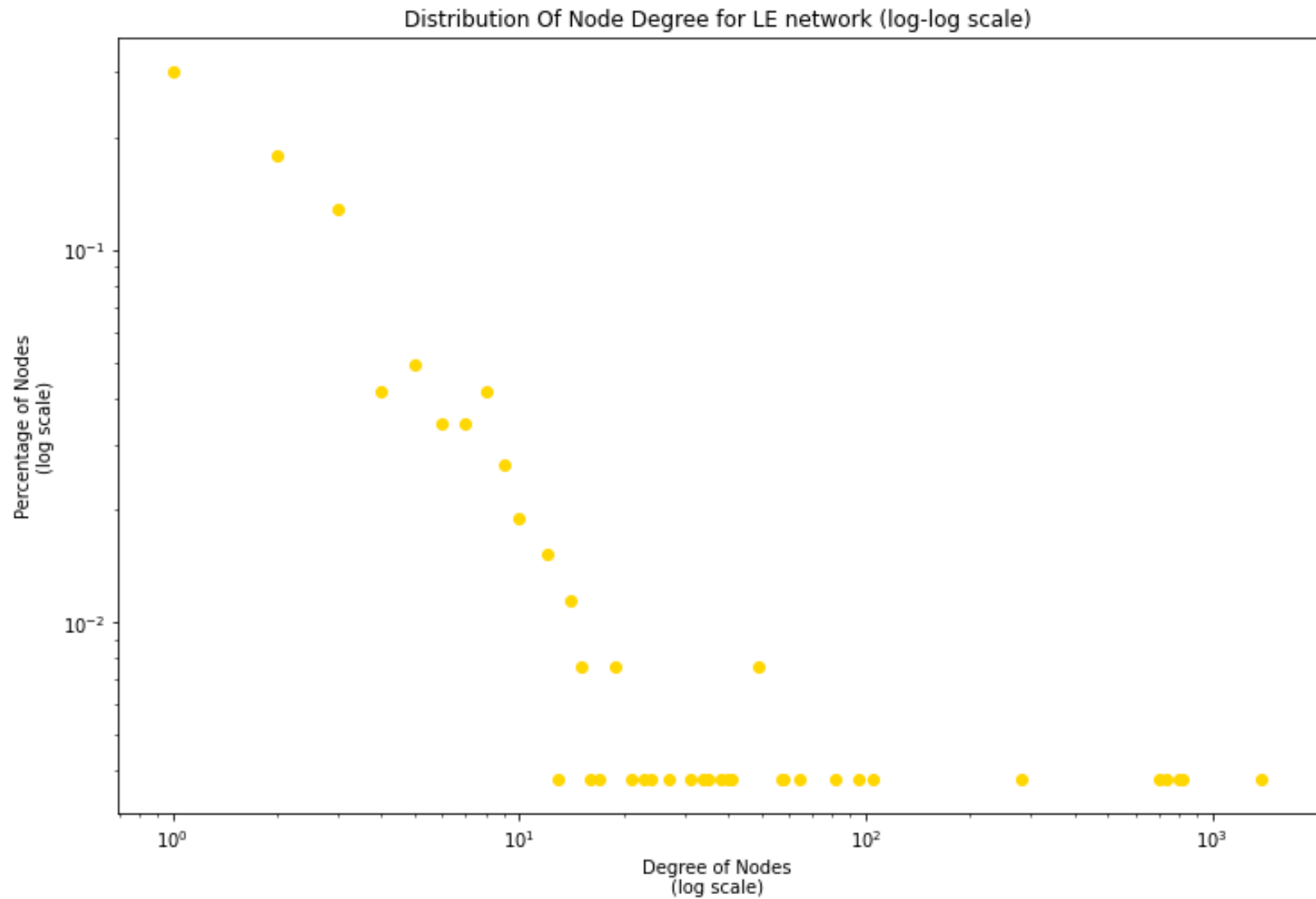
In Figure 71 and Figure 72, we present plots of node degree distributions on a log-log scale for the OpenSSL and the LE trust networks, respectively. In both trust networks, we observe a power law distribution, indicating the networks are examples of scale-free networks. Scale-free networks are a special case of small world networks, and thus we find evidence that both of these networks – as is typical for real world social networks – demonstrate small worldness.

To check the closeness of fit with the power law distribution of node degree, for both networks we perform a log-likelihood ratio (-2LL) test, comparing the fit of the lognormal distribution to the fit of the power law distribution [6.21]. In the OpenSSL network, we find an 82% probability that the distribution observed is a true power law distribution, and for the Let's Encrypt network, we find 92% probability that the distribution observed is a true power law distribution.

In Figure 73 and Figure 74, we present node degree rankings and degree histograms for both the OpenSSL and the LE trust networks. In the histograms, we observe “long tails” in the distribution of node degree – another classic sign of a power law distribution. Having verified the small worldness of both networks, we can proceed with our use of the mean path length in each trust network as the maximum trust propagation distance for our own transitive trust calculations, discussed in a previous section.

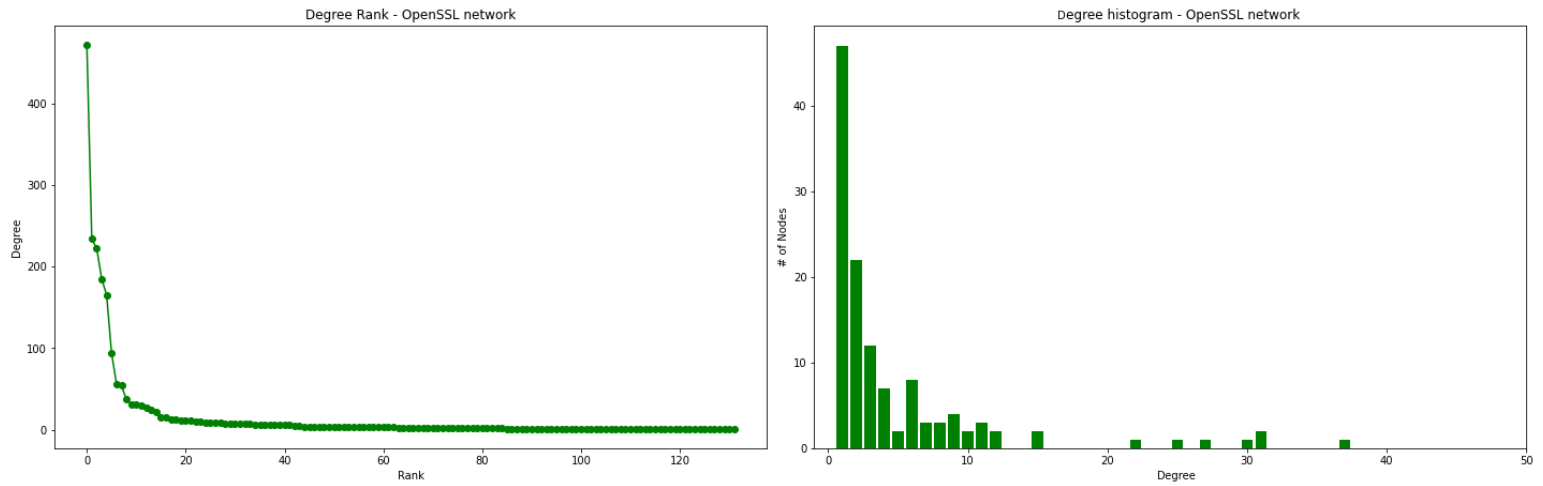


**Figure 71:** Degree distribution in OpenSSL trust network. Log-log scale.

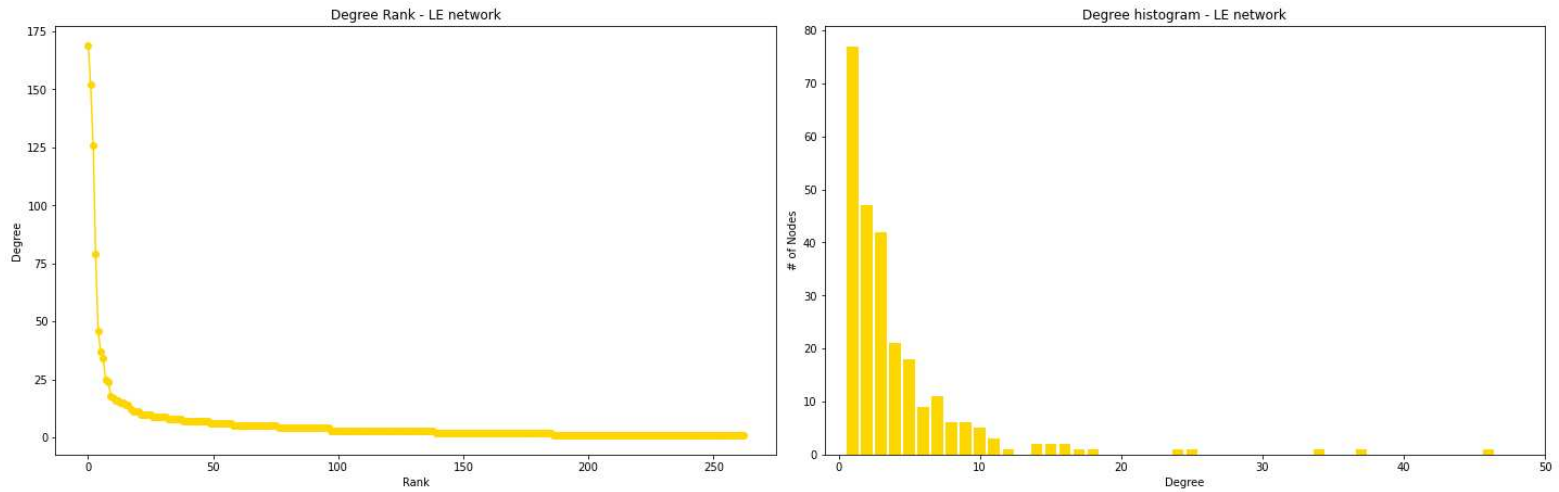


**Figure 72:** Degree distribution in LE trust network. Log-log scale.





**Figure 73:** Degree rank plot (left) and degree histogram (right) of OpenSSL trust network.



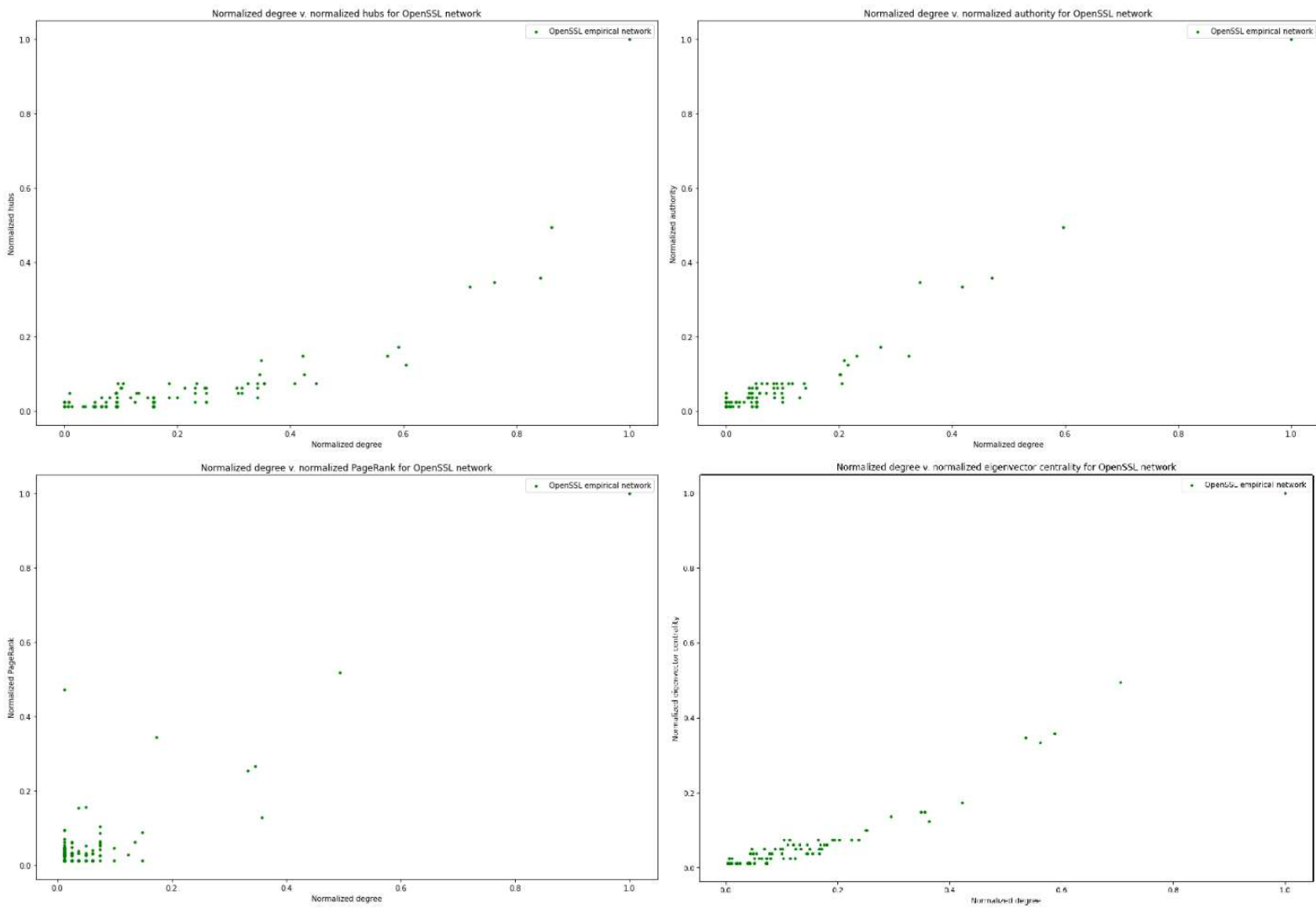
**Figure 74:** Degree rank plot (left) and degree histogram (right) of OpenSSL trust network.

### Direct trust and transitive trust analysis results

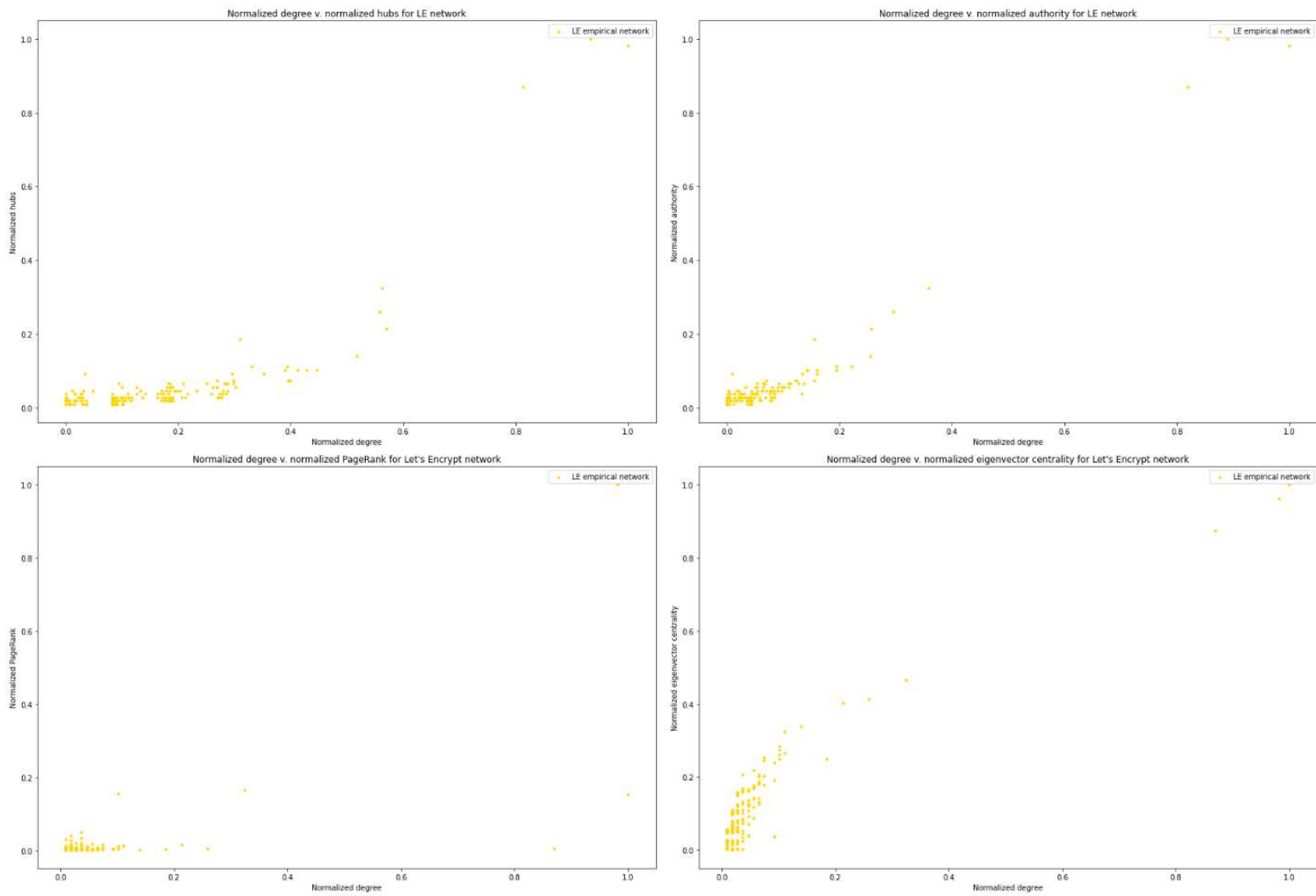
In this section we present and interpret the results of our analyses using direct and indirect (transitive trust) measures in the OpenSSL and LE networks. First, we present comparisons of several different direct and indirect trust evaluations methods, followed by a more detailed discussion about each method's results. Following the methods described in [6.20] for direct trust estimation in online networks, in Figure 75 and Figure 76, we present four different perspectives on direct trust estimation for the OpenSSL and LE networks, respectively. In both figures, reading clockwise from the upper left we present normalized degree against normalized hubs, against normalized authority [6.22], against normalized eigenvector centrality, and against normalized PageRank. Several important observations can be drawn from these plots.

In the case of normalized degree against normalized HITS (hubs and authority), we observe an approximately linear relationship for both networks. One interesting difference between the OpenSSL and the LE networks is seen in the normalized degree v. normalized eigenvector centrality quadrants of the figures. In the case of the OpenSSL network, we observe an approximately linear relationship between the two, similar to the other measures illustrated in Figure 75. However, for the LE network we observe a more exponential relationship between normalized degree and normalized eigenvector centrality, with high relative eigenvector centralities being observed even for nodes with low relative degree. This may indicate a more egalitarian distribution of total network trust than in the OpenSSL network, which in light of this result appears more hierarchical.

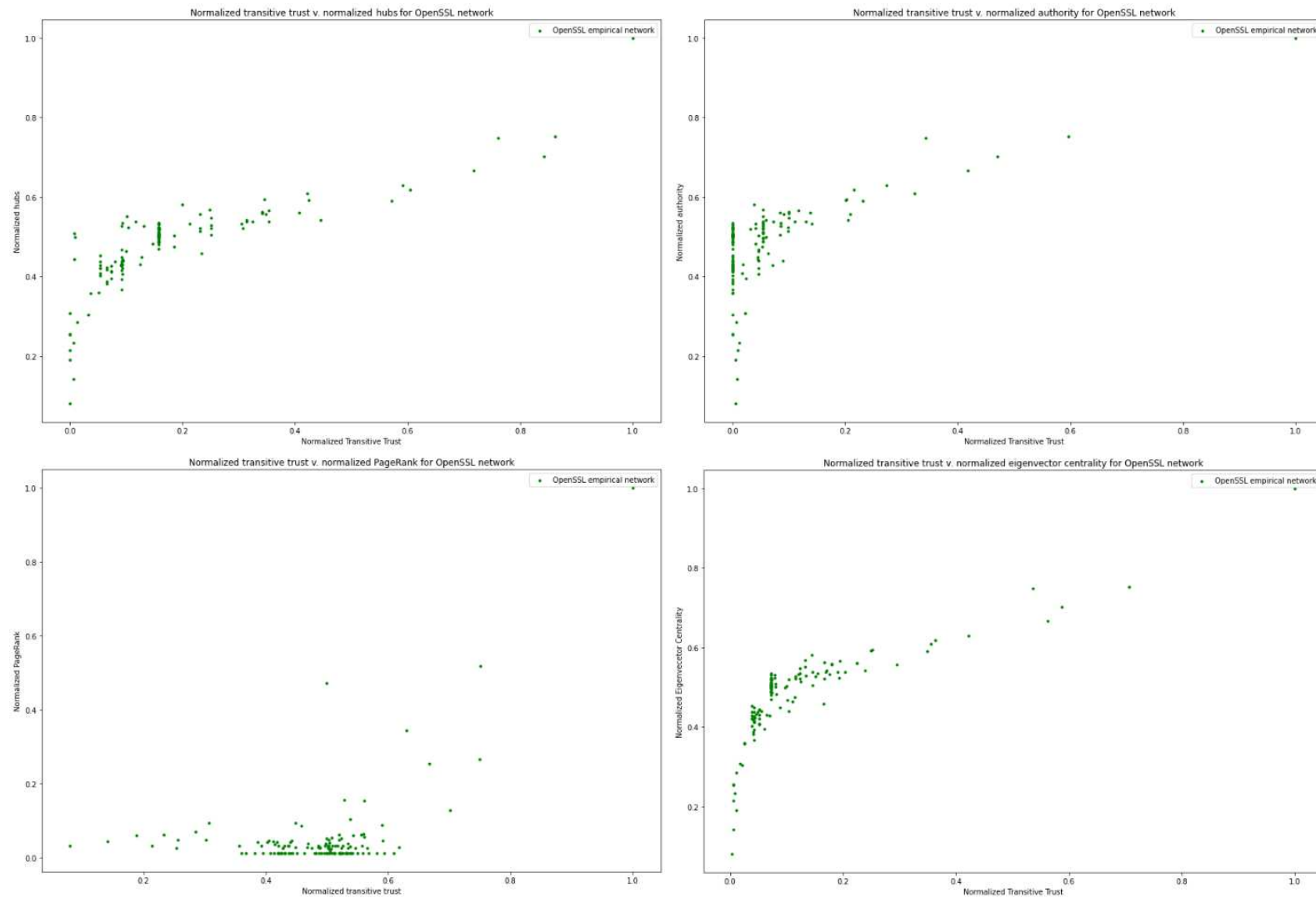
Referring to the quadrants representing PageRank and eigenvector centralities v. generic transitive trust in Figure 77 and Figure 78, we find similar results; there appears to be an important difference in the distribution of trust globally within the OpenSSL network compared to the LE network. Finally, when comparing the trust distributions when measured by a direct trust measure as in Figure 75 and Figure 76, we see clear differences in the trust levels when measured by an indirect trust measure as in Figure 77 and Figure 78.



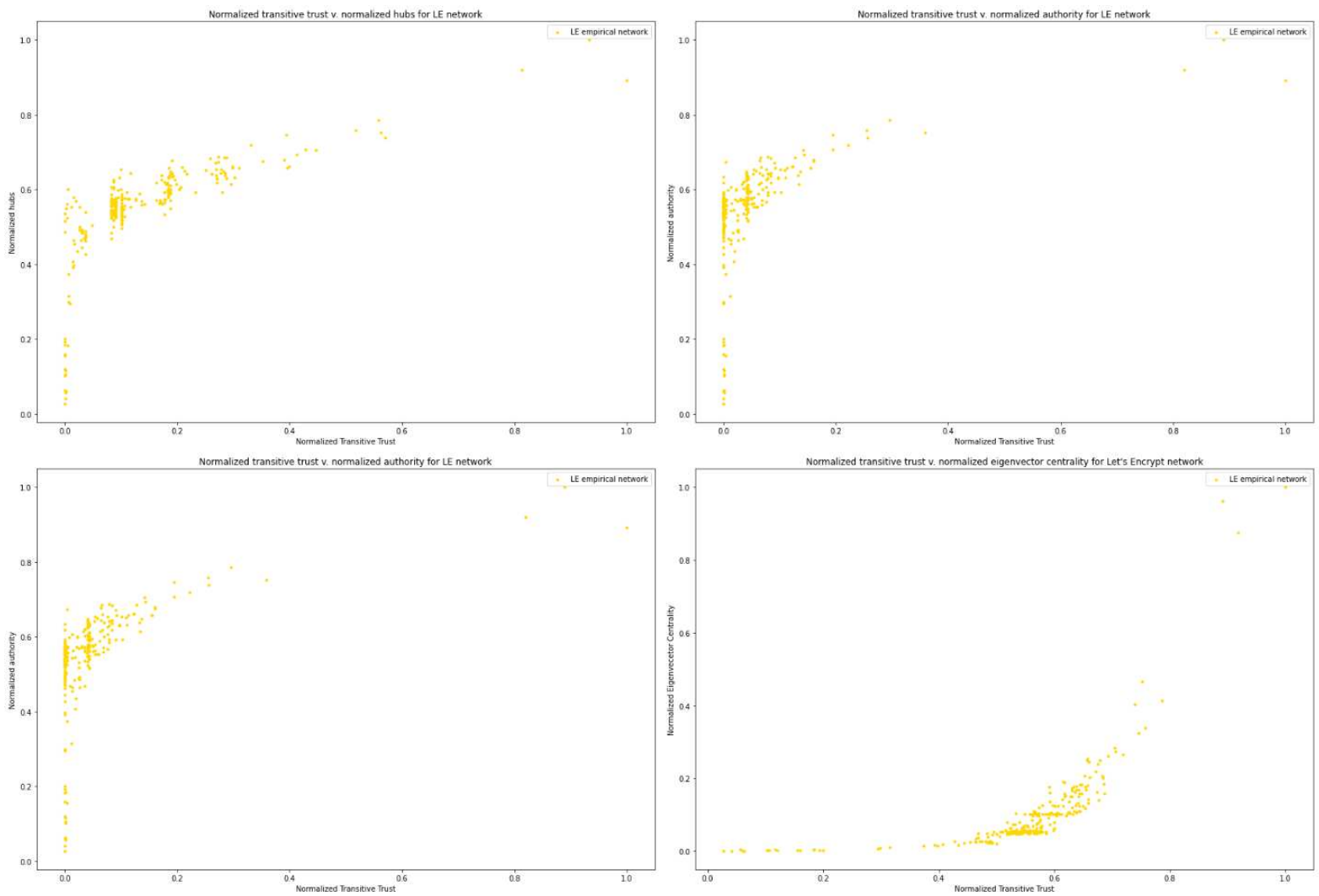
**Figure 75:** Direct trust measures for OpenSSL empirical trust network. Normalized degree v. (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank.



**Figure 76:** Direct trust measures for LE empirical trust network. Normalized degree v. (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank.



**Figure 77:** Transitive trust measures for OpenSSL empirical trust network. Normalized transitive trust v. (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank.



**Figure 78:** Transitive trust measures for LE empirical trust network. Normalized transitive trust v. (from top left, clockwise) normalized hubs, normalized authority, normalized eigenvector centrality, and normalized PageRank.

## Trust measures – direct trust

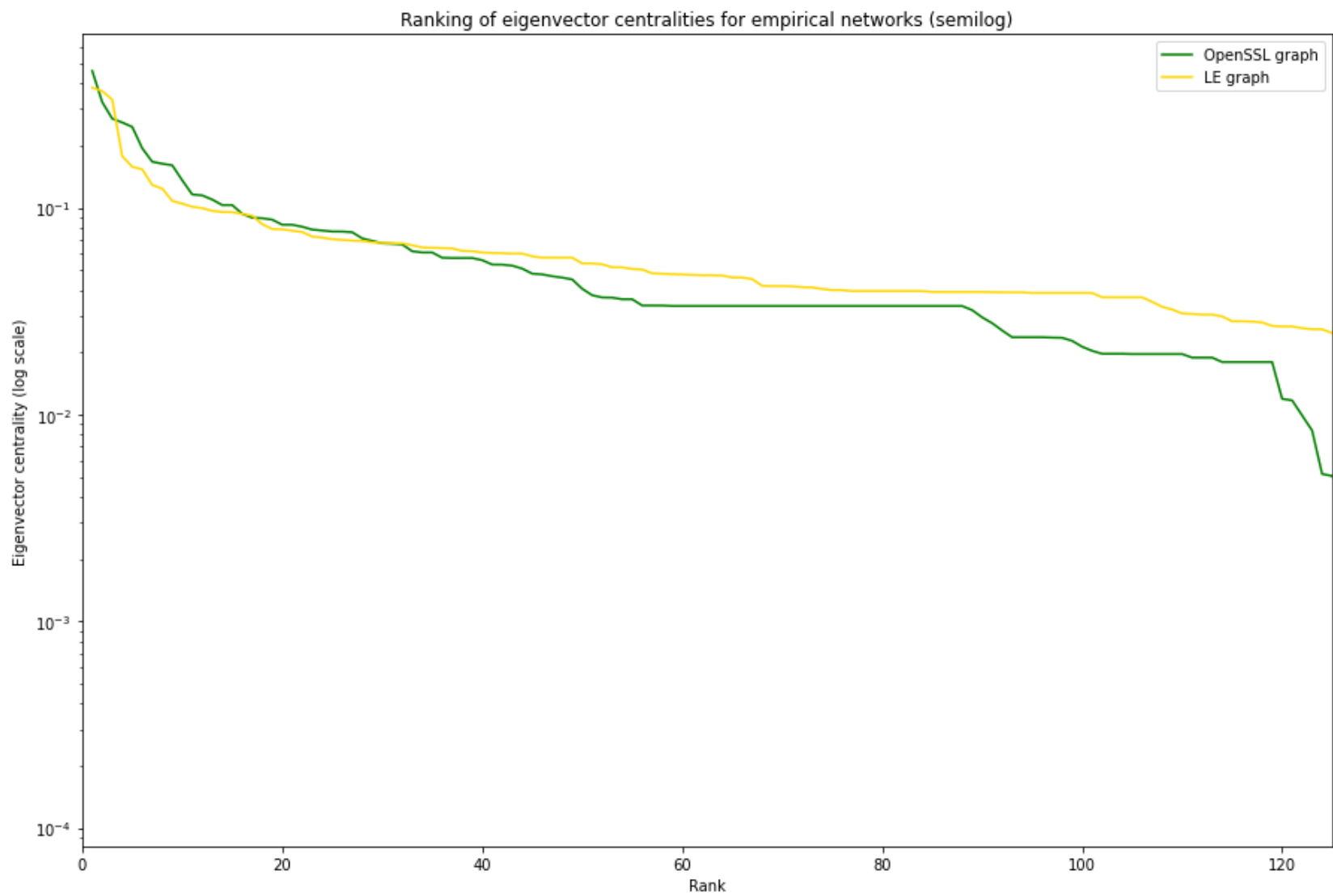
Adapting a method from [6.20] for use in our networks, we apply standard centrality measures as proxies for direct trust measurements. These measures are useful because they give a sense for how much each node is trusted by its direct neighbors; in the following section we discuss how we measure and infer trust in nodes with respect to their indirect neighbors.

In Figure 79 we present rankings by eigenvector centrality for the OpenSSL and the LE empirical networks. For approximately the first quartile of the distribution we observe higher levels of direct trust as measured by eigenvector centrality in the OpenSSL network compared to the LE network. For the latter three quartiles of the distribution, we see the direct trust levels in the LE network outpace those of the OpenSSL network. We interpret this to mean that trusted nodes are (relatively speaking) more trusted in the OpenSSL network than in the LE network, but “everyday” nodes in the LE network enjoy relatively more trust than their similarly-situated counterparts in the OpenSSL network. Figure 80 gives a similar perspective, but instead of eigenvector centrality it considers direct trust as measured by PageRank. This figure shows higher direct trust levels in the OpenSSL network than in the LE network for most of the distribution.

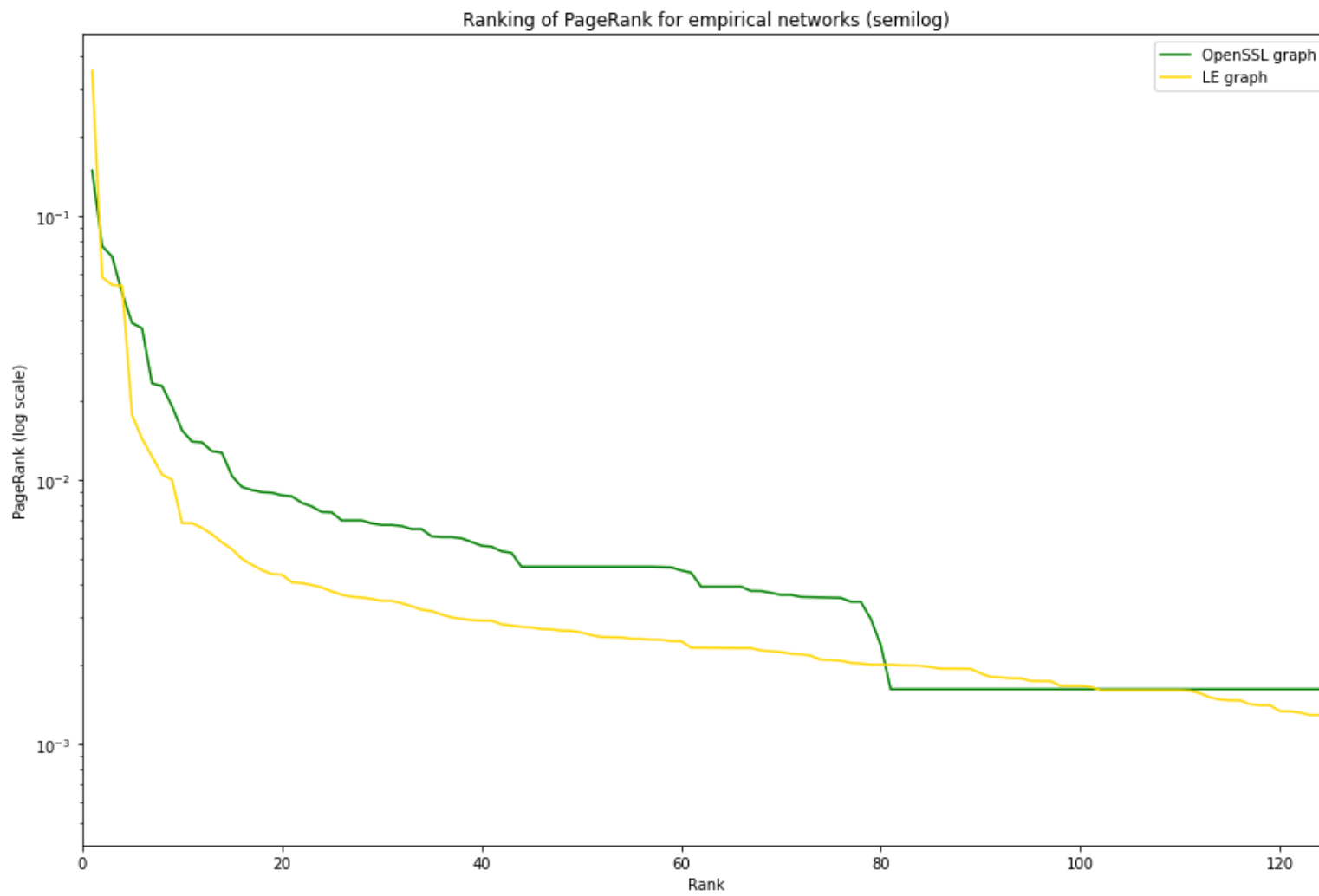
In Figure 81 and Figure 82 we present direct trust as measured by eigenvector centrality for the empirical and random versions of the OpenSSL and LE trust networks, respectively. Figure 83 and Figure 84 provide another perspective, this time plotting direct trust as measured by PageRank for the empirical and random equivalent graphs of the



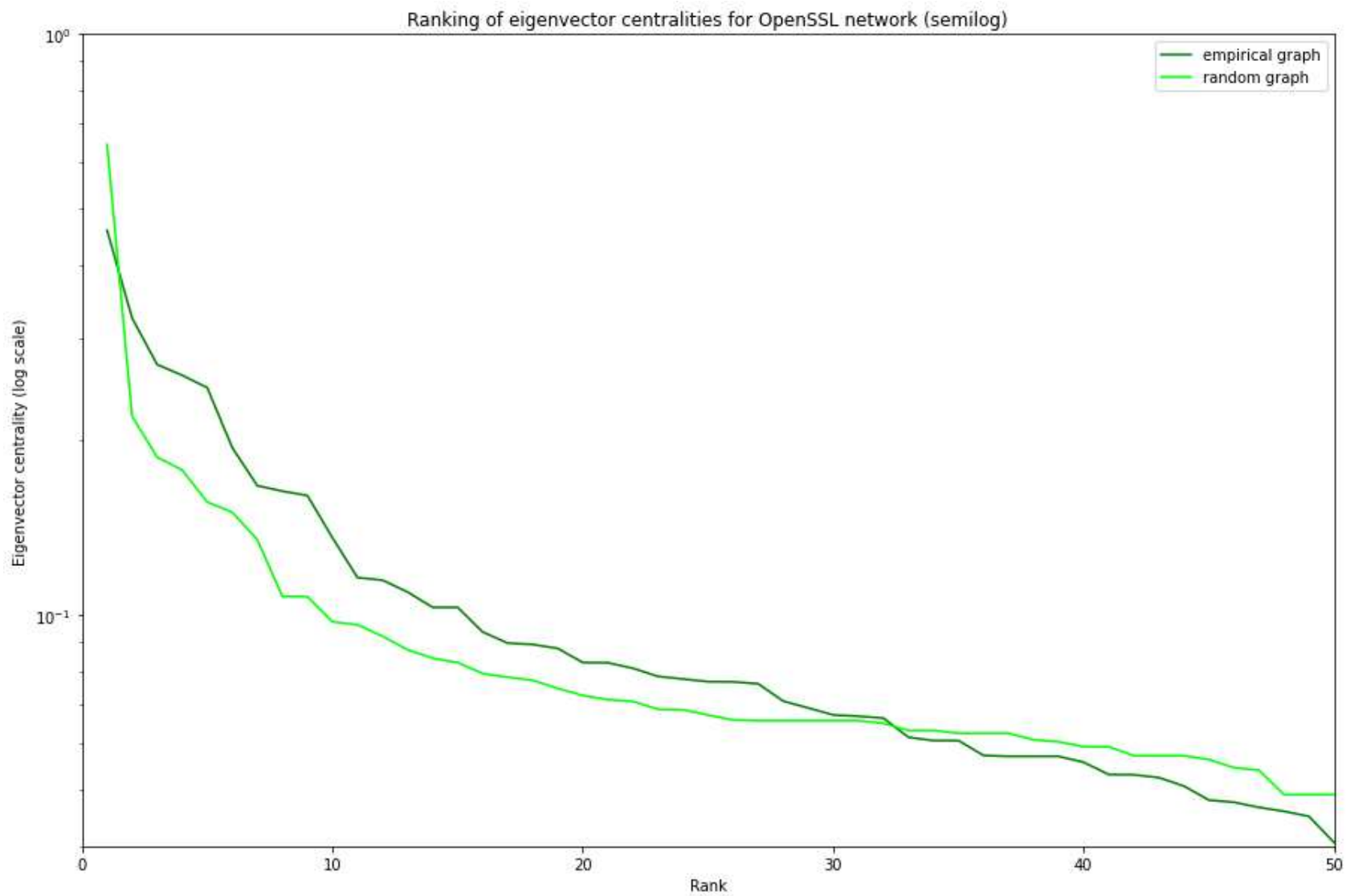
OpenSSL and the Let's Encrypt trust networks, respectively. In both of these figures we observe that for the majority of the distribution there is a significant difference between trust levels in the empirical and the random equivalent graphs. Across each of these figures, we observe a significant difference in direct trust levels at nearly all levels of the distribution for the empirical against the random equivalent networks. We interpret this as additional evidence that these trust networks were not formed by random processes, and that trust may (in part) play a role in explaining the behavior of both empirical networks.



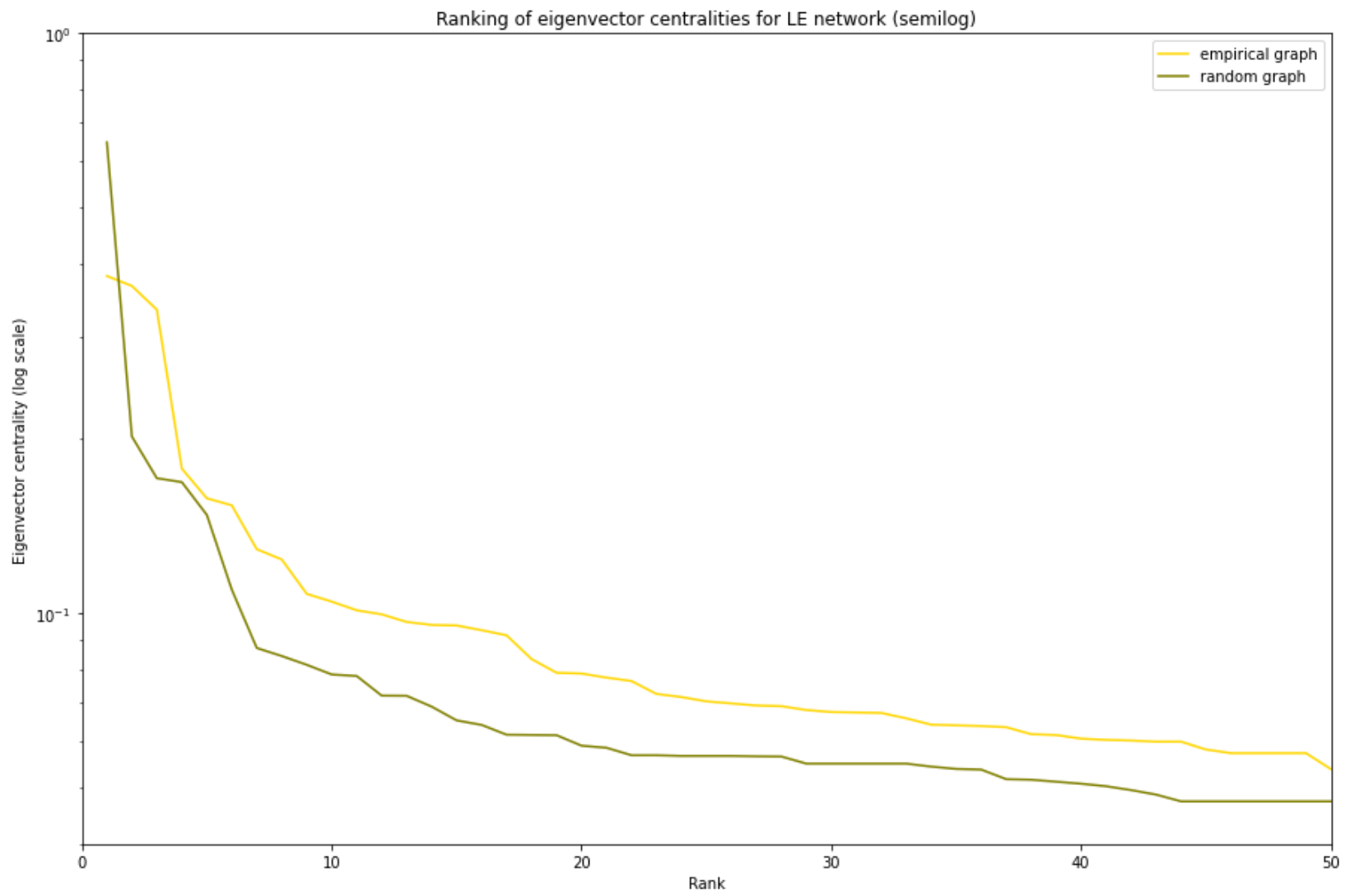
**Figure 79:** Comparing eigenvector centrality rankings for OpenSSL and LE empirical networks



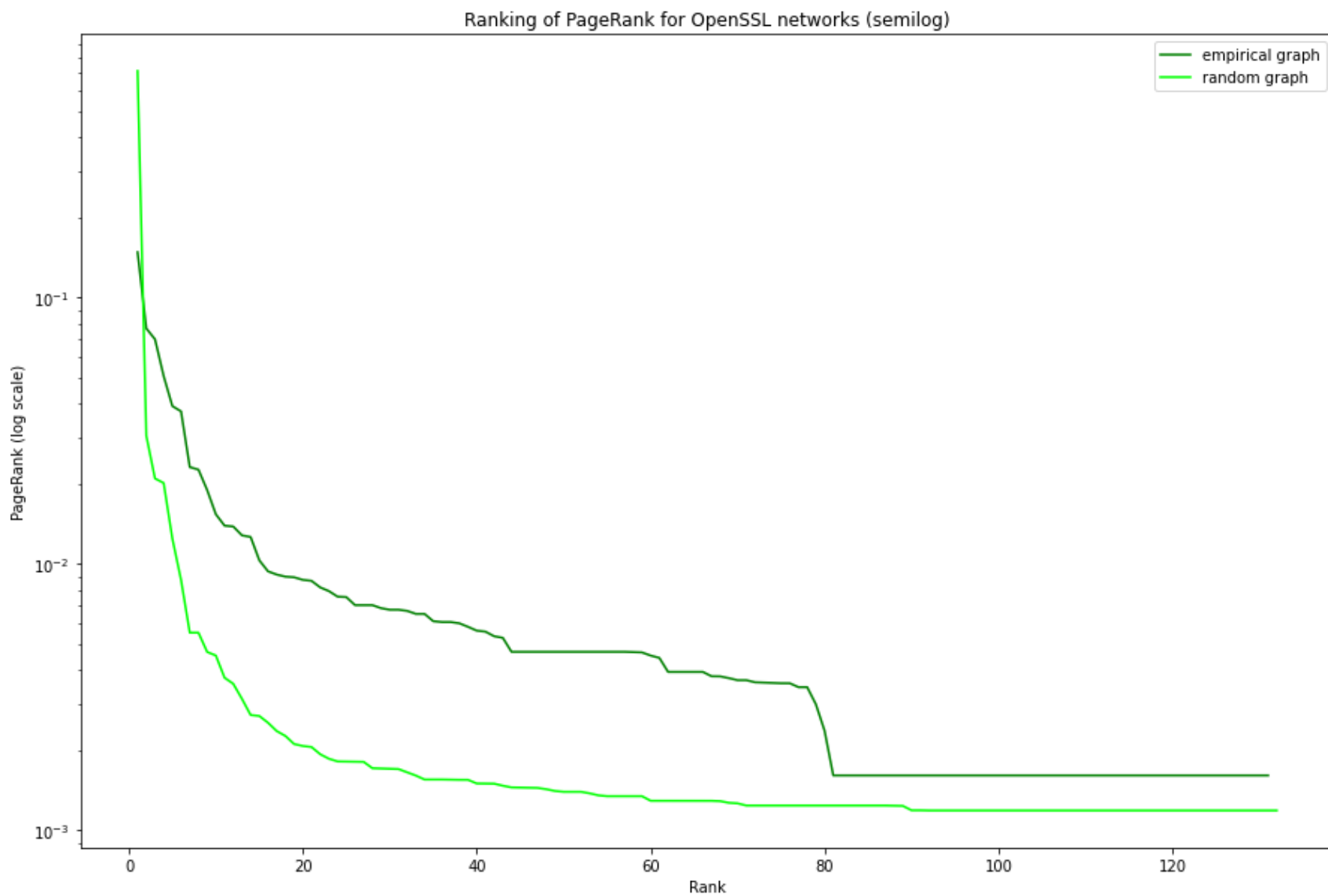
**Figure 80:** Comparing PageRank rankings for OpenSSL and Let's Encrypt empirical trust networks



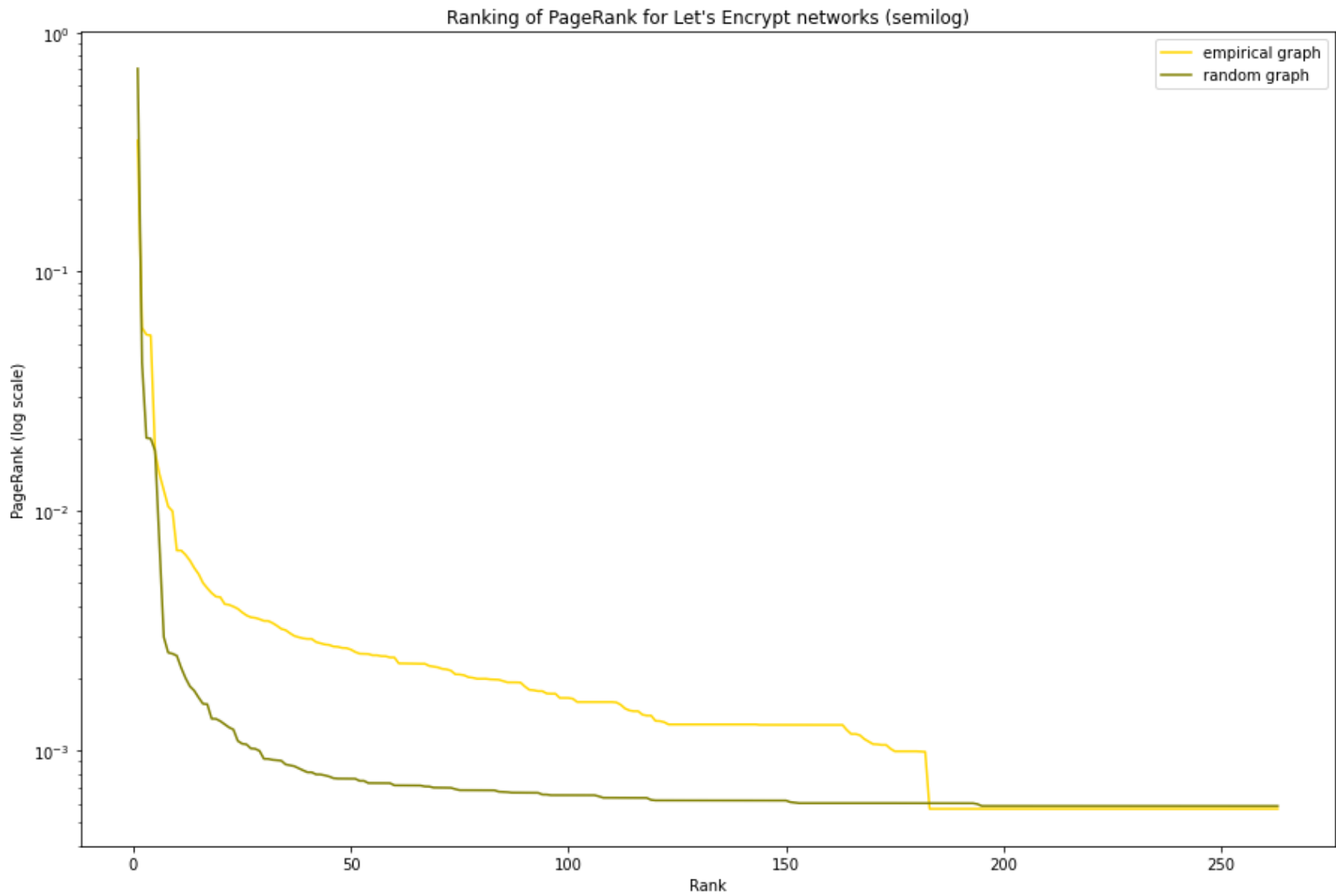
**Figure 81:** Ranking of eigenvector centralities for OpenSSL trust networks (empirical and random)



**Figure 82:** Ranking of eigenvector centralities for Let's Encrypt trust networks (empirical and random)



**Figure 83:** Ranking of PageRank for OpenSSL empirical and random networks. Semilog scale.



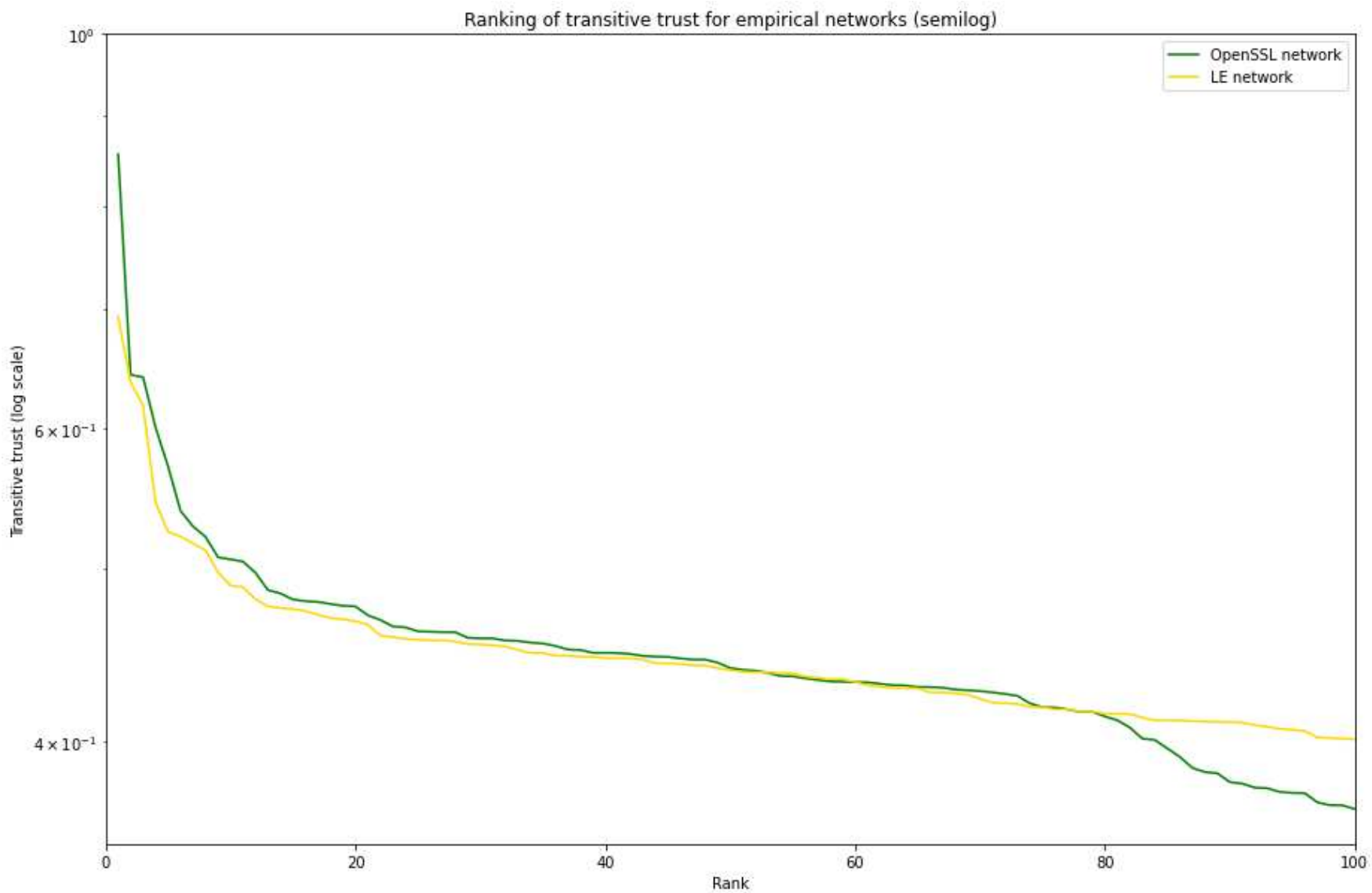
**Figure 84:** Ranking of PageRank for Let's Encrypt empirical and random networks. Semilog scale.

## Trust measures – transitive trust

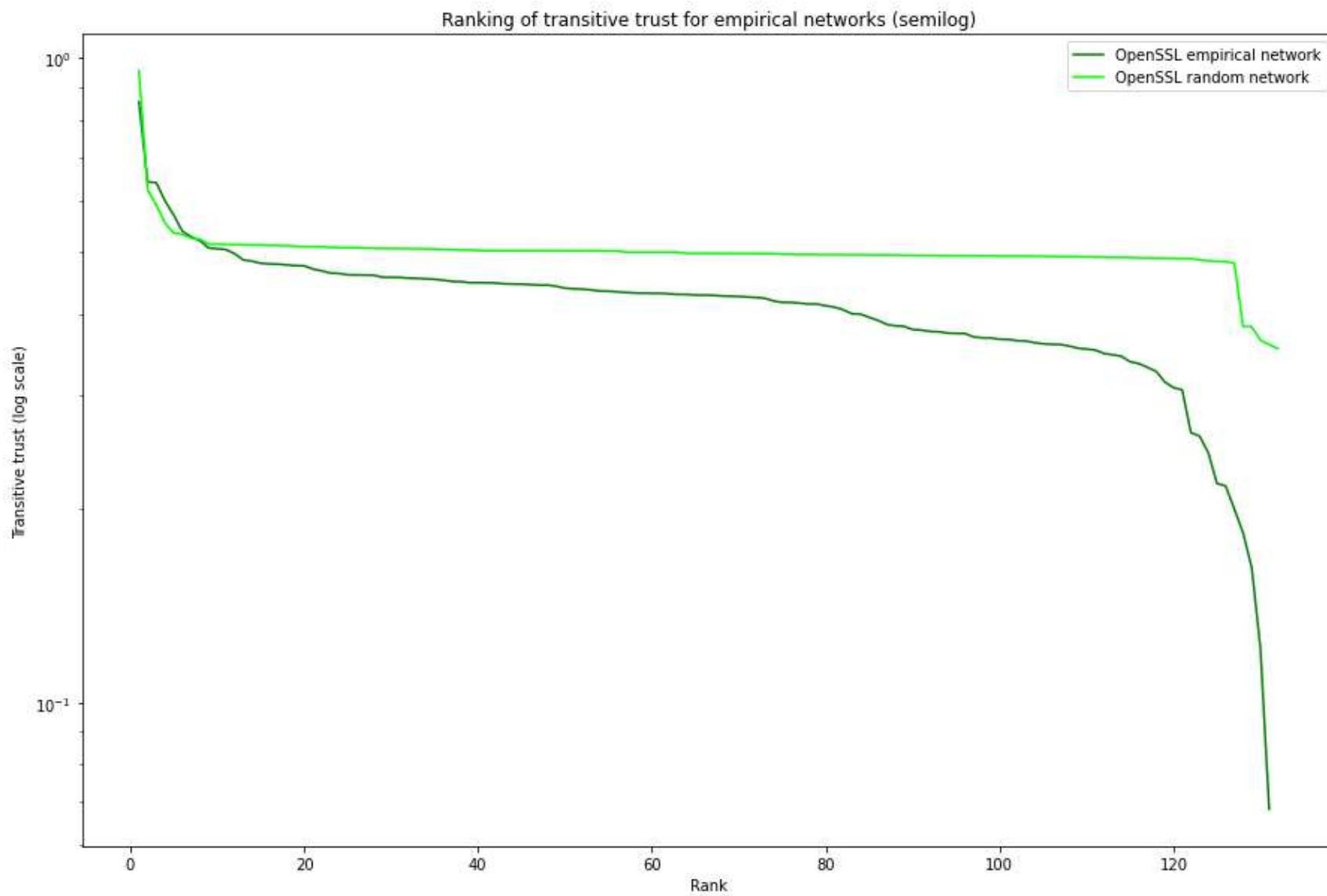
In this section, we present and discuss the results of our transitive trust analyses. The analyses discussed in this section utilized our modified transitive trust measure as described in an earlier section of this chapter. We find that there are generally higher transitive trust (total trust) levels present in the OpenSSL network than in the LE network. On the one hand this is surprising, as the LE network being larger presents more opportunities for nodes to establish more connections and have more interactions with their peers; on the other hand, it is reasonable because in a smaller network (like OpenSSL) developers have more opportunity to engage in deeper and more frequent interactions with a smaller number of peers, leading to deeper trust with those peers. In Figure 85 we present rankings of transitive trust for each empirical network using our transitive trust measure. The figure shows slightly higher total trust levels in the OpenSSL network compared to the LE network for the top approximately 50 nodes (developers), similar trust levels between both networks from approximately node 50 to node 80, and in the tail of the distribution a higher trust level in the LE network for nodes 80+.

In Figure 86 and Figure 87 we present transitive trust rankings for the empirical and random versions of the OpenSSL and the LE trust networks, respectively. In both figures, we observe a clear difference in trust levels for the two versions of the network throughout the distribution, with the random version displaying higher trust levels than the empirical network for most of the distribution. This indicates that trust is generated in the LE network not by random processes but by a specific mechanism or mechanisms.

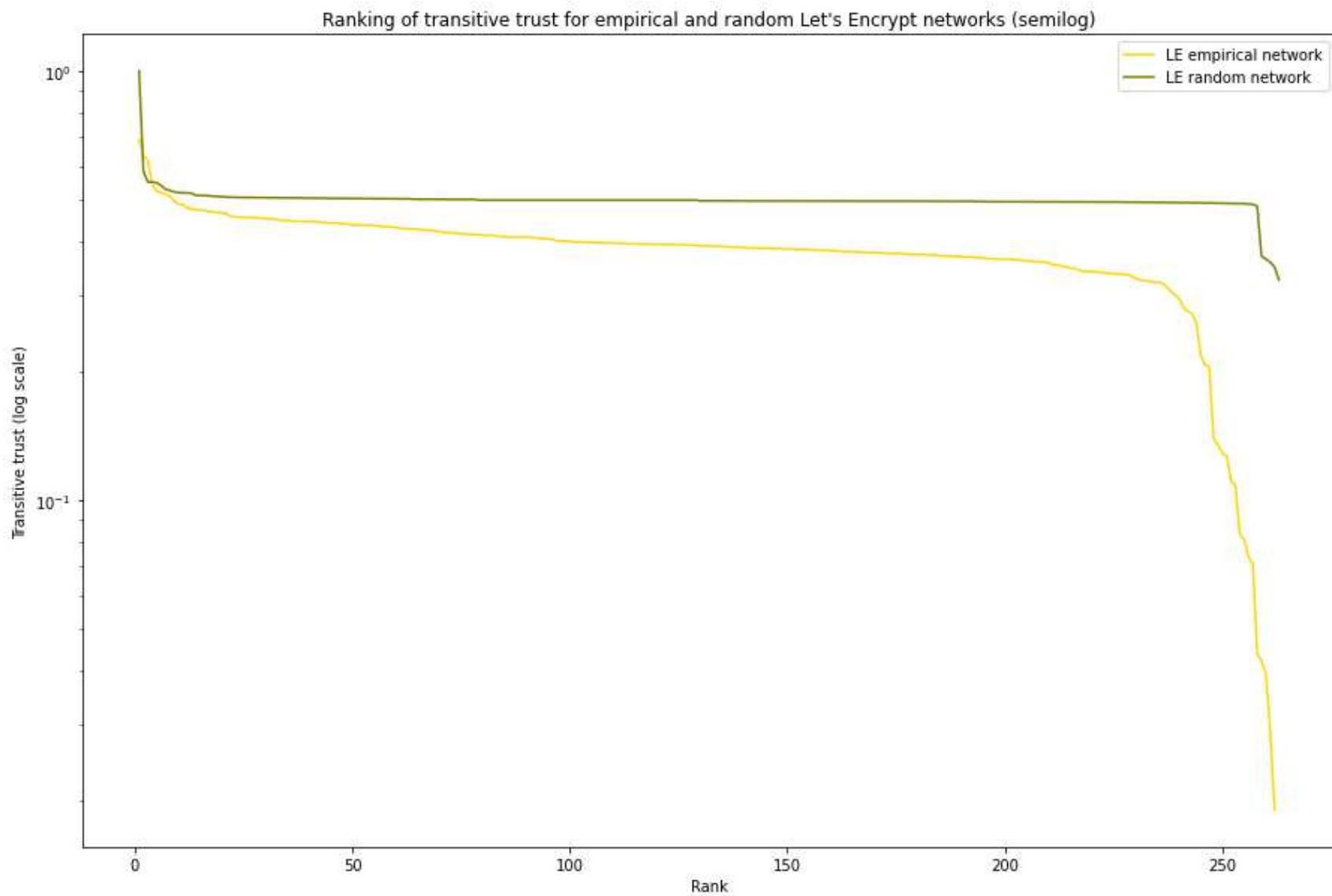




**Figure 85:** Transitive trust rankings for empirical OpenSSL and LE networks. Semilog scale.

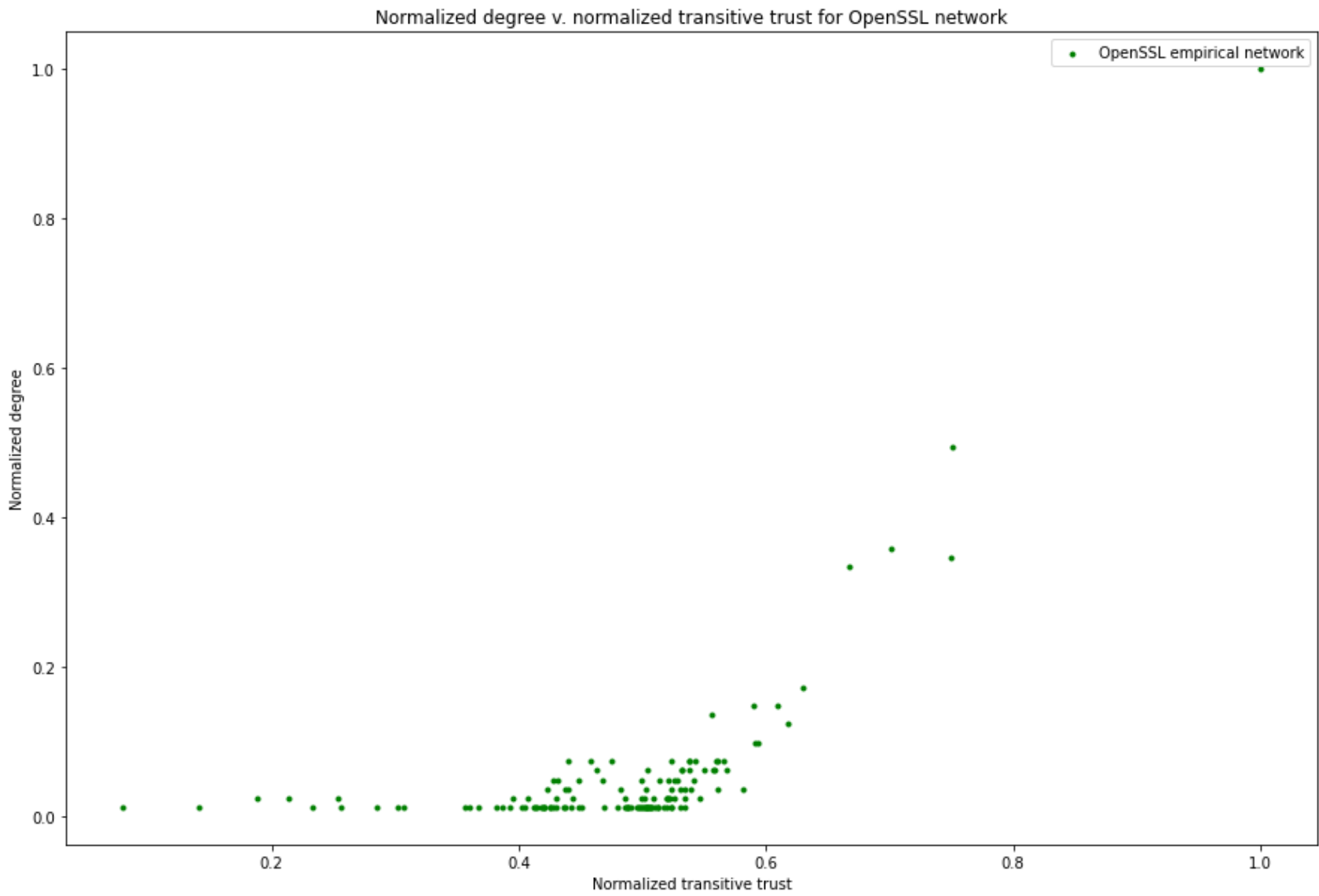


**Figure 86:** Transitive trust rankings for empirical and random OpenSSL network. Semilog scale

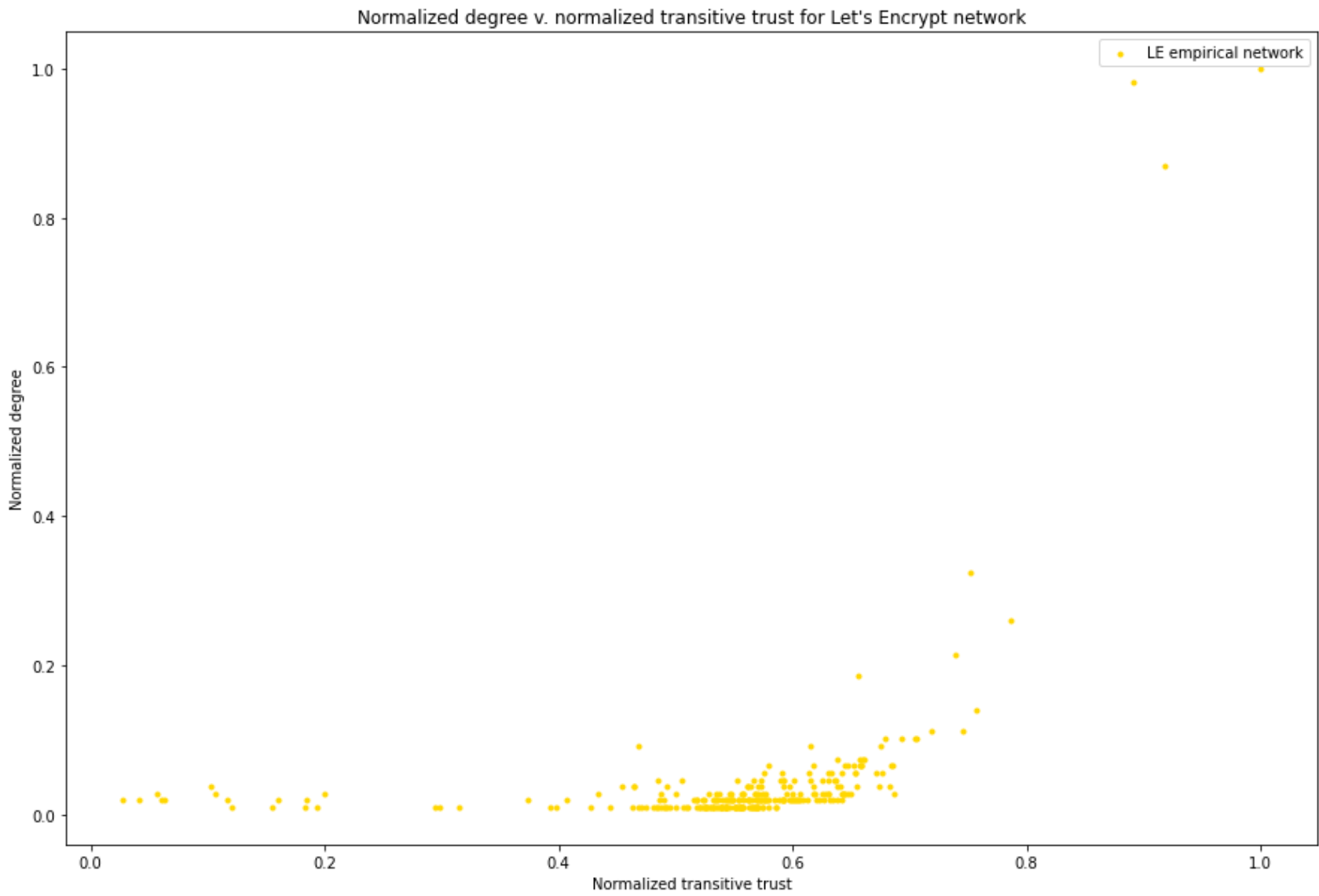


**Figure 87:** Transitive trust rankings for empirical and random LE network. Semilog scale.

In Figure 88 and Figure 89 we present plots of normalized degree against normalized transitive trust scores for each node in the empirical OpenSSL and LE trust networks, respectively. In both networks, we see roughly similar distributions, with an exponential relationship between the two measures (degree and transitive trust). We find that up to approximately 60% of the maximum transitive trust level in a given network, there is very little change in a node's (normalized) degree. After reaching this threshold of approximately 60% of maximum transitive trust level, there is an inflection wherein normalized degree of a node begins to move more closely in relation to changes in normalized transitive trust. We also check the sensitivity of this result by varying the MTDP used in the generic transitive trust measure, and find similar results. This makes sense, because nodes with higher degree are, all other things equal, more likely to also have more paths going through and to them, which will increase their trust values as measured by our generic transitive trust measure. While the patterns are similar in both networks, we find this pattern is even more slightly pronounced in the LE trust network than in the OpenSSL trust network. Our interpretation of this is that 1) there is little difference in how trust forms with respect to the network structure (degree) across the two networks, indicating that this *may* be a feature of growth of trust networks in online settings more generally, and 2) even as a relatively "unimportant" node (one with few connections, i.e., low degree) it is relatively easy to achieve a fairly high position in terms of trust in the network. For example, a node that only has 1% of the degree that the highest degree node has can still potentially achieve a trust level of more than 50% of the trust level of the highest degree node.



**Figure 88:** Normalized degree v. normalized transitive trust for OpenSSL empirical network



**Figure 89:** Normalized degree v. normalized transitive trust for LE empirical network

### Trust correlations table

In Table 25 we present correlations among various trust measures, both direct and indirect, which we have applied in the preceding sections. Several interesting points stand out from the table. First, when comparing the correlations among various trust measures in these networks against the correlations in Table 18 in Chapter V, we see a much closer agreement (higher correlation) between PageRank and eigenvector centrality in the OpenSSL and LE networks than for the networks analyzed in Chapter V. This is to be expected, as the networks in Chapter V are much larger and more complex in terms of their structures and paths. We also observe a stronger correlation between our transitive trust measure and both eigenvector centrality and PageRank (our proxies for direct trust estimations in this chapter) in the OpenSSL trust network compared to the LE trust network. Once again, this is not surprising when considering what each of these measures take as inputs, because in this chapter the LE trust network is approximately twice the size of the OpenSSL trust network when measured by number of nodes, and more than three times the size when measured by number of edges. Based on the design of our transitive trust measure utilized in this chapter, we expect that as the size of the network under analysis grows the agreement between our transitive trust measure and direct trust measures such as PageRank and eigenvector centrality will drop. We take this as a positive gauge of its usefulness (in addition to the other evidence presented previously) for estimating transitive trust in networks that share similar characteristics to those analyzed in this chapter. Additionally, we find additional evidence in support of findings similar to

those from other researchers discussed at length in Chapter III which demonstrate that naïve network measures such as simple degree are typically not good approximations for trust in online networks. One implication of this is that, when designing recommender systems or reputation systems for online settings, a simple recommendation based on number of followers or number of connections will be unlikely to produce satisfactory results for most users.

**Table 25:** Correlations of trust measures in OpenSSL and LE empirical networks

OpenSSL trust network

	eigenvector centrality	PageRank	hubs	authority	degree	transitive trust
eigenvector centrality						
PageRank	0.743					
hubs	0.947	0.627				
authority	0.965	0.779	0.869			
degree	0.928	0.852	0.810	0.956		
transitive trust	0.785	0.492	0.804	0.691	0.623	

Let's Encrypt trust network

	eigenvector centrality	PageRank	hubs	authority	degree	transitive trust
eigenvector centrality						
PageRank	0.600					
hubs	0.946	0.547				
authority	0.976	0.668	0.897			
degree	0.955	0.656	0.844	0.974		
transitive trust	0.648	0.216	0.695	0.515	0.405	

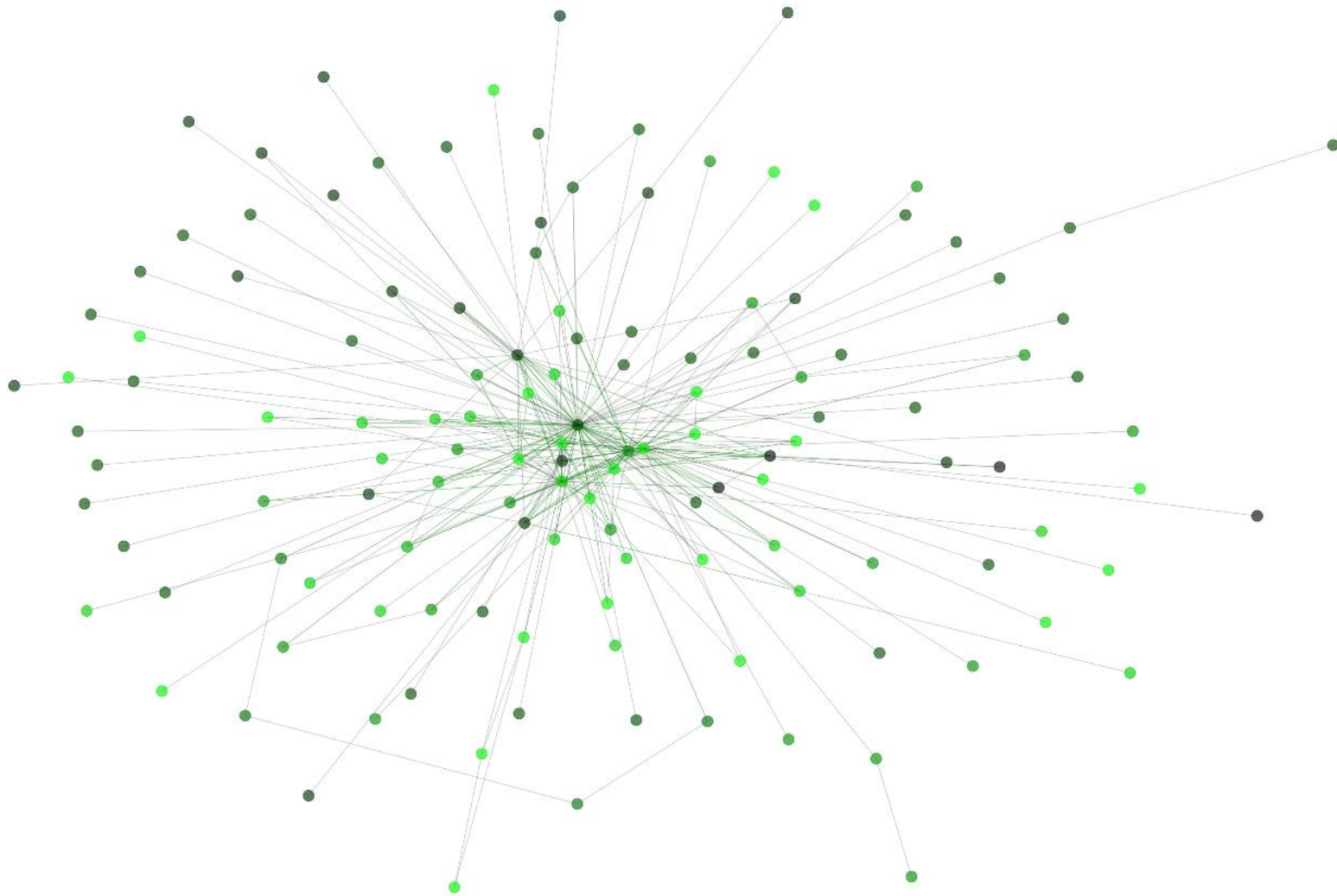
Community detection

Using the greedy modularity maximization algorithm described in the previous section, we apply community detection techniques to our networks. We find eight distinct communities in both the OpenSSL trust network and in the LE trust network. In Figure 90 and Figure 91 we present visualizations of the OpenSSL and LE empirical trust networks. The visualizations were generated in the Python language using the NetworkX library, the

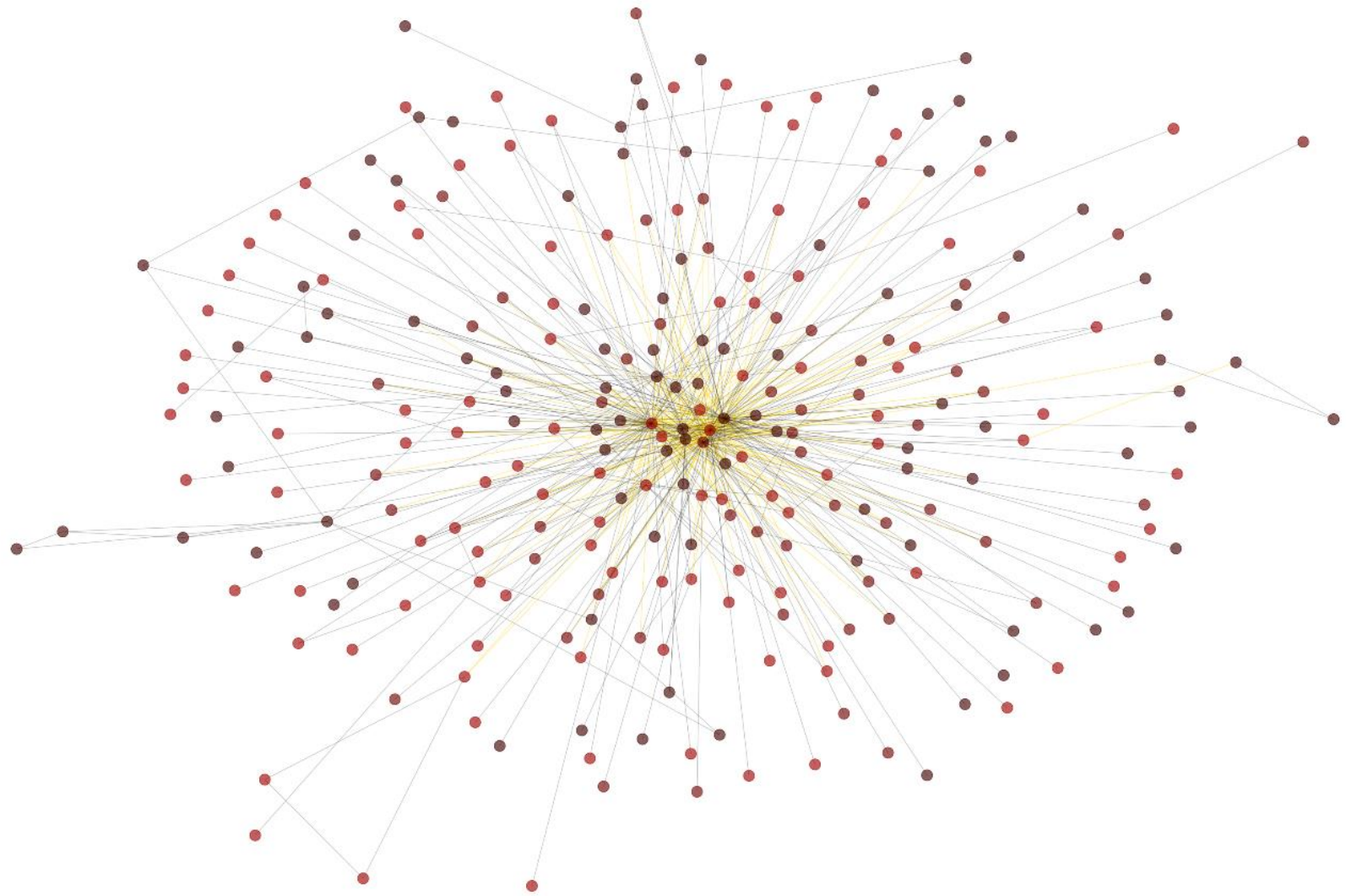


Matplotlib library, and the greedy modularity maximization algorithm described in a previous section. In both figures, the nodes' colors indicate which community they belong to. In the figure corresponding to the OpenSSL network, green edges signify connections across different communities, and black edges signify connections *within* communities. In the figure corresponding to the LE network, gold edges signify connections across different communities, and black edges signify connections *within* communities.

Because the same number of communities are found in both networks, the resulting communities in the LE trust network are larger than those of the OpenSSL trust network, with the direct result that we observe higher total trust levels in the OpenSSL communities than in the LE communities. Additionally, in the OpenSSL network we observe greater inter-community connections than intra-community connections, while in the LE network the mix of the two edge types (connections) is more balanced. One interpretation of this could be that communities within the first network are more insular, leading to higher levels of trust, while communities in the second network are just as likely to engage in cross-community interactions as they are in intra-community interactions, decreasing trust levels in the overall network.



**Figure 90:** Greedy modularity maximization community detection in the OpenSSL empirical trust network. Green colored edges signify edges across communities, and black edges signify edges within communities.



**Figure 91:** Greedy modularity maximization community detection in the LE empirical trust network. Gold colored edges signify edges across communities, and black edges signify edges within communities.

In Table 26, we present the results of the greedy modularity maximization community detection algorithm in the context of trust. Although we find the same number of communities in both networks, we find a smaller mean community size in the OpenSSL trust network compared to the LE trust network; this is to be expected, as the overall size of the OpenSSL trust network is also smaller. In examining which (if any) of the top 10 most trusted nodes as measured by our transitive trust metric defined in this chapter are found in the largest community, we find in the OpenSSL trust network six of the top 10 most trusted nodes are present in the largest community, while only four of them are found in the largest community in the LE trust network. We interpret this as additional evidence that trust levels are higher in the OpenSSL network than in the LE network.

**Table 26:** Greedy modularity maximization (GMM) community detection statistics for OpenSSL and LE trust networks

	OpenSSL trust network	LE trust network
# communities (GMM)	8	8
ave. size of communities	19	33
largest community as percentage of total network	58%	64%
# top 10 trusted nodes contained in largest community	1,3,6,7,8,10	1,2,3,10

## **Conclusions and Future Work**

In this section we consider the research questions that were posed at the beginning of this chapter and interpret them in light of our results from the previous section. After doing so, we present a discussion of limitations of this chapter’s results and future work to expand the findings of this chapter.

With regard to *"RQ6-1: What is the structure and topology of a trust network in free and open source software (FOSS) projects?"* in the preceding sections we have presented an exhaustive analysis of the structural and topological properties of two major FOSS projects in the year 2015. With respect to both trust networks, we have showed that they exhibit classic characteristics of small world networks. Most importantly, they both exhibit a power law distribution in their node degrees; a power law distribution is indicative of a scale-free network, which are one kind of small world network. This finding could have important implications for thinking about how open source projects grow and evolve, and potential implications for security of those projects. We have described the nature of both direct trust and transitive trust – both of which arise directly from the structure of the network itself – in the OpenSSL and the LE networks, and have verified the validity of using traditional centrality measures (eigenvector centrality) to infer direct trust and the usefulness of our new method for inferring indirect trust. We have described the structure of online FOSS project networks in terms of modularity and communities, in terms of connectivity, assortativity, and reciprocity.

With regard to *"RQ6-2: Does trust lead to the network structures that we see in online FOSS networks, or do FOSS networks by their nature give rise to the types of trust relationships that we observe?"*, while we have not yet been able to definitively answer this question, we have found strong evidence that the trust networks analyzed in this chapter are anything but random. We have not yet identified a network generative model that replicates the

creation of these networks, but we have provided convincing evidence that the need for such models exists as the networks are non-random.

With regard to "RQ6-3: *What is the relationship (if any) between the trust network and security incidents in FOSS projects?*", we find that based on our analyses described in this chapter – counter to our original suspicion – there appears to a *relationship between higher trust levels and more frequent security incidences in a FOSS project*. While this statement requires many caveats (described in the following section detailing limitations of this work), we nonetheless find higher levels of trust by several measures in the OpenSSL trust network than in the LE trust network. During the analysis period, there were seven known CVEs associated with the OpenSSL project while there were none associated with the LE project in 2015. One possible explanation for this finding may be that in these types of networks, when trust is higher contributors may unintentionally and inadvertently give less scrutiny to contributions than they would in contexts where trust is lower, possibly leading to security issues. As was discussed in [6.3] trust is a crucial ingredient in online FOSS projects and they wouldn't exist as we know them without trust. Thus, there may be an important paradox present in online FOSS projects of needing to increase trust to enable project contributions to continue, and yet including additional mechanisms to ensure that high levels of trust don't lead to less scrutiny of the security (and quality) of contributions.

### Limitations

This chapter considered hand picked examples of interactions from high-profile FOSS development projects. It may be the case that these examples represent anomalies in

the overall population of FOSS projects. Although within our selected projects we had a large sample size, within the world of FOSS the sample size was small. Thus, caution should be taken in trying to generalize the results too broadly, recognizing that a broader sample of FOSS projects will be needed to draw more robust conclusions. It may be the case that this type of analysis is much more difficult to do on smaller FOSS projects with fewer documents, revisions, lines of code, and collaborators. Conversely, larger sets may benefit from additional small world analysis, as described in [6.23]. From the analysis performed in this chapter, we cannot claim a causal relationship between developer network trust and higher or lower incidence of cybersecurity vulnerabilities; this chapter only highlights evidence of *some* relationship between the two. Interpretation is necessary if attempting to apply these findings to any *specific* FOSS development project. The small sample size used precludes us from performing meaningful statistical analysis (such as regression). However, this work uses empirical data from known interactions, and so provides a platform appropriate for a larger scale experiment. This approach is valid for making inferences about these kinds of relationships but may be more challenging to implement such measurements into a real-time trust system, when the outcome is not known ex-ante (as in these examples).

### Future work

Future research should expand this line of investigation, utilizing the methods and tools described in this chapter as a roadmap. Future research should extend the analysis to a longitudinal analysis, including both more samples of more projects within a given

timeframe as well as expanding the time horizon analyzed (only one year was considered in this chapter). It will be useful to include some consideration of the complexity of the software in question (experience and research indicate that more complex software will inevitably have more vulnerabilities introduced over time compared to simpler software). Additionally, it will be useful to include a consideration of the number of developers who view the code on a regular basis, as the number of eyes viewing code should result in fewer bugs – or, at least, in the bugs being discovered and resolved more quickly.

Temporal aspects were not considered in this chapter, but doing so is another natural extension of this work. Issues related to this include how the trust networks grow and evolve with respect to time, event analysis (examining how or if the trust networks change when a new CVE is announced, or when the vulnerability related to it is first discovered), and identifying (or developing) a network generative mechanism that parallels the actual generation and evolution of these types of trust networks.

This chapter described initial observations of interest to engineers who are designing new Internet-based information or communication systems or maintaining existing ones. This chapter raises intriguing possibilities for developing document management system architectures that may be more resistant to reverse engineering attempts by attackers. Nonetheless, there is much additional work and future research needed to be able to deepen the understanding outlined here. It will be important to expand the sample size of projects considered, to give sufficient statistical explanatory power. It will also be useful to consider different types and measures of trust beyond the



ones mentioned here, and to further explore the sensitivity of the measures considered in this chapter to different combinations and variations of network characteristics. This chapter takes a narrow definition of trust; future research should consider different types of trust outlined by other scholars. Similarly, this chapter considered only a few trust metrics; others should be tested, too. It may also or instead be the case that a trust metric specially designed for applicability to online FOSS projects may be needed. This chapter considers document collections in the context of FOSS, but it is conceivable that the same ideas may be tested and valid in other domains such as proprietary (non-open source) software development, technical publications, and others.

## **VII. Applying the Systems Engineering Process to designing a trust management tool for Reddit**

### **Introduction**

In the preceding chapters of this dissertation we have largely investigated the issue of trust in online networks from an observational perspective: *how can platform managers go about selecting a trust metric from the dozens that have been proposed?; how does trust affect security in GitHub?; how does trust affect spread of misinformation on Twitter?*

In this chapter, we seek to apply our findings from the preceding chapters. We outline our process of systematically designing, developing, and testing a functional prototype of an agent for helping online community managers leverage trust for improving the quality of exchanges in their networks.

Based on our extensive literature review in Chapter III we have, surprisingly, identified little work that has been done to examine trust on Reddit, and how it affects community quality. Reddit is an online social network, news aggregator, and discussion forum founded in 2005 (more detail on Reddit is provided in a subsequent section).

As was discussed in Chapter III, the most effective trust management systems are designed for application in a specific network such as a P2P file sharing network, an e-commerce platform, or an online social network like Twitter. Based on our extensive

literature review conducted for Chapter III of this dissertation, together with a supplemental context-specific literature review for this chapter, we have yet to find previous work by other researchers that proposes a trust management system for Reddit.

Thus, in this chapter, we outline a Systems Engineering-driven approach to designing, prototyping, and testing a trust management tool specifically designed for use in Reddit, which we dub *Coni the Trust Moderating Bot*. Upon the discovery of a lack of trust management systems for Reddit, based on the literature we initially thought a peer-to-peer focus for a Reddit trust model would be useful, but through the course of applying the Systems Engineering process to our design challenge we discovered a more useful and more specific challenge to solve for: the need for improved trust-driven automated tools for Reddit moderators which make use of the transitive property of trust for online systems. In this chapter, we describe this user discovery process as part of the broader Systems Engineering Process.

Although we cover the high points of several key parts of the Systems Engineering Process in this chapter, we also recognize that there are certain key elements of the process which we don't include in this chapter in order to leave sufficient room for presentation and discussion of our results; these include a risk analysis, functional support plans (since this is an open source project not associated with an enterprise), or an operations and maintenance plan. Additionally, we recognize that it is more common for this process to be applied to large teams with different disciplinary backgrounds requiring extensive planning, coordination and integration. Nonetheless, we demonstrate the

usefulness and effectiveness of the process for projects of different types, sizes and scopes by applying it to the design of our Reddit trust tool. We believe that the application of the Process to our design project resulted in a higher-quality end result because it forced us to focus on specific user problems.

In the following sections, we provide additional background on the Reddit platform; describe related research work; consider the extensive ecosystem of Reddit moderator bots that have already been developed; describe our methods for defining, designing, and testing our bot; and analyze and discuss the results.











This chapter's contributions include demonstration of the effectiveness of the Systems Engineering Process for application to Web-based software and social media systems; proposal of a method for collecting and organizing transactions data on Reddit for use in a trust management system; presentation of a complete blueprint for a Reddit bot that uses trust to perform moderating actions in a subreddit; and application of several of this dissertation's key findings from earlier chapters.

## **Overview of Reddit**

Reddit is referred to by many as “the front page of the Internet”. Founded in 2005 by Steve Huffman, Alexis Ohanian, and Aaron Swartz, today Reddit claims more than 50 million daily active users, and is consistently one of the top 10 most-visited websites on the entire Internet. Reddit users can follow news stories, learn a new programming language, study for certifications, watch cute videos of puppies, get advice for speculating on financial investments, share art work, or ensconce themselves in a political echo chamber. Thus,

Reddit in this sense truly is a *platform* that plays host to a vast array of different topics, viewpoints, and user end-goals.

Reddit is home to thousands of discussion forums dedicated to a specific topic, which are known as subreddits. Figure 92 lists the top 10 subreddits by number of subscribers, as of Sept. 2022. It's also interesting to note that there is only a loose correlation between the size of a subreddit and how active it is (by frequency and number of posts and comments).

Rank	Subreddit	Subscribers
1	 <a href="#">announcements</a>	173,960,845
2	 <a href="#">funny</a>	43,087,630
3	 <a href="#">AskReddit</a>	37,285,957
4	 <a href="#">gaming</a>	34,336,147
5	 <a href="#">aww</a>	32,122,028
6	 <a href="#">Music</a>	30,652,073
7	 <a href="#">worldnews</a>	29,691,349
8	 <a href="#">pics</a>	29,337,228
9	 <a href="#">movies</a>	29,063,439
10	 <a href="#">todayilearned</a>	28,794,384

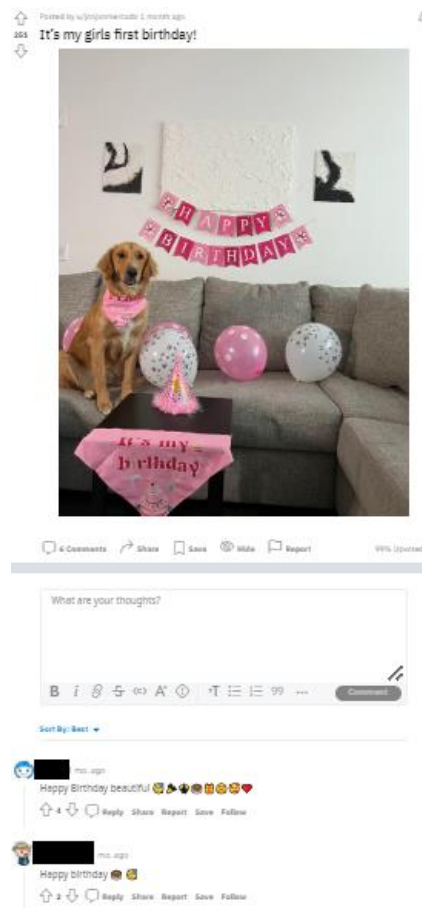
**Figure 92:** Top 10 subreddits by number of subscribers, Sept. 2022. Source: <http://redditlist.com/all>

There are several different types of agents active on Reddit. The primary category of agents are users (redditors), moderators, administrators, and bots. Each subreddit has several moderators (and, in the case of the largest subreddits, many moderators) who do their best to ensure compliance with that subreddit's rules, as well as with general Reddit rules. In most cases, moderators are unpaid volunteers who agree to serve as moderators

out of their passion for the particular subreddit they're moderating. Especially for many of the smaller subreddits, the moderators are also often the same people who first created the subreddit. Additionally, Reddit uses administrators, who have many of the same privileges and responsibilities as moderators, and are Reddit employees (rather than volunteers). Administrators tend to focus more on compliance with general Reddit rules and policies. Finally, there are also numerous bots used in Reddit to perform different tasks (including automated moderation). The first primary category of Reddit bots are those that openly operate as bots (they don't try to conceal the fact that they are bots). Some Reddit bots automatically perform their given tasks (such as the AutoModerator bot), while others need to be invoked by a user (such as the RemindMe Bot, which when a user tells it to will remind them of something at a specified point in the future). Additionally, as is the case on other social media platforms, there is a second category of bots which attempt to conceal the fact that they are bots from other users. The reasons for designers of these types of bots to conceal the true nature of these bots are numerous, but they are often malicious – ranging from spamming to coordinated spread of disinformation and more.

On Reddit, a user (known as a Redditor) can create a post, which is equivalent to opening a new discussion thread in an online forum. Frequently, when users create a post they will link to an external article, include a photo or video, and some commentary or interpretation of the same. Alternatively, they may post an anecdote and ask advice about how to resolve the situation they're in, or ask a question about a specific technical domain.

Once a post has been created, users can comment on the post. Using the previous examples, users might give their reaction to the news article, photo or video that was shared; they might give advice on how to resolve another user's problem; or they might answer a specific how-to question within a technical domain. Of course, posts may also veer off topic or into the personal; for example, posts are frequently made to correct another user's grammar or spelling, or to share memes.



**Figure 93:** Example of a post and comments from Reddit

Users can also upvote or downvote comments and posts, give awards or “Reddit gold” to comments or posts to demonstrate their approval or disapproval of the content of

a post or comment. Some users may feel so passionate about a post or comment that they spend real-world money to purchase virtual Reddit coins and then award Reddit Silver, Reddit Gold, or Reddit Platinum (depending on the amount spent) as a gift to the user whose post or comment they strongly approve of.



**Figure 94:** Examples of awards given to a post on Reddit

Each user who has an account (Reddit can be viewed without an account, but to post, comment, upvote or downvote, requires an account) has a “karma score” attached to their account, which is related to that user’s upvotes and downvotes; however, importantly karma scores are not calculated with a simple one-to-one relationship of upvote (or downvote) to karma point [7.1]. The karma score is directly related to this dissertation’s focus of trust in networks, because in many subreddits users with sufficiently high karma scores can bypass automatic moderation of their posts and comments as a result of being designated a de-facto trusted contributor (this does not, however, mean that moderators of a subreddit can’t moderate their activities at all).

Finally, users and moderators can also designate “flair”. Flair are essentially tags which moderators can utilize to provide additional control over their subreddit, together with providing an opportunity for users to share more of their personality. Flair can be applied to a user account, to a post, or both. One example of flair is to include the home city next to the user’s account name when posting in a specific subreddit. Moderators can



apply flair to users, and then set rules in their subreddit to only allow users with flair (“flaired users”) to perform certain actions.

### **Related Work**

[7.2] extends work of other researchers and combines multiple disciplinary perspectives to develop deeper understanding of the relationship between regular users and moderators on Reddit, and also to extend the understanding of how the various built-in functionalities of Reddit affect development of trust among users and moderators. Among other interesting findings, the author notes that trust on Reddit in large part is linked to authority of a user or a subreddit. Authority on Reddit, in turn, is associated with expertise in a given domain (typically the domain that a given subreddit is dedicated to), and/or a history of high-quality posts. Linking these findings to the functionalities of Reddit (referred to as affordances), the author observes that the use of flair in a subreddit can lead to high levels of trust in a subreddit if managed properly by moderators, but can in fact degrade trust within a subreddit if managed improperly (or not managed at all) by moderators.

[7.3] finds that, similar to many other social or sociotechnical networks, the phenomenon of “answer-people”, i.e., individuals who for various (typically intrinsic) motivations are prolific question answerers, are present on Reddit. Answer-people may or may not be experts, but they are generally trusted by the community. The answer-people feature a large number of connections compared to regular users of a subreddit, as they interact with many different users in answering their questions. The authors also find that,

in general, it is rare for users on Reddit to be significantly active (i.e., acting as answer-people) in more than one subreddit, which lends support to our proposed model (described later in this chapter) wherein an important measure of trust is trust within a subreddit – not overall karma on Reddit generally. This concept in other settings has also been referred to as a *trust neighborhood* [7.4].

[7.5] examines how, in the face of intentional content manipulation by malicious actors, moderating practices by Reddit as well as by volunteer moderators (non-employees) affects trust within subreddits, and on Reddit generally.

In [7.6] the authors examine how vote (upvotes, downvotes) manipulation on Reddit affects the visibility of subreddits and of threads in those subreddits. Through a series of experiments, the authors examine the effects of artificial vote boosting of threads. They find that in subreddits which normally receive comparatively little upvote/downvote activity, artificially boosting the votes of a thread causes massive effects in increasing the thread's visibility. On the other hand, in subreddits in which it is typical for threads to receive large numbers of upvotes (either because there are more users, or they are more active, or both) then the effect – while still positive and large – is less pronounced.

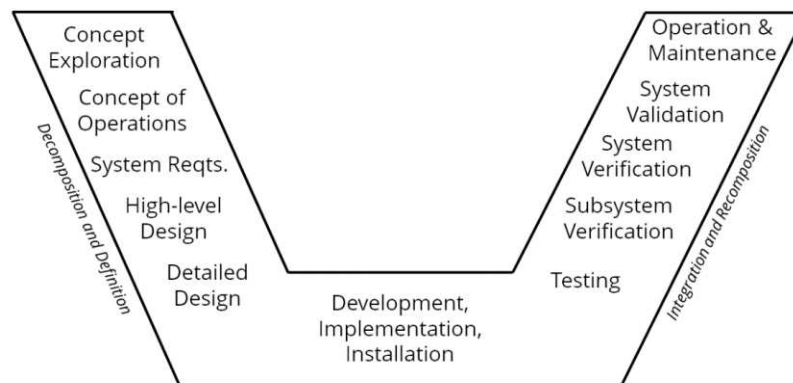
## **Methods**

To identify gaps in the existing solutions for managing trust (and, more generally, managing community quality) on Reddit, we apply the Systems Engineering Process to understand needs of users, and systematically design a solution that meets these needs. In

this section, we describe the methods to be used in this chapter; in the following section, we detail how we applied each of these methods to design our trust tool.

### Overview of the Systems Engineering Process

To design and prototype Coni the Trust Moderating Bot, we follow the Systems Engineering Process, as illustrated in Figure 95.



**Figure 95:** The Systems Engineering Process (or Systems Engineering V). Author's own work, adapted from [7.7]

Not depicted in this figure is a critical “step zero” for setting the direction of the design project, which is needs analysis. In the needs analysis phase of the project, we aim to “show clearly and convincingly that a valid operational need (or potential market) exists for a new system” [7.7, Chapter 6]. This phase helps to answer the “why?” for why a new system is needed, and – if it truly is needed – how it will be different from other existing solutions. One key output from the needs analysis phase is a set of operational requirements (what will the system *do*?).

Following needs analysis, referring to Figure 95 the next step is concept exploration. In the concept exploration phase, the primary objective is to “convert the operationally

*oriented view of the system derived in the needs analysis phase into an engineering-oriented view required in the concept definition and subsequent phases of development” [7.7, Chapter 7].*

One of the key outputs of the concept exploration phase is a set of performance requirements, which are a translation of the operational requirements from the previous phase into a version which can be readily implemented and tested by engineers.

Performance requirements should be defined as part of a formal and careful process of requirements analysis, wherein the operational requirements are analyzed, refined, and then validated. Another important output of this phase is a set of candidate system concepts, which are different possible conceptual combinations of capabilities that, taken together, will support performance (and operational) requirements thus far established.

This is derived from sub-steps of the concept exploration phase of implementation concept exploration and performance requirements validation.

After concept exploration concludes, we proceed with concept definition. In the concept definition phase we *“define the functional and physical characteristics of a new system...that is proposed to meet an operational need defined in the preceding conceptual phases” [7.7, Chapter 8].* The primary output of this phase of the Systems Engineering process is selection of a single configuration of the system that serves as a baseline from which to iterate design. Activities in this phase include functional analysis and formulation, functional allocation, concept selection and validation, systems architecting, and creation of system functional specifications.

In the concept definition or high-level design phase, the first real work of designing the broad outlines of the system begins. It begins to “characterize the system in sufficient detail to enable its operational performance, time of development, and life cycle cost to be predicted in quantitative terms” [7.7, Ch. 8]. One of the primary outputs of this phase is the selection and definition of a specific configuration of subsystems that will enable the overall system to achieve its stated objectives.

In the detailed design phase, the various subsystems that were identified and sketched in the high level design phase are analyzed and described in greater detail, down to the component level.

In the development, implementation and installation phase, the designs outlined in the high level design and detailed design phases of the Systems Engineering Process are translated into tangible end products (in our case for this chapter, this is the writing and deploying of actual code), implementing the design concepts in the physical (or virtual) world.

In the testing phase, the system’s actual performance is compared to its intended performance, using the set of tests that were designed in previous phases of the development process. Major or unexpected discrepancies between as-installed performance and designed performance are investigated and addressed.

In the verification phases, the design team systematically reviews the design to ensure that the system as designed satisfies all of the requirements that were established in earlier phases of the development process.

## Application of the Systems Engineering Process to Our Design Challenge

In this section, we describe the specific activities and steps we took during several phases of the Systems Engineering Process. For the sake of brevity and to leave sufficient room for discussion and interpretation of the results of our process, we don't detail our use of every single phase of the Systems Engineering Process but instead focus on the most critical and relevant ones for our design challenge.

### Needs analysis

For the needs analysis phase of our design challenge, we conducted a series of interviews with Reddit users. Using the insights we gained from the user interviews, we then performed extensive secondary research, reviewing numerous subreddits, GitHub repositories and documentation, the Reddit API documentation, and other third-party sources.

For the user interviews, we aimed to ensure a diverse mix of user types were represented in our interviews, including light users, frequent users, longtime users, and new users. While each individual interview took its own unique direction based on the interviewee's perspective, we began each interview with a standard set of questions. In total, we interviewed 10 different Reddit users to better understand their experiences with and reasons for using the platform. Users ranged from relatively new (had an account for about 18 months) to seasoned (had an account since approx. 2010); most users engaged with Reddit on the dedicated mobile app, but occasionally (especially for more complex subjects) they would engage with Reddit via the Web on their computers; most were

frequent but relatively light users, checking Reddit at least a few times per week but usually spending 15 minutes or less per session; users' purposes for using Reddit varied from seeking to learn a new skill to staying up to date on news items to entertainment; most users did not pay any particular attention to who the other users were they were interacting with, paying attention primarily to the content instead; most users were fairly passive, only posting or commenting occasionally, relative to the total amount of time they spent on the platform; and, when asked, most users stated that when evaluating the reliability of information or claims on Reddit, they tend to take what they see with a dose of skepticism and, if needed, were prepared to investigate other sources to verify or debunk a claim they saw on Reddit.

After the user interviews were finished, we performed a content analysis of the responses given and clustered them by themes. Below, we highlight several of the key insights from our content analysis that are relevant for designing our Reddit trust bot:

1. Community quality in many subreddits is often perceived by longtime members to decline when a subreddit grows in size (number of subscribers) – particularly when a subreddit grows rapidly in size during a short period of time due to some viral effect or timely new event that causes large numbers of new users to join.
2. Following from (1), at least for the users we interviewed, a critical insight is that trust is generally placed more in a particular subreddit than it is in individual users, as would be more common in other online social media platforms like Twitter or

Facebook. Using this realization, we propose that a trust measure that focuses on understanding trust in a subreddit rather than at an individual level.

- Because on Reddit a subreddit's quality is directly linked to the quality and activity level of that subreddit's moderators, it follows that *trust from average Reddit users is inherently and implicitly placed into a subreddit's moderators* – even if a user never has any interaction with those moderators.

3. At the *aggregate* level Reddit users are very active (billions of engagements happen each month), but when considering the *ratio* of page views to concrete engagements (upvoting/downvoting, posting, commenting, etc.) many users are quite passive – they may read through ten posts in a single day and not interact with any of them, or with the comments contained in those posts. This insight is supported by work from [] wherein the researchers identify an approximate 90-9-1 rule with 90% of users in social media “lurking” (passively reviewing content, but not interacting with it), 9% of users interacting with content through actions like upvoting or liking, and only 1% of users actively posting and commenting.
4. Different from most other popular media platforms like Twitter, Facebook or TikTok, the average Reddit user pays comparatively little attention to who the other users are they're interacting with. Instead, they pay more attention to the *content* and the *communities* with which they're engaging. To be clear, an important element of this is paying attention to the comments from other users, but in the case of Reddit, the average user cares less about *who* makes a comment than the *content* of that comment



Upon identification of these needs, we spend some time to better understand the landscape of existing solutions for moderators on Reddit, both from within Reddit and from third party developers.

### *Existing solutions*

Reddit directly provides several useful tools to help moderators of subreddits in their tasks of moderating. These include AutoModerator, which is a bot that helps moderators to automate some of their common tasks. AutoModerator performs actions specified by the moderators of a subreddit based on rules that the moderators set up. AutoModerator acts on posts and comments and depending on the rules set up by the moderators, can approve, remove, filter, mark as spam, or report the content posted by a user. Additionally, AutoModerator can leave a comment on an item if it meets the specified rules, it can automatically send a private message to the moderators or to the user who posted, and several other actions. On the other hand, AutoModerator cannot re-check content that has already been posted, if that content is later edited. AutoModerator also cannot detect reposts, base its decisions on upvotes/downvotes, cannot make decisions based on flair, cannot make decisions about a user's history a post's history, and several others. Thus, AutoModerator performs critical functions for moderators of subreddits, but its functionality is quite limited in scope and can't consider many of the factors that would go into estimating trust [7.7].

Reddit also provides a suite of other tools for moderators which the moderators have to apply manually. These include community settings, whereby moderators can set rules (which are automatically enforced) related to content, safety, and privacy for their

subreddit. And, of particular interest for this dissertation, Reddit also includes Crowd Control as a tool for moderators, which *"lets moderators automatically collapse/filter comments and filter posts from people who aren't trusted users within their community yet"* [7.8]. Crowd Control provides moderators with more control over user actions (which, as described above, can't be done by AutoModerator).

Additionally, there is an extensive ecosystem of bots developed by third party contributors for purposes for improving moderation activities on Reddit. These third party Reddit moderator bots provide automated moderation functions customized to the specific needs of a particular subreddit. One subreddit, called r/Bot with more than 2000 members, provides a forum within Reddit for moderators to discuss moderator bots, and to request specific automatic moderation functions from volunteer third party developers. An example is Clippy, which scans users' posts and awards them points for high quality posts, with the goal of encouraging more thoughtful discourse in subreddits [7.11]. The BotDefense bot helps subreddit moderators to identify and ban other bots which are posting spam or performing other undesirable actions [7.12]. AssistantBot (also known as Artemis) helps subreddit moderators to better organize their communities by enforcing flair and by gathering statistics on the community [7.13]. RepostSentinel is a moderator bot that helps to detect reposts of existing posts [7.14]. Reposting of popular posts is one strategy used by malicious bot accounts and by malicious user accounts in an attempt to quickly and artificially inflate their karma score. Another moderator bot utilizes machine learning to automatically detect and remove hate speech in a subreddit [7.15]. User \_pacjax

[7.16] has developed a moderator bot that calculates scores of users in a subreddit, and ranks them based on scores. However, the source code for this bot is not currently available for inspection on an open platform like GitHub, so neither moderators of subreddits nor developers can inspect the code to clearly understand how it works – which is an important aspect of increasing trust in these types of software tools.

Finally, there is also a robust set of third party tools called Toolbox. Toolbox is implemented as a browser extension for the Chrome browser (not a Reddit bot) which enables subreddit moderators to perform actions including user history analysis, tools for modifying the moderation queue, and more [7.17]. Toolbox claims more than 8000 moderators have installed and used their set of tools.

#### *Operational requirements*

Based on our user interviews as well as our secondary research, subreddit moderators would benefit from having an addition tool at their disposal to help them make a quick and rough judgement as to how much to trust an unknown user seeking to join or post in their subreddit. In Table 27 we present the operational requirements of such a tool.

**Table 27:** *Operational requirements for our trust tool*

OR #	Operational Requirement
O1	A Reddit trust tool should be simple and straightforward to install and operate for a non-technical moderator
O2	The recommendations given by a Reddit trust tool should be easy to interpret and explain for non-technical moderators
O3	A Reddit trust tool should make the job of the moderator easier, faster, and/or more effective
O4	The Reddit tool should increase the quality of service (QoS) in a given community (subreddit) in which it is used

### Concept exploration

We considered two primary concept alternatives. In the first, we would rely primarily on existing bots for providing data related to users, posts, and subreddits that has already been processed and transformed into some form of reputation or quality assessment. In the second, we would obtain data directly from the Reddit API, process and transform it ourselves, and utilize these data for computing trust and then performing actions based on these trust levels.

### *Concept of Operations (CONOPS)*

Operational requirements detail the “why?” of a system. In addition to the why, we also consider the “how?” and the “who?” which, in the needs analysis phase of the Systems Engineering Process, manifests itself in the form of a Operational Concept/Concept of

Operations (CONOPS) statement. A CONOPS statement for designing Coni the Trust

Moderating Bot follows:

1. Increase the quality of interactions among users in subreddits (quality of interaction is defined later on), using the principles of trust in online networks that have been detailed in earlier chapters of this dissertation, and by numerous other researchers
  - a. Improve the “I” of the CIA triad – integrity of information
2. Using pre-existing tools including the Reddit API, Python, and various Python libraries, provide a tool for moderators of subreddits to know how much or if to trust users posting in their subreddit when there is otherwise no previous history of interaction with said user.

#### *Operational context (Scenarios)*

The operational requirements answer the “why?” of a system, the CONOPS addresses the “how?” and “who?” of a system, and the remaining piece – the “where?” and the “when?” – of how the system is to be used is addressed through scenarios (operational context description).

The primary and typical scenario in which Coni is needed and for which she is designed is that of moderating untrusted or distrusted users’ actions within a single subreddit. Typically, these subreddits are smaller in terms of number of subscribers (the most popular subreddits generally receive strong support from volunteers who develop tools and help enforce community standards) and may be focused on fairly mundane subjects. Frequently, previously unknown users may join these subreddits, and moderators

are unlikely to know how to treat these users initially with no history in the subreddit. Most of the time, these unknown users are acting in good faith and seeking to authentically engage with the community. However, in a non-negligible portion of cases, unknown users may be malicious actors with various goals: to post or spread spam; to spread disinformation; to spread hate or other toxic language; to promote illegal activity or to attempt to commit cybercrime (like stealing users' credentials through phishing); and other malicious acts. These malicious actors may come and go in a seemingly random fashion; other times, there may be some external news event which causes a sudden and unexpected influx of malicious users intent on some coordinated goal.

#### *Candidate features considered for Reddit trust system*

To identify which features are available for potential use in our Reddit trust system, we refer to the Reddit API documentation and the PRAW documentation [7.18]. PRAW (from **P**ython **R**eddit **A**PI **W**rapper) is a wrapper for the Reddit API to interact with it using the Python language. We use PRAW both to retrieve data needed for our trust computations, and also to send commands to our bot when it needs to perform an action.

We consider four classes from PRAW for our Reddit trust system: subreddits, Redditor (users), Comments, and Submission (posts). Examining the documentation for each of these classes, we identify the methods and attributes listed in Table 28 as candidate features which could be used in construction of our Reddit trust system. We return to this list of candidate features later in our design process to select a smaller subset of these features for use.

**Table 28:** Candidate PRAW features for use in Reddit trust system

<b>subreddits</b>	<b>"Redditor" (users)</b>	<b>"Comments" (comments)</b>	<b>"Submission" (posts)</b>
created_utc	comment_karma	author	author
display_name	comments	created_utc	author_flair_text
id	submissions	distinguished	comments
public_description	created_utc	id	created_utc
subscribers	has_verified_email	parent_id	distinguished
controversial	id	replies	id
	link_karma	score	is_self
	subreddit	submission	link_flair_text
	subreddit["public_description"]	subreddit	locked
	subreddit["subscribers"]	subreddit_id	name
	controversial	controversial	num_comments
	downvoted		score
	gilded		subreddit
	gildings		title
	trophies		upvote_ratio
	upvoted		controversial

*Trust insights from earlier in this dissertation*

Our results from Chapter IV, Chapter V, and Chapter VI indicate that there may be an ideal middle ground between too much trust and not enough trust for supporting healthy functioning of a community. We highlight several of these findings which will be useful for design of our trust system in this chapter.

In Chapter IV, we identified ranges of graph structural characteristics that appear to be correlated with improved accuracy in estimating ground truth trust values by trust metrics.

In Chapter V, we found that the global trust levels were markedly higher within conspiracy-oriented communities that were responsible for spreading misinformation on Twitter than the global trust levels in non-conspiracy communities.

And, In Chapter VI, we found that higher global trust levels within a community of open source software developers was correlated with higher rates of cybersecurity vulnerabilities (though no causation can be claimed based on the work completed in this dissertation). We consider these findings as we go about designing our trust bot for Reddit.

We will utilize these insights from our earlier research to inform the design of our trust metric, to be proposed in a later section.

### System requirements

Based on the operational requirements established in an earlier section, we begin to translate these into more specific system requirements. In Table 29 we present functional and performance requirements for Coni the Trust Moderating Bot. We develop these requirements using the operational requirements as our guide and starting point, and to ensure that operational requirements are met, in Table 29 we provide a direct trace from each functional requirement back to at least one operational requirement. We identify a total of 22 different top-level functional requirements which flow from the four operational requirements.



**Table 29: Functional Requirements and Performance Requirements for Coni the Trust Moderating Bot**

	Trace to OR	FR #	Functional Requirements	Performance Requirements
Data retrieval and manipulation module	O4	1.1	The trust tool must be capable of retrieving data from the Reddit API	Request and receive data at least 1x/minute
	O4	1.2	The trust tool can manipulate <i>string</i> data	No more than 4 bytes/character
	O4	1.3	The trust tool can manipulate <i>int</i> data	No more than 24 bytes/int
	O4	1.4	The trust tool can manipulate <i>float</i> data	No more than 24 bytes/float
Graph construction module	O2, O4	2.1	Construct subreddit structure graph	Use standard graph data structure
		2.1.1	Add edges from users to posts they create	
		2.1.2	Add edges to comments from posts they reference	
		2.1.3	Add edges between comments	
	O2, O4	2.2	Construct trust overlay network	Implement trust model specified in this chapter
		2.2.1	Increase edge weights between user and comment for each time the comment created by the user is upvoted	
2.2.2		Decrease edge weights between user and comment for each time the comment created by the user is downvoted		
2.2.3		Increase edge weights between user and post for each time the post created by the user is upvoted		
	2.2.4	Decrease edge weights between user and post for each time the post created by the user is downvoted		
Trust computation module	O4	3.1	The trust tool must be able to quickly compute trust scores for subreddits	Subreddit trust score computation time takes 10 seconds or less
	O4	3.2	The trust tool must be able to quickly compute trust scores for users	User trust score computation time takes 1 second or less (per user)
	O4	3.3	The trust tool must be able to quickly compute trust scores for posts	Post trust score computation time takes 2 seconds or less (per post)
	O2, O3	3.4	The trust tool can identify trusted users	Assign a specific flair to trusted users
	O2, O3	3.5	The trust tool can identify untrusted users	Assign a specific flair to untrusted users
	O2, O3	3.6	The trust tool can identify distrusted users	Assign a specific flair to distrusted users
Trust Storage Module	O4	4.1	The trust tool can write trust scores for a user within the subreddit	Trust scores can be written to storage in 1s or less
	O4	4.2	The trust tool can read trust score for a user within the subreddit	Trust scores can be read from storage in 1s or less
	O4	4.3	The trust tool can modify (including deleting) trust scores that have already been written	R/W permission must be granted to the trust bot
Bot module	O2	5.1	The trust tool can display a user's trust score to the moderator	Display trust score in a human-readable form (stars, letters, colors, or similar)
	O2	5.2	The trust tool can display a subreddit's trust score to the moderator	Display trust score in a human-readable form (stars, letters, colors, or similar)
	O2	5.3	The trust tool can display a post's trust score to the moderator	Display trust score in a human-readable form (stars, letters, colors, or similar)
	O2	5.4	The trust tool can designate trusted, untrusted, and distrusted users	>= 95% F1 score
	O4	5.5	The trust tool can modify privileges for users within the subreddit according to their trust level	Dynamic and automatic privilege management, with ability to override by moderators
	O1, O3, O4	5.6	The trust tool should be capable of retrieving data from and sending data to other moderator bots, as necessary	Interface with AutoModerator, Fact-Checker-Bot, AssistantBOT
	O1	5.7	The trust tool can be installed and activated without any coding	Moderator engages with bot through graphical user interface

## High-level design

Following the definition and analysis of system requirements, we advance to high-level design of Coni the Trust Moderating Bot. We complete three primary steps within the high-level design phase: design of our trust computation system; design of system architecture diagram; and design of a functional flow block diagram (FFBD).

### *Our system*

Before we can continue any further with design of our Reddit trust system, we first need to solidify how we will compute trust values on Reddit. After detailed consideration and exploration of the insights from the needs analysis, the operational requirements, and the system requirements defined in previous sections of this dissertation we propose the following methods for computing trust in Reddit.

First, as was outlined previously, we seek to support moderators of subreddits in their ongoing charge of fostering high-quality discourse within their subreddits. As a result, we focus our trust system on measuring and managing trust within a single subreddit – not across the entirety of Reddit.

In this chapter we don't propose a new way to *compute* trust – we utilize PageRank for computing direct trust, and our generic transitive trust (GTT) metric first presented in Chapter IV of this dissertation for computing indirect trust. Instead, our method includes two novel additions. The first of these is in deciding which values (available to us from the Reddit API) to use as trust values and how to weight them, before inputting them into our trust measures. The second novel contribution is the utilization of the computed trust

scores by an autonomous agent (Coni the Trust Moderating Bot) to perform actions and/or give recommendations to moderators for improving the quality of service of a subreddit.

Our method takes as input a trust overlay network, which we describe how to construct in the following sections. To aggregate trust values which will be input into the trust metrics (PageRank and GTT), we propose the following.

Taking a subreddit as the unit of analysis, we first propose computing a total global trust value for a given subreddit, to be used by that subreddit's moderator(s) (via Coni the Trust Moderating Bot) as a measure of how healthy and vibrant their community is. Higher trust values should be correlated with more frequent and high-quality discourse within their subreddit.

Additionally, we propose computing a within-subreddit trust value for each *user* who interacts with the subreddit. This within-subreddit trust value will also be used by the subreddit's moderators (via Coni the Trust Moderating Bot) to specify or recommend privileges for users, and to encourage or discourage different actions by the subreddit's users.

#### *Why the standard Reddit karma score isn't sufficient*

As described at the beginning of this chapter, Reddit already includes a rough measure of trust in users, so why do we need another one? Reddit karma measures actions taken across the *entirety* of Reddit – not within a specific subreddit – and thus, while the Reddit karma score may provide useful information as to the direction of a user's trustworthiness, it is less useful for application in any one subreddit.

One reason for this is that malicious actors have numerous methods at their disposal for manipulating and artificially boosting their karma scores. Methods include copying popular posts or comments and reposting them as their own with the expectation of many upvotes; pay 'click farms' to upvote content; and stealing credentials of established accounts to make use of the established account's karma (until it gets shut down by Reddit).

Additionally, consider an example where a user has a total karma score of 10,000, with 9,000 of that karma score having been earned through actions in the History subreddit (r/history) because of high-quality posts and comments. The user is a trained, professional historian and thus has many opportunities to share her expertise with r/history. The fact that 90% of the user's karma score was earned in the r/history subreddit is indicative that the user is highly-trusted in this subreddit. The user's high karma score of 10,000, having been earned primarily in r/history, is much less useful in helping us to assess the trust (or expected trust) that should be placed in the user if she then goes to engage in the Technology (r/technology) subreddit – but Reddit makes no distinction about where or how karma was earned when publicly displaying a user's karma score.

For these reasons, we focus in on the gap of a subreddit-specific trust score for users as a method for predicting the quality of a user's contributions within a subreddit and, by extension, the long-term quality of discourse within a subreddit. Reddit does not provide this for users by default, but it can be feasibly computed using Reddit's API.

*Our system*

In this section, we describe our proposed system for computing trust values for any user that engages in a specific subreddit (whether or not they are subscribers to that subreddit), which can then be used by the bot module of our system to take actions (or make recommendations, depending on the permission level granted to it by a moderator) to improve the quality of a subreddit.

As was discussed earlier in this chapter, we can compute trust based on actions taken by users with respect to posts and with respect to comments. From the candidate features outlined previously in Table 28, in Table 30 we present the features selected as those to be used for computing trust in our system. In the table, features highlighted in blue are those which are directly used as inputs into trust scores; features in which are used as informational or reference features to help organize the data (such as "id"). We select these features from the larger list of candidate features in part because they are primarily graph-based features which require little to no context to apply to a trust measure.

**Table 30:** Selected features for computing trust in our system

subreddits	"Redditor" (users)	"Comments" (comments)	"Submission" (posts)
created_utc	comment_karma	author	author
display_name	comments	created_utc	author_flair_text
id	submissions	id	created_utc
subscribers	created_utc	parent_id	id
	has_verified_email	replies	is_self
	id	score	link_flair_text
	link_karma	submission	name
	subreddit	subreddit	num_comments
	subreddit["public_description"]	subreddit_id	score
	subreddit["subscribers"]		subreddit
	gilded		title
	trophies		upvote_ratio

To compute total trust in a subreddit, we propose computing the total trust of each user who is active within that subreddit – based solely on interactions that take place *within* that subreddit. To compute the trust in each user, we propose computing the trust placed in that user’s posts and that user’s comments made *within the subreddit under consideration*. From these premises, we propose our trust aggregation system as follows.

To compute the total trust invested in each user, we apply the following method. First, each user is given a base trust score. To this base trust score, we add the sum total (positive or negative) of trust scores that derive from the posts or comments made by the user. For all users in the subreddit, we then normalize their raw trust scores.

To compute a user’s trust:

1. *Compute the base trust of the user,*
2. *Compute the sum of trust in all posts created by the user within the given subreddit,*
3. *Compute the sum of trust in all comments created by the user within the given subreddit,*
4. *Add (1) + (2) + (3) together to obtain the raw trust value,*
5. *Normalize by the total raw trust score within the subreddit under consideration*

To compute a user’s base trust value:

1. *Compute the user’s account age; if account age > 1 year, add 100 points to their base trust,*

2. *Check to see if the user has a verified email address; if yes, add 100 points to their base trust,*
3. *Get all of the trophies for the user, and for each trophy add 1000 points to their base trust,*
4. *Get all of the awards for the user, and for each award add the “value” of that trophy to their base trust (refer to the “This is a list of known global awards” section in [7.19] for award values),*

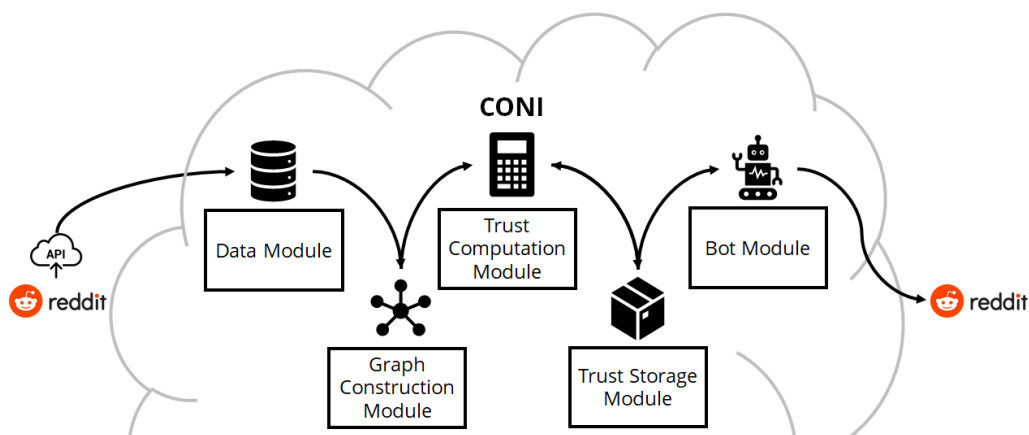
To compute a post’s or a comment’s trust:

1. *Using the upvote ratio and the score, calculate the total number of upvotes and total number of downvotes:*
  - a. *total votes = score / upvote ratio,*
  - b. *upvotes = score,*
  - c. *downvotes = total votes – upvotes*
2. *Get the total number of comments that have been made on the post*
3. *Get the total number of awards for the post; for each award, add the value of the award to the post’s raw trust score*
4. *For each upvote, add 0.5 trust*
5. *For each downvote, add -1 trust (we posit that intensity of the negative reaction that provokes a downvote is, all other things being equal, greater than the intensity of the positive reaction that provokes an upvote)*
6. *For each comment made to the post, add 0.5 trust*

7. Add (4) + (5) + (6) to obtain the raw trust value,
8. Normalize the raw trust value

### System architecture diagram

In Figure 96, we present the system architecture diagram for Coni. This diagram derives from the system requirements defined in an earlier section, together with the trust system defined in the previous section. Through using the Systems Engineering Process, we were able to identify the proper relationships among the different modules (in an earlier design, we had the graph construction module depending on the trust computation module, rather than the reverse as in the design manifested in the figure). Our system takes inputs of data from Reddit (via the API), holds them temporarily in the data module, then passes them to the graph construction module where they are organized into a structure that can be readily processed by the trust computation module. When trust scores and levels are determined in the trust computation module, they are passed to the trust storage module, where they are made available to the bot module to enable it to make recommendations and take actions on the Reddit platform.



**Figure 96:** System Architecture of Coni the Trust Moderating Bot



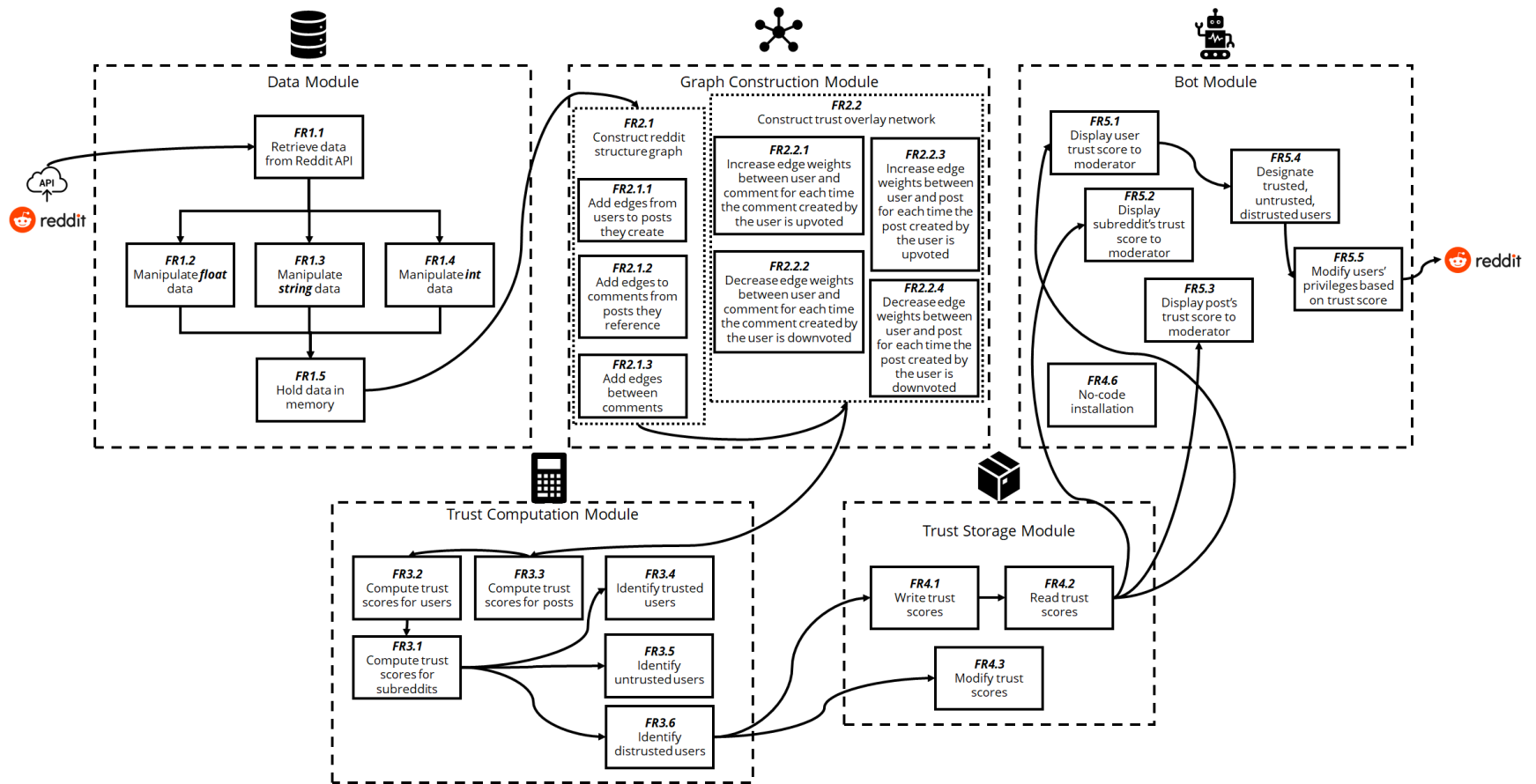


Figure 97: Functional Flow Block Diagram for Coni the Trust Moderating Bot

### *Functional flow block diagram*

To better conceptualize the relationships between different functional requirements and the timing of when different functions need to be executed for Coin the Trust Moderating Bot, we develop a functional flow block diagram (FFBD) for our system, which we present in Figure 97. The figure provides a visual representation of how data enters our system, flows within it, is transformed, and is utilized to perform actions which are then manifested back on the Reddit platform.

### Detailed design

In this section, we describe our work that was completed as part of the detailed design phase of our overall design process. Utilizing the results of the high-level design phase, we propose a trust computation model specific to Reddit, and then describe the design of the primary software modules needed to implement and utilize scores from the trust model. For the sake of brevity, we don't describe the trust storage module in detail, since it is similar to any other storage module one would encounter in a typical Web-based software application.

### *Data retrieval module*

The primary data source for a Reddit trust tool is through Reddit's API, accessed using PRAW. Reddit makes a multitude of detailed public information available through its API, including detailed information about users and their actions and histories, posts, comments, and related items. We perform an in-depth review of the PRAW documentation [7.20] to better understand its capabilities, and how different components and functions might fit together as part of a trust tool.

The data retrieval module is responsible for interacting with the Reddit API (via PRAW, in our case) and pulling data about users, subreddits, posts, and comments which will be needed later to perform the trust computations. The data retrieval module takes as inputs commands about which data to collect and the data from the Reddit API, and outputs these data to the trust computation module. Pseudocode for the data retrieval module is presented in Figure 98.

```

For user in Redditors:
    Get the user's (id, has_verified_email, created_utc) #get metadata
    Get the user's (comment_karma, link_karma, submissions, comments, trophies, gilded) #get data for trust computation

For subreddit in Subreddits:
    Get the subreddit's (id, display_name, public_description, subscribers) #get metadata
    Get the subreddit's (comments, posts) #get data for trust computation

For comment in Comments:
    Get comment's (author, created_utc, id, parent_id, replies, score, submission, subreddit, subreddit_id) #get metadata
    Get comment's (replies, score, upvote_ratio) #get data for trust computation; score is the total number of upvotes, upvote_ratio is the percent of upvotes out of total votes

For post in Submission:
    Get the Submission's (author, author_flair_text, created_utc, id, is_self, link_flair_text, name, subreddit, title) #get metadata
    Get the post's (num_comments, score, upvote_ratio) #get data for trust computation; score is the total number of upvotes, upvote_ratio is the percent of upvotes out of total votes

```

**Figure 98:** Pseudocode for the data retrieval module of Coni the Trust Moderating Bot

### Graph construction module

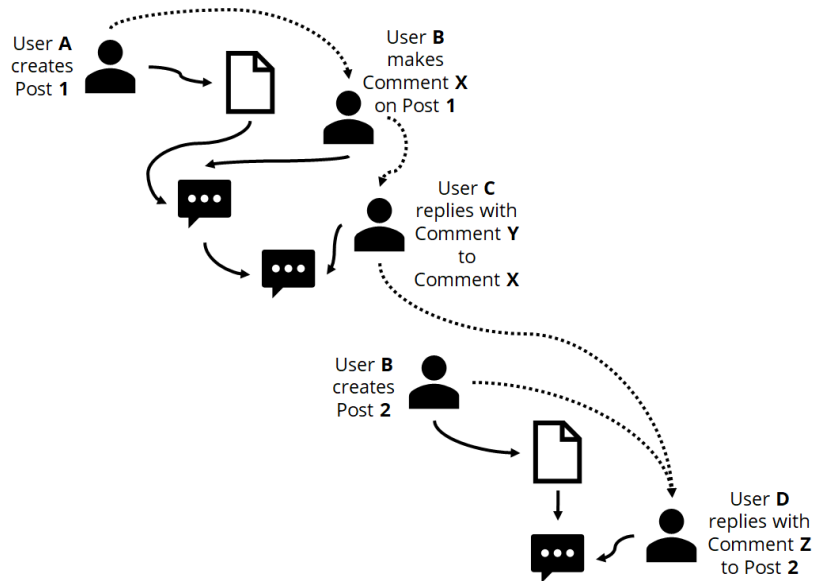
The graph construction module is responsible for organizing the data from the data retrieval module into a structure that can be processed by our trust model. It takes as inputs data about users, posts, comments, and subreddits from the data retrieval module, and it outputs a structural graph and a trust overlay network in the form of edge lists. The trust overlay network, in turn, serves as the input for the trust computation module.

### *Constructing a subreddit graph*

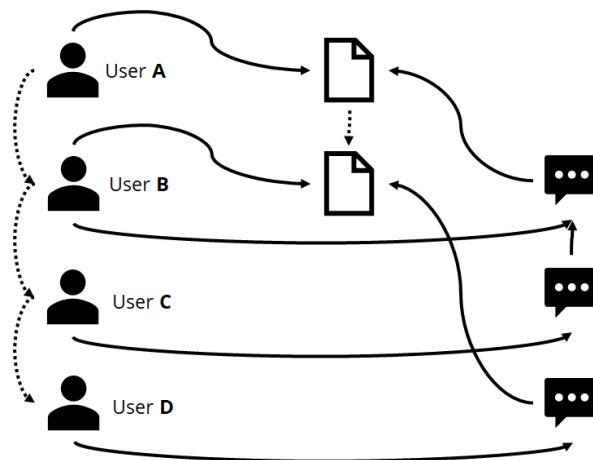
Before computing trust values, we need to first structure the available Reddit data in a way that will allow us to do efficiently and correctly compute trust values. To do so, our first step is to construct a graph from the available Reddit data. This graph will serve as the structural foundation, from which we will next construct the trust overlay network (trust graph). The boundaries of analysis for Coni the Trust Moderating Bot are a single subreddit, and thus, the graphs we construct represent the structure of a specific subreddit. The subreddit structure network will be constructed as an unweighted, directed multigraph. A multigraph allows for parallel edges between the same pair of nodes; inclusion of directionality is key, since comments have a source and a target. In the following step (construction of the trust overlay network), we will collapse multiple parallel edges into single, weighted edges.

From the Reddit API we can obtain data about users, posts that the users create, and comments that users make about a post. We consider each of these objects to be nodes, but nodes of different classes. When a user creates a post, a directed out-edge from the user to the post will be created. When a user creates a comment within a post, a directed out-edge from the post to the user is created. And, when a user replies to another user's comment, an out-edge is created from the reply to the previous comment. We illustrate this structure conceptually in Figure 99. In this figure, solid lines represent direct actions that link the two nodes at either end of the edge; dashed lines represent connections that are indirectly created as a result of user actions (i.e., User **A** is *indirectly* connected to User **B** when **B** comments on the post made by **A**). Following, in Figure 100,

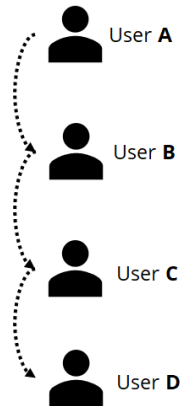
we present a version of the same graph from the previous figure that considers the network as a bipartite graph, wherein users only directly connect to posts or comments (not to other users). Finally, in Figure 101 we provide the user projection network, wherein users connect to one another indirectly as a result of the posts and comments they make.



**Figure 99:** Conceptual structure of a subreddit graph



**Figure 100:** Bipartite subreddit graph

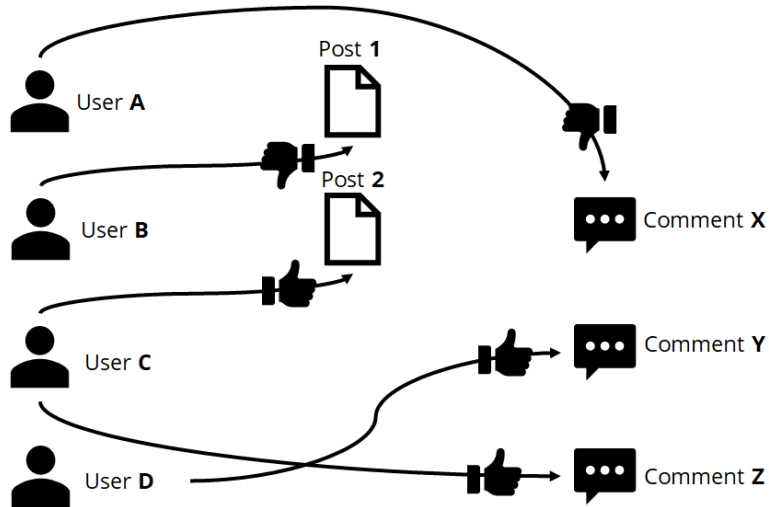


**Figure 101:** Network projection of user graph within a subreddit

#### *Constructing a trust graph*

Using the previously-constructed graph as a structural foundation, we next proceed to add in actions taken by users in the subreddit under consideration that will be used to compute trust values. We refer to this more detailed network as the trust overlay network, and it will be a directed, weighted graph.

Starting from the Reddit structural graph, we add in data related to upvotes, downvotes, number of comments, trophies, etc. for each post and comment. Each of these actions are completed by a user (i.e., a post cannot upvote another post, a comment cannot upvote a comment, etc.) and ultimately are aimed at another user (with posts and comments serving as intermediate nodes between user nodes). Taking a simple example that continues with the example from the previous section, we obtain Figure 102.



**Figure 102:** Conceptual trust overlay network

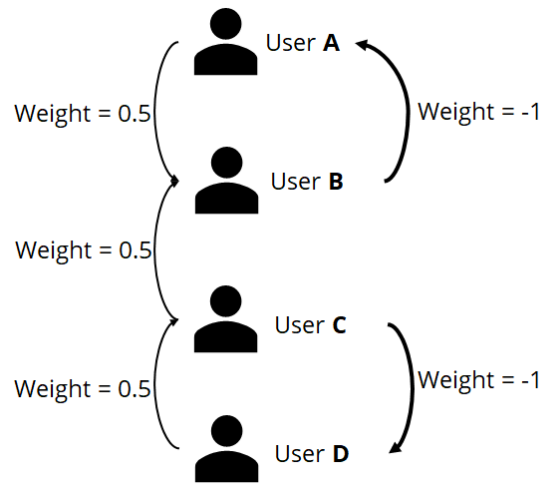
In the example above, using our method we would compute total global direct trust in each object as in Table 31, and edge weights for trust relationships between users as in Table 32. From the computations of Table 31 and Table 32, we can then construct a trust graph which, for this example, looks like Figure 103.

**Table 31:** Sample computation of raw trust values for each object in the example

Object	Raw trust values
User A	-0.5
User B	0.5
User C	0.5
User D	0.5
Post 1	-0.5
Post 2	1.5
Comment X	-1
Comment Y	0.5
Comment Z	0.5

**Table 32:** Trust adjacency matrix for sample trust calculation

	A	B	C	D
A		0.5		
B	-1			
C		0.5		0.5
D			0.5	



**Figure 103:** Resulting trust overlay network for this example

### Trust computation module

The trust computation module is responsible for implementing the trust computation model which we proposed and described earlier in this chapter. The trust computation module takes as its inputs the graph data generated by the graph construction module.

The trust computation module computes trust scores for a given subreddit and for the users who have engaged with that subreddit. Once trust scores are computed, they are sent as outputs to the trust storage module. Pseudocode for the trust computation module is presented in Figure 104.



```

Get a list of all users who have engaged with this subreddit
  For post in posts:
      Get post (author), append to authors
  For comment in comments:
      Get comment (author), append to authors

Compute each user's base trust
  create user, trust dictionary trusts{}
  For user in users:
      account_age = today - created_date
      if account_age > 365 days, age = 100; else, age = 0
      if has_verified_email = true, verified = 100; else, verified = 0
      base_trust = (verified) + (age) + (trophies * trophy_value) + (awards * award_va
      append user, base_trust to trusts{}

Iterating through each user (author) active in this subreddit and get their posts and comments
  create user, transactions dictionary transactions{}
  For author in authors:
      Get author's (posts, comments)
      For posts, comments:
          Get post's/comment's (score, upvote_ratio, num_comments)
          total_votes = score / upvote_ratio
          upvotes = score
          downvotes = total_votes - score
          transactions = (upvotes * 0.5) + (downvotes * -1) + (num_comments * 0.5)
          append author, transaction to transactions{}

Compute direct trust using PageRank and NetworkX
  with graph, for node in graph:
      pageranks = NetworkX.PageRank(graph)
      append node, pagerank to trusts{}

Compute indirect trust using GTT and NetworkX
  compute geodesic path lengths
  MTPD = geodesic path length rounded up to nearest whole number
  with graph, number of samples:
      for number of samples:
          source = random node, target = random node
          if path(source->target) > MTPD, trust = 0
          if path(source->target) <= MTPD and >= 1, trust = 1/path
          if no path(source->target), trust = 0
          append target, trust to trusts{}

Normalize the raw trust scores
  Get the sum of trusts in trusts{}
  For user, trust in trusts{}:
      trust_norm = trust / (sum of trusts{})

Identify users as trusted, untrusted, or distrusted
  create user, trust_level dictionary trust_levels{}
  For user in users:
      If trust > 0.67, trust_level = 1 (trusted)
      If trust > 0 and < 0.67, trust level = 0 (untrusted)
      If trust < 0, trust level = -1 (distrusted)
      append user, trust_level to trust_levels{}

```

**Figure 104:** Trust computation module pseudocode

### *Bot module*

Finally, the bot module – which is the one module in our system that can take an action (if a moderator grants it the proper permission) on Reddit – takes input from the trust computation module in the form of trust levels for users, and based on these inputs performs actions on behalf of the moderators of a subreddit.

In our initial design, the bot module can take the following actions if it has the proper permissions granted to it by a subreddit’s moderator:

- Display a user’s trust score to moderator
  - Assign (either automatically or manually, depending on moderator’s preference) “trusted” flair to trusted users, to allow them to post/comment without restriction
- Display subreddit’s trust score to moderator
  - If total subreddit trust level is too high (refer back to “Trust insights from earlier in this dissertation” section of this chapter), admit more untrusted users (or lower the threshold for becoming a trusted user), and
  - If total subreddit trust level is too low, increase the threshold for trusted users
- Flag a distrusted user’s posts/comments for manual review by moderators
- Give awards (through the *gilding* method in PRAW) or trophies to trusted users to encourage others to follow their example of high-quality posts or comments

## Prototyping and Development

We develop a working prototype of Coni the Trust Moderating Bot using Python3, PRAW to interact with the Reddit API, Pandas to organize and manipulate the data, NetworkX to construct the graphs, and NumPy to perform the trust computations. Using the system architecture diagram, FFBD, and pseudocode designed and presented in the previous section, we translate these artifacts into working Python code, which we can then test using the testing plan described in the following section. For our prototype, we do not implement the trust storage module, instead passing trust values directly to the bot module. This method works properly for testing, but for a real-world application the trust storage module should not be neglected, as it is what will allow trust information to remain persistent for a moderator over time.

## Testing

Continuing with application of the Systems Engineering Process, we propose a testing plan that uses a combination of validation with empirical datasets and simulations for evaluating the performance and the effectiveness of Coni the Trust Moderating Bot.

To test the first module of our system – the data retrieval module – we directly test the ability of our code to retrieve data via the Reddit API and hold in memory for subsequent use. The Reddit API does not require any credentials to pull data from it, but it does require authentication to be able to perform actions on Reddit (such as those which we propose for Coni). Thus, as part of the data retrieval module we also include an authentication function, to login to our test account created for the purposes of this

chapter's design challenge. Through inclusion of diagnostic messages in our code (for example `print("logging in")` when first initiating the authentication function, and `print("logged in")` once the authentication function has successfully completed), we verify that all functional requirements are executing properly in this module.

To test the second, third and fourth modules of our system (graph construction module, trust computation module, and trust storage module) we propose utilization of an existing third party Reddit dataset such as [7.21]. Measures to be tested for the graph construction module include checking to ensure the graphs as generated are properly connected and with the correct directionality; for the trust computation module we check that no errors are returned when computing trust values, and that trust values computed by our code align with randomly-selected spot-checked trust values that we compute manually for comparison.

To test the final module – the bot module – we propose testing using a combination of empirical validation and simulation. While the previous tests allow us to test our modules' functionality without affecting anyone else, because the bot module is designed to take actions in the "real" (virtual) world, we must take additional precautions when testing it. Reddit has a subreddit specifically designed for developers to test the functionality of their bots, called the "testing" (r/test) subreddit. We can use this subreddit to test the functionality of our bot module for making posts or comments. However, to test moderator capabilities (such as assigning flair to a user based on that user's computed trust level), we will need to have our own subreddit in which we can assign moderator

privileges to our bot. As a first test before such a “live” test, we instead propose a simulation. We can develop a simulated subreddit in Python, generating random users, random posts and comments from those users, and random actions (upvotes and downvotes) of users with respect to posts or comments. We can then compute trust values within this simulated subreddit, and feed them to the bot module to test her ability to complete actions that align with the various trust scores.

### **Discussion of Results**

In this chapter, we have applied the Systems Engineering Process to the design challenge of improving quality of service on the Reddit platform. We’ve provided a complete and feasible conceptual design for a Reddit bot – Coni the Trust Moderating Bot – that can help moderators improve the quality of service within the communities (subreddit) that they modify. Coni provides estimates of trust for specific subreddits, and trust of users engaging in those subreddits. Coni’s trust estimates are more context-specific than the universal karma score provided by Reddit for all users.

Through application of the Systems Engineering Process, we identified a specific set of challenges faced by users and moderators of subreddits (communities) within Reddit. From this needs analysis, we proposed a novel method for aggregating transactions on Reddit to be used in as inputs into existing trust metrics (PageRank and GTT). We completed high level design of Coni the Trust Moderating Bot, translating operational requirements into functional and performance requirements. These functional and performance requirements, in turn, were translated into a system architecture design and

a functional flow block diagram, to help us better understand how data flow in our system, and how they need to be transformed to achieve our purposes. Next, in the detailed design phase, we break our overall system down into five smaller subsystems, or modules: a data retrieval module, a graph construction module, a trust computation module, a trust storage module, and a bot module. We proposed specific methods for how to collect and aggregate transactions data from Reddit and transform these into trust graphs, and then trust scores. We designed specific actions that the bot module, with the proper permissions from a subreddit's moderator, can perform in a subreddit to monitor and manage the quality of service within that subreddit.

### **Limitations**

While useful, this chapter's contributions also have their limitations. One of the most important ones (which should also be addressed through future research, as described in the following section) is the limited work done thus far to explore the robustness of the tool's trust estimates by performing sensitivity analyses. This chapter's work only considers a few basic combinations of attributes and values for those attributes to be used in computing trust values. Additionally, this chapter's system does not (yet) include any mechanism for mitigating effects of attacks on the trust measurement system.

### **Future Work**

Future work should extend both the functionality and the robustness of our design, and carry out further evaluations as to the effectiveness of the designs.

More robust testing of the prototype created in this chapter should be carried out before finalizing development of the complete system. Tests should include both empirical analysis and simulations. The empirical tests should check the validity of the trust estimates using a much larger empirical dataset; candidate datasets could include [7.21] or [7.22] as examples, or construction of a new dataset for these purposes using the Reddit API. In the empirical tests, user rankings of general Reddit karma scores can be compared against user rankings of the within-subreddit trust scores computed using this chapter's methods. Simulations can be used to further test the effectiveness and robustness of the bot module's abilities to monitor and manage interactions within a community (subreddit). Upon completion of additional testing, the findings from the tests should be used to incorporate design improvements into a complete development.

In addition to more testing and development, future work should also explore the sensitivity of the trust measures to the inclusion or exclusion of different features identified in Table 28, and also how any such changes affect the code execution speed for Coni.

Coni's functionalities may also be extended in future work to enable her to interact with other Reddit bots. For example, Coni may be extended to be able to request data from AssistantBOT, which is a Reddit bot that tabulates statistics for different users and subreddits. Coni could also be extended to function with other non-Reddit APIs to add even more capabilities related to content of posts and comments, such as sentiment analysis or fact checking. While this chapter focuses on graph-based measures for estimating trust in subreddits, additional information can be added to the trust equation by considering

content of posts and comments made by users. To this end, numerous sentiment analysis libraries have been developed which could be integrated with Coni, and Google offers a fact checker API [7.23].

Future work could also examine the effectiveness of expanding the realm of analysis for users' trust. When computing trust for a specific subreddit, future work could also take into account the trust levels present in other subreddits in which users are also engaged; in this sense, such an extension would function similar to PageRank, treating entire subreddits as pages (rather than the comments, posts, and users within those subreddits, as we did in this chapter), but to do so would first need to collapse each subreddit into a single trust value. As part of this work, we could consider how activity in high-quality or low-quality (or controversial) subreddits affects the usefulness of trust measures.

Finally, Coni in her current form does not include an attack resistance mechanism, other than those inherent directly in the design of Reddit's transaction systems. Most other successful trust management systems such as EigenTrust propose a basic (non attack-resistant) version, and an attack resistant version. In our case, types of attacks that Coni could be subjected to could include sybil attacks (wherein a coordinated group of fake accounts are created with the purpose of increasing the reputation – in this case, the karma score), badmouthing attacks, or ballot stuffing.



## **VIII. Discussion, Limitations, and Future Work**

In this chapter, we summarize the findings of this dissertation, putting them into context (including discussion of the limitations of the findings), and propose future research work to extend the work completed in this dissertation.

### **Discussion of this dissertation's contributions**

Throughout this dissertation, we have completed an investigation of trust in online settings that is both broad and deep. We provided an updated and comprehensive overview of the current state of the field of research related to online trust metrics and trust management systems (Chapter III). In Chapter IV, we completed an objective analysis of the performance of several well-known trust metrics (both direct trust and indirect trust), measured by accuracy of the trust estimates and speed of code execution. In Chapter IV, we also collapsed the findings of the experimental benchmarking work into generalized categories of trust metrics and graph structural characteristics. This allowed us to produce a heuristic model that can be used by others (particularly practitioners) who may be considering use of a trust management system for their online networks as an initial screening tool. Our heuristic model can help them narrow the field of candidate trust management systems that will be most fitting for their network's needs. In Chapter V, we explored the relationship between trust and the spread of misinformation in online social media (which, according to the US Cybersecurity and Infrastructure Security Agency, is a

form of cybersecurity threat) by examining a labeled dataset from Twitter. In this chapter, we contributed to knowledge in the field through discovery of small worldness effects in online conspiracy trust networks; through an exhaustive characterization of the structural and topological forms taken by online misinformation trust networks; and (among other contributions) through the interesting discovery of the key role played by brokers in the spread of misinformation (users who were active both in the non-misinformation networks and the misinformation networks). In Chapter VI, we explored the relationship between trust and security vulnerabilities in open source developer networks on GitHub. Among other findings, this chapter contributed to knowledge and understanding in the field through discovery of small worldness in online developer networks; through an exhaustive characterization of the structure and topology of online open source developer networks; through discovery of an inverse relationship between security vulnerability incidences (as measured by CVEs) and trust levels in the developer network; and through discovery of evidence supporting the need for a graph generative model to be developed for use in open source online developer networks. In Chapter VII, we contributed to knowledge and understanding in the field through production of a comprehensive blueprint and prototype system and agent for understanding and managing trust on Reddit; application of the Systems Engineering Process to a Web-based open source design challenge, demonstrating its usefulness in this domain; in both cases, to the best of our knowledge our work is the first to address these questions from the Systems Engineering perspective.

## **Limitations**

While we have produced contributions to several areas of the field, this dissertation's findings also have their limitations.

In Chapter IV, the heuristic framework we developed is constrained by several limitations. Chapter IV's experiments only considered global trust, not local trust. Global trust is a useful measure for managers of platforms who have access to data about the platform, but is less useful for inferring trust relationships between specific nodes (which local trust will do). Also, our framework was derived only from graph structural characteristics, so when considering platforms that have access to other data types such as content or behavior this framework will be incapable of considering these measures. We also don't consider all categories of trust models (though our framework does encapsulate the most popular categories of trust management systems). Our framework is derived from a limited dataset and limited number of experiments, so the sensitivity of results should be explored further. Finally, the framework only provides heuristic guidance – not specific quantitative predictions – as to the direction and approximate intensity of performance metrics' relationships to graph structural characteristics.

In Chapter V, we only considered direct trust measures, not indirect trust measures. Thus, there are likely additional insights to be drawn from the dataset using indirect trust measures which our work did not explore. The work of Chapter V also only considered graph characteristics for estimating trust; additional insights can most likely also be discovered by a content-focused analysis of the data, too. Another important of Chapter V's

findings is that, while large, the dataset we used focused only on misinformation stemming from conspiracy theories related to the COVID-19 pandemic; it could be the case that the spread of misinformation of a different origin may respond differently to varying trust levels in a network.

In Chapter VI, we only used hand-picked examples of interactions from high-profile FOSS development projects. It could be that these examples represent anomalies in the overall population of FOSS projects. Our selected projects had a large sample size over the *within* the projects, but within the universe of FOSS the sample size was comparatively small. It may be that this type of analysis is more difficult to do on smaller FOSS projects (smaller than the ones we analyzed in this dissertation) which feature fewer documents, fewer revisions, fewer lines of code, and fewer contributors. Conversely, larger sets than the ones we examined in this dissertation could benefit from additional small world analysis. Our analysis does not enable us to claim a causal relationship between developer network trust and higher or lower incidence of cybersecurity vulnerabilities; our work only underscores evidence that there is *a* relationship between the two. Interpretation is necessary if attempting to apply these findings to any *specific* FOSS development project. Finally, our approach is valid for making inferences about these kinds of relationships but may be more challenging to implement such measurements into a *real-time* trust system, in which the outcome (in our case, the presence or absence of a CVE) is not known ex-ante.

Chapter VII's results are useful, but also have their limitations. A critical limitation (which should also be addressed as part of future research on the subject) is the limited

work done thus far to explore the robustness of the tool's trust estimates by performing sensitivity analyses. Chapter VII's work only considers a few basic combinations of attributes and values for those attributes to be used in computing trust values, and – critically – our design does not yet include any mechanism for mitigating effects of attacks on the trust measurement system, which would limit its usefulness in a live application as malicious agents will inevitably discover and attempt to leverage this fact.

### **Future Work**

In considering future research directions related to this dissertation, we first point out several high-level directions that will be promising. Next, we also summarize and recapitulate more specific and narrowly-focused future research recommendations related to the work from each chapter.

This dissertation utilized some methods and techniques from artificial intelligence (AI) and machine learning (ML) such as community detection, clustering, search and optimization, and others. However, the focus of this dissertation was not specifically on utilizing these techniques. For example, further development and application of link prediction, trust neighborhood prediction, classification of nodes with respect to trust are all promising areas for future research (and are areas which many researchers are already pursuing).

This dissertation only gave basic consideration to temporal aspects of trust networks. However, this (temporal aspects of graphs) is an area of active research for many

network scientists, and it is an area of research that would help to further advance the understanding of the nature and effects of trust on networks.

Turning our attention to more specific and narrowly-defined future research directions resulting from each of this dissertation's chapters, from Chapter IV we note the need to focus on better understanding the robustness and sensitivities of the various measures investigated in the chapter. Future work related to Chapter IV should also evaluate differences between performance of trust metrics applied to directed v. undirected graphs; this chapter considered only undirected versions, but inclusion of benchmarking on directed graphs represents an important extension for real world networks. This is a critical area for future research, since PageRank (one of the trust measures used in Chapter IV's experiments) was originally designed for use in directed networks (though it can still be applied to undirected ones). Experiments should be run for more iterations than we did in our work to smooth out differences owing to random factors, and add statistical power to the findings. Additional trust metrics should also be benchmarked. Our experiments only consider global trust values, so future benchmarking research should also consider local trust values.

Related to Chapter V's work, future research should consider models for epidemic spreading on networks (which include temporal aspects as referenced elsewhere). A traditional SIS model could be adapted and may prove useful for modeling epidemic spread of misinformation in online networks, and how this relates to trust. We posit that  $\beta$  (the probability of contact with an infected node) in the classical SIS model can be

estimated using the proportion of misinformation nodes in the overall network, while  $\gamma$  (the probability of recovery) can be approximated using the rate of contact of broker nodes (discussed in depth in Chapter V) with the rest of the population. Future research related to Chapter V should also explore generative network models for reproducing online misinformation networks. If existing generative network models are unable to adequately reproduce the generative process for online misinformation networks, efforts should be made to develop a generative model that mimics the process for generation of online misinformation trust networks. Having such generative models available could be useful in helping to slow the spread of online misinformation – particularly when paired with other methods like AI/ML models that identify misinformation.

Future work related to Chapter VI should extend the analysis to a longitudinal analysis, including both more samples of more projects within a given timeframe as well as expanding the time horizon analyzed (only one year was considered). It would also be useful and informative to consider the complexity of the software in question (experience and research indicate that more complex software will inevitably have more vulnerabilities introduced over time compared to simpler software) as a control. Additionally, it will be useful to include a consideration of the number of developers who view the code on a regular basis, as the number of eyes viewing code should result in fewer bugs – or, at least, in the bugs being discovered and resolved more quickly. Temporal aspects were not considered in Chapter V, but doing so is another natural extension of this work, including investigating how the trust networks grow and evolve with respect to time, and event

analysis (examining how or if the trust networks change when a new CVE is announced, or when the vulnerability related to it is first discovered). Additionally, identifying (or developing) a network generative mechanism that parallels the actual generation and evolution of these types of trust networks is an important area for future research.

Finally, future work related to Chapter VII should extend both the functionality and the robustness of our design, and carry out further evaluations as to the effectiveness of the designs. More robust testing of the prototype created in Chapter VII should be carried out; tests should include both empirical analysis and simulations, as well as sensitivity analysis of the parameters included in the trust measure developed for use by the bot. Our bot's (Coni the Trust Moderating Bot) functionalities may also be extended in future work to enable her to interact with other Reddit bots, as well as to interact with third party APIs (for example, Google's fact checker API). Future work could also examine the effectiveness of expanding the realm of analysis on which users' trust scores are based. When computing trust for a specific subreddit, future work could also take into account the trust levels present in other subreddits in which users are also engaged; in this sense, such an extension would function similar to PageRank, treating entire subreddits as pages.



## References

### **Chapter I**

- [1.1] RH Coase, "The nature of the firm." *Economica* 4.16 (1937): 386-405.
- [1.2] D. Kahneman, Daniel. *Thinking, fast and slow*. Macmillan, 2011.
- [1.3] J. Golbeck, "Computing and applying trust in web-based social networks". PhD Dissertation, University of Maryland, College Park, 2005.
- [1.4] Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030. 2022. Statista Web page. Retrieved from <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>
- [1.5] P. Suci, "Americans Spent On Average More Than 1,300 Hours On Social Media Last Year", *Forbes*, 2021. Web page. Retrieved from <https://www.forbes.com/sites/petersuci/2021/06/24/americans-spent-more-than-1300-hours-on-social-media/>
- [1.6] D. Tighe, "Online shopping behavior in the United States - statistics & facts". Statista Web page. Retrieved from <https://www.statista.com/topics/2477/online-shopping-behavior/>
- [1.7] The 2021 State of the Octoverse. 2021. Web page. Retrieved from: <https://octoverse.github.com/>

### **Chapter II**

- [2.1] Sherchan, Wanita, Surya Nepal, and Cecile Paris. "A survey of trust in social networks." *ACM Computing Surveys (CSUR)* 45.4 (2013): 1-33.
- [2.2] Ghafari, Seyed Mohssen, et al. "A survey on trust prediction in online social networks." *IEEE Access* 8 (2020): 144292-144309.
- [2.3] work.nation demo prototype. [n.d.]. Web page. Retrieved from <https://demo1.worknation.io/>
- [2.4] Trust Graph. [n.d.]. Retrieved from <https://github.com/trustgraph/trustgraph>

### **Chapter III**

- [3.1] R. Ureña, F. Chiclana, and FE Herrera-Viedma. "DeciTrustNET: A graph based trust and reputation framework for social networks", *Information Fusion*, Volume 61, September 2020, Pages 101-112.
- [3.2] Virgule module, available <http://virgule.sourceforge.net/>.

- [3.3] J. Golbeck. "Trust and nuanced profile similarity in online social networks." *ACM Transactions on the Web (TWEB)* 3.4 (2009): 1-33.
- [3.4] W. Yuan et al. "The small-world trust network." *Applied Intelligence* 35.3 (2011): 399-410.
- [3.5] D. Meyerson, KE Weick, and RM Kramer. "Swift trust and temporary groups." *Trust in organizations: Frontiers of theory and research* 166 (1996): 195.
- [3.6] M. Osterloh and S. Rota, "Trust and community in open source software production," *Analyse & kritik* 26.1 (2004): 279-301.
- [3.7] J. Lopez, Jorge, S. Maag, and G. Morales. "Behavior evaluation for trust management based on formal distributed network monitoring." *World Wide Web* 19.1 (2016): 21-39.
- [3.8] IBM, "z/OS Integrated Security Services Network Authentication Service Administration: Realms of Trust", available <https://www.ibm.com/docs/en/zos/2.1.0?topic=service-realm-trust-relationships>
- [3.9] Microsoft Active Directory, "Managing Trusts", available <https://forsenergy.com/en-us/domadmin/html/7296dc81-0672-4023-9937-c060fd7eef2f.htm>
- [3.10] PowerShellMafia, "PowerSploit". Repository on GitHub, available <https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1>
- [3.11] Wang, Yingjie, et al. "A game theory-based trust measurement model for social networks." *Computational social networks* 3.1 (2016): 1-16.
- [3.12] V. Kant and KK Bharadwaj. "Fuzzy computational models of trust and distrust for enhanced recommendations." *International Journal of Intelligent Systems* 28.4 (2013): 332-365.
- [3.13] H. Sapkota, PK Murukannaiah, and Y. Wang, "A network-centric approach for estimating trust between open source software developers," *PLOS One* 14.12: e0226281, 2019.
- [3.14] Li, W. Zhao, J. Yang, and J. Wu, "CoTrRank: Trust Ranking on Twitter," *IEEE Intelligent Systems*, 36(1), 35-45, 2021. <https://doi.org/10.1109/MIS.2020.3045001>
- [3.15] S. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, 2003. Association for Computing Machinery, New York, NY, USA, 640-651. DOI:<https://doi.org/10.1145/775152.775242>.
- [3.16] L. Xiong and L. Liu. "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities." *IEEE transactions on Knowledge and Data Engineering* 16.7 (2004): 843-857.

- [3.17] J. Golbeck, "Computing and applying trust in web-based social networks". PhD Dissertation, University of Maryland, College Park, 2005.
- [3.18] C de Kerchove and P Van Dooren. "The pagetrust algorithm: How to rank web pages when negative links are allowed?." *Proceedings of the 2008 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2008.
- [3.19] R. Zhou and K. Hwang. "Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing." *IEEE Transactions on parallel and distributed systems* 18.4 (2007): 460-473.
- [3.20] FE Walter, S. Battiston, and F. Schweitzer. "Personalised and dynamic trust in social networks." *Proceedings of the third ACM conference on Recommender systems*. 2009.
- [3.21] C. Borgs et al. "A novel approach to propagating distrust." *International Workshop on Internet and Network Economics*. Springer, Berlin, Heidelberg, 2010.
- [3.22] W. Jiang, G. Wang, and J. Wu. "Generating trusted graphs for trust evaluation in online social networks." *Future generation computer systems* 31 (2014): 48-58.
- [3.23] G. Liu, Y. Wang, and MA Orgun. "Trust transitivity in complex social networks." *twenty-fifth AAAI conference on artificial intelligence*. 2011.
- [3.24] W. Xue et al. "DHTrust: A robust and distributed reputation system for trusted peer-to-peer networks." *Concurrency and Computation: Practice and Experience* 24.10 (2012): 1037-1051.
- [3.25] X. Fan et al. "EigenTrust++: Attack resilient trust management." *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. IEEE, 2012.
- [3.26] F. Liu et al. "A Web Service trust evaluation model based on small-world networks." *Knowledge-Based Systems* 57 (2014): 161-167.
- [3.27] S. Agreste, P. de Meo, E. Ferrara, S. Piccolo, and A. Provetti, "Trust Networks: Topology, Dynamics, and Measurements," *IEEE Internet Computing*, 19(6), 26–35, 2015. <https://doi.org/10.1109/MIC.2015.93>
- [3.28] MS Mariani, M. Medo, and YC Zhang. "Ranking nodes in growing networks: When PageRank fails." *Scientific Reports* 5.1 (2015): 1-10.
- [3.29] Trust Graph: Reputation for Decentralized Ecosystems. Available: <https://trustgraph.net/>
- [3.30] N. Thorp, "Decentralized Cooperation needs Decentralized Reputation", Rebooting the Web of Trust I: San Francisco (November 2015). Available: cite <https://github.com/WebOfTrustInfo/rwot1-sf/blob/master/topics-and-advance-readings/DecentralizedCooperationNeedsDecentralizedReputation.md>

- [3.31] Trust Graph, repository from GitHub. Available <https://github.com/trustgraph/trustgraph/blob/master/README.md>
- [3.32] B. Liu, DC Parkes, and S. Seuken. "Personalized hitting time for informative trust mechanisms despite sybils." *Proceedings of the International Conference on Autonomous Agents & Multiagent Systems*. ACM, 2016.
- [3.33] Almuzaini, Fatimah, et al. "WhatsTrust: A trust management system for WhatsApp." *Electronics* 9.12 (2020): 2190.
- [3.34] N. Al-Otaiby, A. Alhindi, and H Kurdi. "AntTrust: An Ant-Inspired Trust Management System for Peer-to-Peer Networks." *Sensors* 22.2 (2022): 533.
- [3.35] X. Meng, Y. Ding, and Y. Gong. "@ Trust: A trust model based on feedback-arbitration in structured P2P network." *Computer Communications* 35.16 (2012): 2044-2053.
- [3.36] J. Lopez et al, "Behavior evaluation for trust management based on formal distributed network monitoring." *World Wide Web* 19.1 (2016): 21-39.
- [3.37] V. Woloszyn and W. Nejdl. "Distrustrank: Spotting false news domains." *Proceedings of the 10th ACM Conference on Web Science*. 2018.
- [3.38] Y. Wang, and J. Vassileva, "Bayesian network-based trust model." *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*. IEEE, 2003.
- [3.39] U. Kuter and J. Golbeck. "SUNNY: A new algorithm for trust inference in social networks using probabilistic confidence models." *AAAI*. Vol. 7. 2007.
- [3.40] A. Jøsang and T. Bhuiyan. "Optimal trust network analysis with subjective logic." *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2008.
- [3.41] O. Richters and TP Peixoto. "Trust transitivity in social networks." *PLOS One* 6.4 (2011): e18384.
- [3.42] S. Song et al. "Trusted P2P transactions with fuzzy reputation aggregation." *IEEE Internet Computing* 9.6 (2005): 24-34.
- [3.43] M. Lesani and N. Montazeri. "Fuzzy trust aggregation and personalized trust inference in virtual social networks." *Computational Intelligence* 25.2 (2009): 51-83.
- [3.44] R. Feng et al "An incentive mechanism based on game theory for trust management", *Security Comm. Networks*, 7, 2318– 2325, 2014. doi: 10.1002/sec.941
- [3.45] W. Jiang et al. "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges." *ACM Computing Surveys (Csur)* 49.1 (2016): 1-35.
- [3.46] H] Li et al. "Exploring the trust management mechanism in self-organizing complex network based on game theory." *Physica A: Statistical Mechanics and its Applications* 542 (2020): 123514.

- [3.47] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank." *Proceedings of the 30th international conference on very large data bases (VLDB)*. 2004.
- [3.48] Z. Abrams, R. McGrew, and S. Plotkin, "Keeping peers honest in EigenTrust." *Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems*. 2004.
- [3.49] Srivatsa, Mudhakar, Li Xiong, and Ling Liu. "TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks." *Proceedings of the 14th international conference on World Wide Web (WWW)*. 2005.
- [3.50] S. Hamdi, et al. "IRIS: A novel method of direct trust computation for generating trusted social networks." *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012.
- [3.51] Q. Pei et al. "A strong and weak ties feedback-based trust model in multimedia social networks." *The Computer Journal* 58.4 (2015): 627-643.
- [3.52] G. Liu, et al, "TOSI: A trust-oriented social influence evaluation method in contextual social networks." *Neurocomputing* 210 (2016): 130-140.
- [3.53] M. Imran, et al, "Calculating trust using multiple heterogeneous social networks." *Wireless Communications and Mobile Computing 2020* (2020).
- [3.54] E. Bertino, E. Ferrari, and AC Squicciarini. " Trust- $\chi$ : A Peer-to-Peer Framework for Trust Establishment," *IEEE Transactions on Knowledge and Data Engineering*, 16.7 (2004): 827-842.
- [3.55] The National Strategy for Trusted Identities in Cyberspace, 2010. Available: <https://obamawhitehouse.archives.gov/blog/2010/06/25/national-strategy-trusted-identities-cyberspace>
- [3.56] National Strategy for Trusted Identities in Cyberspace, 2010. Available: [https://www.dhs.gov/xlibrary/assets/ns\\_tic.pdf](https://www.dhs.gov/xlibrary/assets/ns_tic.pdf)
- [3.57] Identity Ecosystem Steering Group, 2013. Available: <https://web.archive.org/web/20130815232538/http://www.idecosystem.org/page/adherence-ntic-guiding-principles>
- [3.58] National Institute of Standards and Technology (NIST), Applied Cybersecurity Division, "Identity and Access Management Pilots", retrieved from <https://www.nist.gov/pilots>
- [3.59] H. Yu, et al, "A survey of multi-agent trust management systems," *IEEE Access* 1: 35-50, 2013.
- [3.60] M. Tagliaferri, and A. Aldini, "A trust logic for pre-trust computations," *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018.

[3.61] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture, Special Publication (NIST SP)", *National Institute of Standards and Technology*, Gaithersburg, MD, [online], <https://doi.org/10.6028/NIST.SP.800-207>, [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=930420](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=930420)

[3.62] YH Liou and AJ Daly, "Closer to learning: Social networks, trust, and professional communities," *Journal of School Leadership*, 24(4), 753-795, 2014.

[3.63] RM Kramer, "Trust and distrust in organizations: Emerging perspectives, enduring questions," *Annual Review of Psychology*, 50 (1999): 569.

## **Chapter IV**

[4.1] P. Chandrasekaran and B. Esfandiari, "Toward a testbed for evaluating computational trust models: experiments and analysis," *Journal of Trust Management* 2.1: 1-27, 2015.

[4.2] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Computing Surveys (CSUR)*, 45.4, 1-33, 2013.

[4.3] H. Yu, et al, "A survey of multi-agent trust management systems," *IEEE Access*: 35-50, 2013.

[4.4] S. Agreste, P. de Meo, E. Ferrara, S. Piccolo, and A. Provetti, "Trust Networks: Topology, Dynamics, and Measurements," *IEEE Internet Computing*, 19(6), 26-35, 2015. <https://doi.org/10.1109/MIC.2015.93>

[4.5] M.E.J. Newman, *Networks*. Oxford: Oxford University Press, 2010, pp. 169-180.

[4.6] S. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, 2003. Association for Computing Machinery, New York, NY, USA, 640-651. DOI:<https://doi.org/10.1145/775152.775242>.

[4.7] D. Roble (GitHub username danielrobleM), "Simulación de EigenTrust Simple", repository from *GitHub*, 2013. <https://github.com/danielrobleM/-SimulationEigenTrust/>.

[4.8] A. Pahari (GitHub username AyanPahari), "TrustRank Implementation (Python)", repository from *GitHub*, 2021. <https://github.com/AyanPahari/TrustRank-Implementation-Python>.

[4.9] W. Yuan, D. Guan, Y.K. Lee, and S. Lee, "The small-world trust network," *Applied Intelligence*, 35(3), 399-410, 2011. <https://doi.org/10.1007/s10489-010-0230-7>.

[4.10] H. Sapkota, PK Murukannaiah, and Y. Wang, "A network-centric approach for estimating trust between open source software developers," *PLOS One* 14.12: e0226281, 2019.

[4.11] P. Erdős, and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci* 5.1: 17-60, 1960.

- [4.12] DJ Watts and SH Strogatz, "Collective dynamics of 'small-world' networks," *Nature* 393.6684: 440-442, 1998.
- [4.13] NetworkX, "watts\_strogatz\_graph". Available [https://networkx.org/documentation/stable/reference/generated/networkx.generators.random\\_graphs.watts\\_strogatz\\_graph.html](https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.watts_strogatz_graph.html)
- [4.14] NetworkX, "gnp\_random\_graph". Available [https://networkx.org/documentation/stable/reference/generated/networkx.generators.random\\_graphs.gnp\\_random\\_graph.html](https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.gnp_random_graph.html)
- [4.15] R. Dunbar, "Coevolution of neocortical size, group size and language in humans," *Behavioral and Brain sciences*, 16.4: 681-694, 1993.
- [4.16] B. Gonçalves, N. Perra, and A. Vespignani, "Modeling users' activity on Twitter networks: Validation of Dunbar's number," *PLOS ONE*, 6.8: e22656, 2011.
- [4.17] R. Dunbar, "How many friends does one person need?," *How Many Friends Does One Person Need?*, Harvard University Press, 2022.
- [4.18] D. Striga and V. Podobnik, "Benford's law and Dunbar's number: Does Facebook have a power to change natural and anthropological laws?," *IEEE Access* 6: 14629-14642, 2018.
- [4.19] D. Wedding, "Have We All Exceeded Dunbar's Number?," *APA Psycnet*, 2009. <https://psycnet.apa.org/doi/10.1037/e672152011-001>.
- [4.20] A. Hernando, D. Villuendas, C. Vesperinas, et al, "Unravelling the size distribution of social groups with information theory in complex networks," *Eur. Phys. J., B* 76, 87-97, 2010. <https://doi.org/10.1140/epjb/e2010-00216-1>
- [4.21] T. Chai and RR Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?-Arguments against avoiding RMSE in the literature," *Geoscientific model development*, 7.3: 1247-1250, 2014.
- [4.22] Python, "Time Module". Available: <https://docs.python.org/3/library/time.html>
- [4.23] B. Dadachev, A. Balinsky, H. Balinsky and S. Simske, "On the Helmholtz Principle for Data Mining," *2012 Third International Conference on Emerging Security Technologies*, 2012, pp. 99-102, doi: 10.1109/EST.2012.11
- [4.24] AL Barabási, *Network Science*. Cambridge: Cambridge University Press, 2016.

## **Chapter V**

- [5.1] Cybersecurity and Infrastructure Security Agency, "Mis, Dis, Malinformation", 2022. Available: <https://www.cisa.gov/mdm>

- [5.2] World Economic Forum, "Global Risks 2013 Eighth Edition," 2013. Available: <https://reports.weforum.org/global-risks-2013/risk-case-1/digital-wildfires-in-a-hyperconnected-world/>
- [5.3] G. Liu, Y. Wang, & M. Orgun, "Trust Transitivity in Complex Social Networks," *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2011.
- [5.4] S. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, 2003. Association for Computing Machinery, New York, NY, USA, 640–651. DOI:<https://doi.org/10.1145/775152.775242>.
- [5.5] U. Kuter and J. Golbeck. "SUNNY: A new algorithm for trust inference in social networks using probabilistic confidence models," *AAAI*, Vol. 7, 2007.
- [5.6] J. Ratkiewicz, M. Conover, M Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer, "Truthy: Mapping the spread of astroturf in microblog streams," *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011*, 249–252. <https://doi.org/10.1145/1963192.1963301>
- [5.7] S. Flaxman, S. Goel, and J.M. Rao, "Filter bubbles, echo chambers, and online news consumption," *Public Opinion Quarterly*, 80 (Specialissue1), pp. 298–320, 2016. <https://doi.org/10.1093/poq/nfw006>
- [5.8] M. del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H.E. Stanley, and W. Quattrociocchi, "The spreading of misinformation online," *Proceedings of the National Academy of Sciences of the United States of America*, 113(3), pp. 554–559, 2016. <https://doi.org/10.1073/pnas.1517441113>
- [5.9] C. Shao, P.M. Hui, L. Wang, X. Jiang, A. Flammini, F. Menczer, and G.L. Ciampaglia, "Anatomy of an online misinformation network," *PLoS ONE*, 13(4), 2018. <https://doi.org/10.1371/journal.pone.0196087>
- [5.10] X. Lou, A. Flammini, and F. Menczer, "Manipulating the Online Marketplace of Ideas," preprint from <http://arxiv.org/abs/1907.06130>, 2019.
- [5.11] C. Shao, G.L. Ciampaglia, A. Flammini, and F. Menczer, "Hoaxy: A Platform for Tracking Online Misinformation," *Proceedings of the 25th International Conference Companion on World Wide Web (WWW '16 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 745–750, 2016. <https://doi.org/10.1145/2872518.2890098>
- [5.12] M. Granik and V. Mesyura, "Fake news detection using naive Bayes classifier," *IEEE first Ukraine conference on electrical and computer engineering (UKRCON)* (pp. 900-903), 2017 IEEE.



- [5.13] F.A. Ozbay and B. Alatas, "Fake news detection within online social media using supervised artificial intelligence algorithms," *Physica A: Statistical Mechanics and its Applications*, 540, 123174, 2020.
- [5.14] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys (CSUR)*, 53(5), 1-40, 2020.
- [5.15] S. Agreste, P. de Meo, E. Ferrara, S. Piccolo, and A. Provetti, "Trust Networks: Topology, Dynamics, and Measurements," *IEEE Internet Computing*, 19(6), 26–35, 2015. <https://doi.org/10.1109/MIC.2015.93>
- [5.16] W. Yuan, D. Guan, Y.K. Lee, and S. Lee, "The small-world trust network," *Applied Intelligence*, 35(3), 399–410, 2011. <https://doi.org/10.1007/s10489-010-0230-7>
- [5.17] P. Li, W. Zhao, J. Yang, and J. Wu, "CoTrRank: Trust Ranking on Twitter," *IEEE Intelligent Systems*, 36(1), 35–45, 2021. <https://doi.org/10.1109/MIS.2020.3045001>
- [5.18] K. Pogorelov, D.T. Schroeder, P. Filkukov, S. Brenner, and J. Langguth, "WICO Text: A Labeled Dataset of Conspiracy Theory and 5G-Corona Misinformation Tweets," *Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks*, 21–25, 2021. Association for Computing Machinery, New York, NY, USA, DOI 10.1145/3472720.3483617,.
- [5.19] M.E.J. Newman, *Networks*. Oxford: Oxford University Press, 2010.
- [5.20] J.M. Kleinberg, "Method and system for identifying authoritative information resources in an environment with content-based links between information resources" US Patent 6112202A, August 29, 2000, <https://patents.google.com/patent/US6112202>.
- [5.21] M.E.J. Newman, "Mixing patterns in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics* 67 2 Pt 2, 2003.
- [5.22] R.S. Burt, "Structural Holes: The Structure of Competition", Cambridge, MA: Harvard University Press, 1992.
- [5.23] R. Albert and A.L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, Vol. 75, Issue 1, pp. 47-97, 2002. American Physical Society, DOI 10.1103/RevModPhys.74.47, <https://link.aps.org/doi/10.1103/RevModPhys.74.47>.
- [5.24] A.L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, Vol 286, Issue 5439, pp. 509, Oct. 1999. DOI 10.1126/science.286.5439.509.
- [5.25] D. Wang and Y. Qian. "Echo chamber effect in rumor rebuttal discussions about COVID-19 in China: social media content and network analysis study." *Journal of Medical Internet Research* 23.3 (2021): e27009.

## **Chapter VI**

- [6.1] Synopsys 2021 Open Source Security and Risk Analysis Report. [2021]. Web page. Retrieved from <https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html>
- [6.2] GitHub users listing, available <https://github.com/search?q=type:user&type=Users>
- [6.3] M. Osterloh and S. Rota, "Trust and community in open source software production," *Analyse & kritik* 26.1 (2004): 279-301.
- [6.4] C. Cowan. "Software security for open-source systems." *IEEE Security & Privacy* 1.1 (2003): 38-45] [cite Payne, Christian. "On the security of open source software." *Information systems journal* 12.1 (2002): 61-78.]
- [6.5] FS Gysin and A. Kuhn. "A trustability metric for code search based on developer karma." *Proceedings of 2010 icse workshop on search-driven development: Users, infrastructure, tools and evaluation*. 2010
- [6.6] H. Sapkota, PK Murukannaiah, and Y. Wang, "A network-centric approach for estimating trust between open source software developers," *PLOS One* 14.12: e0226281, 2019.
- [6.7] J. Fan, Y. Li, S. Wang and TN Nguyen. "A C/C++ Code Vulnerability Dataset with Code Changes and CVE Summaries". In *MSR '20: The 17th International Conference on Mining Software Repositories*, May 25–26, 2020, MSR, Seoul, South Korea. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3379597.3387501>. GitHub repository available [https://github.com/ZeoVan/MSR\\_20\\_Code\\_Vulnerability\\_CSV\\_Dataset](https://github.com/ZeoVan/MSR_20_Code_Vulnerability_CSV_Dataset)
- [6.8] OpenSSL Cryptography and SSL/TLS Toolkit. [n.d.]. Web page. Retrieved from <https://www.openssl.org/>
- [6.9] Let's Encrypt [n.d.]. Web page. Retrieved from <https://letsencrypt.org/>
- [6.10] Let's Encrypt Usage Statistics [n.d.]. Web page. Retrieved from <https://letsencrypt.org/stats/>
- [6.11] B. Schneier, "Let's Encrypt Vulnerability", *Schneier on Security* Web page. Retrieved from [https://www.schneier.com/blog/archives/2020/03/lets\\_encrypt\\_vu.html](https://www.schneier.com/blog/archives/2020/03/lets_encrypt_vu.html)
- [6.12] GH Archive [n.d.]. Web page. Retrieved from <https://www.gharchive.org/>
- [6.13] N. Parlante. "Linked list basics." Stanford University Computer Science Library, 2006. Retrieved from <http://117.211.166.170:8080/jspui/bitstream/123456789/1552/1/Linked.pdf>
- [6.14] R. Boyle P. Bonacich. "The Development of Trust and Mistrust in Mixed-Motive Games." *Sociometry* 33.2 (1970): 123–139. Web.

- [6.15] RM Kramer, "Trust and distrust in organizations: Emerging perspectives, enduring questions," *Annual Review of Psychology*, 50 (1999): 569.
- [6.16] AA Hagberg, DA Schult and PJ Swart, "Exploring network structure, dynamics, and function using NetworkX", in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008.
- [6.17] M.E.J. Newman, *Networks*. Oxford: Oxford University Press, 2010.
- [6.18] A. Clauset, ME Newman, and C. Moore, "Finding community structure in very large networks." *Physical Review E* 70(6), 2004.
- [6.19] W. Yuan, D. Guan, Y.K. Lee, and S. Lee, "The small-world trust network," *Applied Intelligence*, 35(3), 399–410, 2011. <https://doi.org/10.1007/s10489-010-0230-7>
- [6.20] S. Agreste, P. de Meo, E. Ferrara, S. Piccolo, and A. Provetti, "Trust Networks: Topology, Dynamics, and Measurements," *IEEE Internet Computing*, 19(6), 26–35, 2015. <https://doi.org/10.1109/MIC.2015.93>
- [6.21] A. Clauset, Aaron, CR Shalizi, and MEJ Newman. "Power-law distributions in empirical data." *SIAM Review* 51.4 (2009): 661-703.
- [6.22] J.M. Kleinberg, "Method and system for identifying authoritative information resources in an environment with content-based links between information resources" US Patent 6112202A, August 29, 2000, <https://patents.google.com/patent/US6112202>.
- [6.23] H. Balinsky, A. Balinsky and S. Simske. "Document sentences as a small world", presented at *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Anchorage, AK, USA, 9-12 October 2011. Published in: Tunstel, E. and Nahavandi, S. eds. *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Los Alamitos, CA: IEEE, pp. 2583-2588. 10.1109/ICSMC.2011.6084065

## **Chapter VII**

- [7.1] "Why isn't my link karma accurate?", 2014. Web page. Retrieved from <https://www.Reddit.com/r/NoStupidQuestions/comments/2flcgr/comment/ckace35/>
- [7.2] T. Squirrell, Tim. "Platform dialectics: The relationships between volunteer moderators and end users on Reddit." *New Media & Society* 21.9 (2019): 1910-1927
- [7.3] C. Buntain and J. Golbeck. "Identifying social roles in Reddit using network structure." *Proceedings of the 23rd International Conference on World Wide Web*. 2014.
- [7.4] PT Metaxas and J. DeStefano. "Web spam, propaganda and trust." *International Conference on World Wide Web*, 2005.
- [7.5] M. Potter. "Bad actors never sleep: content manipulation on Reddit." *Continuum* 35.5 (2021): 706-718
- [7.6] M. Carman et al. "Manipulating visibility of political and apolitical threads on Reddit via score boosting." *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018
- [7.7] Kossiakoff, Sweet, Seymour, and Biemer. *Systems Engineering--Principles and Practice*. 2d ed. Wiley and Sons. Hoboken, N.J., 2011
- [7.8] E. Hargittai and G. Walejko. "THE PARTICIPATION DIVIDE: Content Creation and Sharing in the Digital Age1." *Information, Communication & Society* 11.2 (2008): 239-256. Web.
- [7.9] AutoModerator Documentation. [n.d.]. Web page. Retrieved from <https://www.Reddit.com/r/AutoModerator/wiki/index/>
- [7.10] Reddit moderation tools documentation. "What is crowd control?" [n.d.]. Retrieved from <https://mods.reddithelp.com/hc/en-us/articles/360038129231-What-is-Crowd-Control->
- [7.11] Reddit user u/Clippy\_Office\_Asst, "Introducing /u/Clippy\_Office\_Asst (aka Clippy)", 2020. Web page. Retrieved from [https://www.Reddit.com/r/Clippy\\_Office\\_Asst/comments/jlme7b/introducing\\_uclippy\\_office\\_asst\\_aka\\_clippy/](https://www.Reddit.com/r/Clippy_Office_Asst/comments/jlme7b/introducing_uclippy_office_asst_aka_clippy/)
- [7.12] Reddit user BotDefense. [n.d.]. Web page. Retrieved from <https://www.Reddit.com/user/botdefense>
- [7.13] Reddit user AssistantBOT. [n.d.]. Web page. Retrieved from <https://www.Reddit.com/user/assistantbot>
- [7.14] Reddit user repostsentinel. [n.d.]. Web page. Retrieved from <https://www.Reddit.com/user/repostsentinel>

- [7.15] Reddit user u/toxicitymodbot, "Updated bot backed by moderation-oriented ML for automatically reporting + removing hate speech, personal attacks, insults", 2022. Web page. Retrieved from [https://www.Reddit.com/r/redditdev/comments/xdscbo/updated\\_bot\\_backed\\_by\\_moderation\\_oriented\\_ml\\_for/](https://www.Reddit.com/r/redditdev/comments/xdscbo/updated_bot_backed_by_moderation_oriented_ml_for/)
- [7.16] Reddit user u/\_pacjax\_, "I made a bot that calculates user scores & ranks on a sub", 2022. Web page. Retrieved from [https://www.Reddit.com/r/Bot/comments/w4nc3p/i\\_made\\_a\\_bot\\_that\\_calculates\\_user\\_scores\\_ranks\\_on/](https://www.Reddit.com/r/Bot/comments/w4nc3p/i_made_a_bot_that_calculates_user_scores_ranks_on/)
- [7.17] Toolbox for Reddit. [n.d.]. Web page. Retrieved from <https://www.Reddit.com/r/toolbox/>
- [7.18] B. Boe, "Working with PRAW's Models » Comment", 2022, *Documentation for PRAW: The Python Reddit API Wrapper*. Web page. Retrieved from [https://praw.readthedocs.io/en/latest/code\\_overview/models/comment.html](https://praw.readthedocs.io/en/latest/code_overview/models/comment.html)
- [7.19] B. Boe, "Working with PRAW's Models » Submission", 2022, *Documentation for PRAW: The Python Reddit API Wrapper*. Web page. Retrieved from [https://praw.readthedocs.io/en/latest/code\\_overview/models/submission.html](https://praw.readthedocs.io/en/latest/code_overview/models/submission.html)
- [7.20] B. Boe, *Documentation for PRAW: The Python Reddit API Wrapper*. Web page. Retrieved from <https://praw.readthedocs.io/>
- [7.21] H. Gupta (Kaggle user hritik7080), "Reddit's 2400 Posts Dataset", 2020. Web page. Retrieved from [cite https://www.kaggle.com/datasets/hritik7080/Reddit-flair-dataset](https://www.kaggle.com/datasets/hritik7080/Reddit-flair-dataset)
- [7.22] Kaggle user Chris (rootuser), "Worldnews on Reddit from 2008 to Today", 2016. Web page. Retrieved from <https://www.kaggle.com/rootuser/worldnews-on-Reddit>
- [7.23] Google Developers, "Fact Check Tools API". [n.d.]. Web page. Retrieved from <https://developers.google.com/fact-check/tools/api/>