Graduate School ETD Form 9
(Revised 01/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By NEWELL, ANDREW, JOHN

Entitled
ACHIEVING RESILIENT NETWORKS WITH DIVERSITY AND NETWORK CODING

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Cristina Nita-Rotaru

Sonia Fahmy / Sandra Freeman

Dongyan Xu

Susanne Hambrusch

To the best of my knowledge and as understood by the student in the *Thesis/Dissertation Agreement. Publication Delay, and Certification/Disclaimer (Graduate School Form 32)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Cristina Nita-Rotaru

Approved by Major Professor(s): _____

Approved by: Sunil Prabhakar / William J Gorman 07/08/2014

Head of the Department Graduate Program Date

ACHIEVING RESILIENT NETWORKS WITH DIVERSITY AND NETWORK CODING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Andrew Newell

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2014

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I would not have pursued a PhD if it were not for the excellent mentors in my undergraduate program at Southern Illinois University at Carbondale. Kenny Fong, Prof. Thomas Porter, and Prof. Kemal Akkaya are a few which gave me special attention and encourged my academic pursuits. I would especially like to point out that Prof. Kemal Akkaya included me on numerous research projects despite me being an undergraduate student. Every project he included me on turned into a successful publication which was great experience and start of my publication record.

For my successes at Purdue University, I have to give my biggest thanks to my adviser, Prof. Cristina Nita-Rotaru. At the start of my PhD candidacy she had the patience to help me with skills such as writing and presenting in which I had little prior experience. Throughout my career at Purdue University, she constantly pushed me to the point where I could lead successful research projects. She has been a great collaborator on research always providing insightful critiques which help form a successful paper. She both encouraged and helped me find four different internship opportunities helping me figure out the best future career path for myself. The internship with Facebook has led to a full-time offer which I plan to take after graduating. Finally, from my first day she was able to provide research funding for me on projects related to my ultimate dissertation topic which not all professors are willing or are capable of doing.

I would like to thank numerous collaborators throughout my career at Purdue University. First, Prof. Reza Curtmola was a vital part of the work on entropy attacks in network coding systems which is part of this dissertation. Second, the group at Johns Hopkins University has helped me tremendously with the work on network diversity. This group includes Prof. Yair Amir, Prof. Vladimir Braverman, Daniel Obenshain, and Thomas Tantillo. Furthermore, they are continuing along this

line of work which is necessary to make network diversity a widely used technique. Finally, Dr. Rahul Potharaju whom I have worked with on numerous non-dissertation related topics Purdue University. I always worked with him despite the topics being unrelated to my dissertation because he is a talented researcher, and I always enjoyed working with him.

On a personal note, I would like to thank my family Stephen Newell (father), Leah Newell (mother), Dr. Elizabeth Newell (sister), and all of the extended family for their support. I would also like to thank close friends Justin Chen, Ian Wilson, and Jacob Gleason who have consistently been there for me since high school.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

ABBREVIATIONS xi

ABSTRACT xiii

1 Introduction 1
1.1 Our Contributions 2
1.2 Organization 5

2 Increased Network Resilience by Assigning Diverse Variants to Routing Nodes 6
2.1 Network Diversity Model 10
2.1.1 Network Model 10
2.1.2 Adversarial Model 11
2.2 Diversity Assignment 12
2.2.1 Diversity Assignment Problem (DAP) 12
2.2.2 MIP Approach to DAP 18
2.2.3 DAP on the Case Study Topology 23
2.2.4 Near-optimal Assignments on Case Study Topology 27
2.3 Scaling Diversity Assignment 29
2.3.1 Approximate DAP (A-DAP) 29
2.3.2 Greedy Approach to A-DAP 29
2.3.3 A-DAP on the Case Study Topology 30
2.3.4 A-DAP on Random Topologies 32
2.4 Diversity Assignment for Dynamic Topologies 34
2.4.1 Online DAP (O-DAP) 35
2.4.2 MIP Approach to O-DAP 36
2.4.3 O-DAP on the Case Study Topology 37
2.5 Diversity Assignment for Specific Applications 39
2.5.1 Connected Component DAP (CC-DAP) 40
2.5.2 MIP Approach to CC-DAP 41
2.5.3 CC-DAP on Example Ring Topology 43
2.5.4 CC-DAP for Paxos on the Case Study Topology 45
2.5.5 CC-DAP for BFT on the Case Study Topology 47
2.6 Optimizing Client Traffic Patterns 47
2.6.1 Weighted DAP (W-DAP) 48

Page
2.6.2 MIP Approach to W-DAP . . . 49
2.6.3 W-DAP on the Case Study Topology . . . 50
2.7 Errors in Variant Compromise Estimation . . . 52
2.7.1 Methodology to Investigate Erroneous Information . . . 52
2.7.2 Error Analysis on Random Topologies . . . 54
2.8 Summary . . . 56

3 Null Space Based Defense for Pollution Attacks in Wireless Network Coding 57
3.1 Pollution Attack Model . . . 61
3.1.1 Network Coding System . . . 61
3.1.2 Adversarial Model . . . 62
3.2 Split Null Keys (SNK) . . . 63
3.2.1 Null Space Properties . . . 63
3.2.2 SNK Overview . . . 65
3.2.3 Null Keys Splitting Procedure . . . 67
3.2.4 Null Key Distribution and Verification . . . 69
3.3 Security Analysis . . . 70
3.4 Evaluation . . . 75
3.4.1 Simulation Methodology . . . 76
3.4.2 Performance Evaluation . . . 77
3.4.3 Scalability with Multiple Adversaries . . . 78
3.4.4 SNK vs. Traditional Secure Routing . . . 80
3.4.5 Overhead Results . . . 81
3.5 Summary . . . 83

4 Entropy Attacks and Countermeasures in Wireless Network Coding . . . 84
4.1 Entropy Attack Model . . . 88
4.1.1 Local Entropy Attacks . . . 89
4.1.2 Global Entropy Attacks . . . 89
4.2 Simulation Methodology . . . 90
4.3 Local Entropy Attacks . . . 91
4.3.1 Attack Impact . . . 91
4.3.2 Defense . . . 93
4.4 Global Entropy Attacks . . . 95
4.4.1 Attack Impact . . . 95
4.4.2 Global Entropy Defenses Overview . . . 97
4.4.3 Upstream Buffer Propagation . . . 98
4.4.4 Buffer Monitoring . . . 103
4.5 Security Analysis . . . 107
4.5.1 Attack Strength of UBP . . . 108
4.5.2 Watchdog Selection Constraints of BM . . . 110
4.6 Entropy Defense Overhead Analysis . . . 112
4.6.1 Computation Overhead . . . 112

Page
4.6.2 Communication Overhead 112
4.7 Summary 115
5 Related Work 118
5.1 Network Diversity 118
5.2 Byzantine Resilient Network Coding 120
6 Conclusion and Future Work 124
LIST OF REFERENCES 126
VITA 133

LIST OF TABLES

Table		Page
2.1	Notation for network diversity	20
2.2	Network characteristics for case study topology.	25
2.3	Values of three metrics for ring topology for three assignments that each maximize their own metric	45
2.4	Probability distributions for different α values.	54
3.1	Notation for network coding systems and the SNK protocol	61
4.1	Throughput results of various attack and defense scenarios with the topology in Figure 4.4	96
4.2	Notation for BCP and BM protocols	107

LIST OF FIGURES

Figure Page

2.1 Diversity assignment with 0.838 expected client connectivity on example topology. 14

2.2 Diversity assignment with 0.957 expected client connectivity on example topology. 14

2.3 DAP construction to solve the 3-SAT problem, $(\beta_1 + \bar{\beta}_2 + \beta_4)(\bar{\beta}_1 + \beta_7 + \beta_s)...(\bar{\beta}_8 + \beta_9 + \beta_s)$. Note that the y nodes are not included in this diagram. 16

2.4 Optimal assignment of one variant on case study topology achieving 0.9 expected client connectivity. 24

2.5 Optimal assignment of two variants on case study topology achieving 0.985 expected client connectivity. 24

2.6 Optimal assignment on case study topology of three variants achieving 0.997 expected client connectivity. 24

2.7 Histogram of expected client connectivity of 100,000 random assignments on case study topology. 24

2.8 Random assignment on case study topology achieving 0.881 expected client connectivity 28

2.9 Greedy assignment on case study topology achieving 0.992 expected client connectivity 28

2.10 Number of solutions within a given disconnectivity factor bound of optimal assignment on case study topology. 28

2.11 Expected client connectivity of random, optimal, and greedy assignments on random topologies for 25 nodes and varied density. 34

2.12 Time of optimal and greedy assignments on random topologies for 25 nodes and varied density. 34

2.13 Expected client connectivity of random, optimal, and greedy assignments on random topologies for 6 density and varied number of nodes. 35

2.14 Time of optimal and greedy assignments on random topologies for 6 density and varied number of nodes. 35

Figure Page

2.15 Expected client connectivity achieved by reconfigure and online assignments as case study topology grows. 38

2.16 Proportion of reconfigure's expected client connectivity achieved by online assignments as case study topology grows. 38

2.17 Assignment from optimizing probability of Paxos progress on ring topology. 44

2.18 Assignment from optimizing probability of BFT progress on ring topology. 44

2.19 Assignment from optimizing expected client connectivity on ring topology. 44

2.20 Four variant CC-DAP assignment for connected components of size 8 for BFT. 44

2.21 Three variant CC-DAP assignment for connected components of size 6 for Paxos. 46

2.22 Four variant DAP assignment. 46

2.23 Assignment with W-DAP with dominant weights between gray-filled client-pair on case study topology. 51

2.24 Depiction of the difference between independent events on the left ($\alpha = 0$) and full depedence on the right ($\alpha = 1$). 53

2.25 Error values with a discrepency between real and available information in Δ_2. 55

2.26 Error values with a discrepency between real and available information in α. 55

3.1 Throughput of SNK, DART, KFM, and MORE. 77

3.2 Latency of SNK, DART, KFM, and MORE. 77

3.3 Communication overhead of SNK, HOMOMAC-2, DART, and EDART. 78

3.4 Throughput of SNK and EDART with 0 attackers. 79

3.5 Throughput of SNK and EDART with 5 attackers. 79

3.6 Throughput of SNK and EDART with 10 attackers. 79

3.7 Throughput of SNK which provides defense against any number of adversaries and HOMOMAC-x that is configured to provide defense against x adveraries. 79

3.8 Throughput of SNK, MORE, and ARAN. 81

3.9 Latency of SNK, MORE, and ARAN. 81

Figure Page

3.10 Time for forwarder computation of verifying a packet for SNK, DART/EDART, and HOMOMAC-2. 83

3.11 Time for source computation of generating null keys, checksums, and homomorphic MACs for SNK, DART/EDART, and HOMOMAC-2 for a generation. 83

4.1 Throughput of MORE with varying entropy attackers. 92

4.2 Proportion non-innovative coded packet receptions of MORE for varying entropy attackers. 92

4.3 Throughput of MORE, MORE-NLA (entropy defense), and the ideal entropy defense. 94

4.4 Topology for global entropy attack scenario. 95

4.5 Topology for monitoring defense scenario. 105

4.6 Timeline for source, attacker, and victim for the UBP protocol. 108

4.7 Maximum valid assignment of n watchdogs per forwarder. 111

4.8 Given flows in the Roofnet topology, we show the communication overhead of reliable multicasts in BM varying the number of watchdogs per node to 1, 2, and 3. 114

ABBREVIATIONS

A-DAP	Approximate Diversity Assignment Problem
AODV	Ad hoc On demand Distance Vector Routing
ARAN	Secure, wireless best path routing protocol
BCP	Buffer Checksum Packet
BFT	Byzantine Fault Tolerance, a protocol for state machine replication
BM	Proposed Buffer Monitoring defense against global entropy attacks
CC-DAP	Connected Component Diversity Assignment Problem
CDF	Cumulative Distribution Function
CDP	Coded Data Packet
CHP	Coding Header Packet
DAP	Diversity Assignment Problem
DART	Homomorphic checksum defense against pollution attacks
DDP	Dropped Data Packet
EDART	Adaptive version of DART defense
HOMOMAC	Homomorphic MAC defense against pollution attacks
KFM	Homomorphic signature defense against pollution attacks
MIP	Mixed Integer Program
MORE	Random linear network coding protocol
NLA	Proposed Non-innovate Link Adjustment defense against local entropy attacks
O-DAP	Online Diversity Assignment Problem
PRF	Pseudo Random Function

SNK	Proposed Split Null Keys defense against pollution attacks
SNT	Sequence Number Table
UAT	Upstream Accusation Table
UBP	Proposed Upstream Buffer Propagation defense against global entropy attacks
W-DAP	Weighted Diversity Assignment Problem
WBT	Watchdog Buffer Table

ABSTRACT

Newell, Andrew Ph.D., Purdue University, August 2014. Achieving Resilient Networks with Diversity and Network Coding. Major Professor: Cristina Nita-Rotaru.

This dissertation provides strong resilience techniques applying to general networks. Examples of important networks are private wired networks connecting large datacenters, overlay topology networks delivering live television broadcasts, and wireless mesh networks rapidly set up after a disaster to replace existing damaged infrastructure. Given the trends of increased reliance on networks and capabilities of attackers, network security is vital to national security. Network attacks can be characterized along the two dimensions of access and motivation. Attacker access can be either as an insider or outsider. An insider has more capabilities of having full control of some routing nodes. Attacker motivation can be targeting confidentiality, data integrity, or availability where availability is the only one that cannot typically be dealt by the known cryptographic techniques of encryption, digital signatures, or message authentication codes. In this dissertation, we focus on insider attackers that attack the availability of the network.

Our first step towards resilience is to ensure that an attacker cannot compromise nodes that partition the network since such an attack trivially succeeds in preventing availability. Such large compromises are likely in today's typical network deployment where all routers have identical components and a single successful exploit can be repeatedly used against all routers in the network. In our work, we demonstrate how diversity alleviates such problems when assigning diversity optimally to routers in the network. Routers that are diverse enough to not permit common exploits must have different components such as hardware, operating systems, routing code, and even administrators. These types of diversity are limited, so our assignment of diversity to

routers typically has very few variants which must be assigned to a large number of routing nodes. We provide a comprehensive study of diversity assignment in networks by proposing problems for various network goals, techniques to solve these problems optimally or at scale, and demonstrated benefits of applying such analysis to real topologies.

Diversity ensures that a network remains well-connected by honest nodes even after sophisticated compromise attempts. However, an attacker can still succeed in attacking availability by attacking the routing protocol. We provide techniques resistant to insider attacks when using network coding. Network coding offers higher performance in a network by performing encoding techniques on packets. Insiders can attack the encoding technique in two ways by either forcing incorrect decoding or delaying decoding. For pollution attacks, forcing incorrect decoding, our work proposes a new defense against pollution attacks overcoming limitations of prior work which includes expensive security computation at routers, communication overhead that scales with the number of insiders, and delayed verification. For entropy attacks, delaying decoding, to the best of our knowledge our work is the first to demonstrate the effectiveness of such attacks along with considering defenses for sophisticated entropy attackers which collude.

1 INTRODUCTION

We improve resilience to malicious insider routers (i.e., byzantine resilience) along two dimensions. Firstly, we study how to best utilize diversity within a network to ensure that compromises which target a particular attack surface cannot cripple an entire network by repeating the attack against numerous homogeneous nodes. Network diversity is the use of different components such that they are vulnerable in different ways. Secondly, we consider strengthening the security of network coding protocols to ensure they are practical in a network with untrusted routers. These two dimensions are orthogonal as there are situations where only diversity may be applied or strengthened network coding protocols could be applied. They both work towards a common goal of network resilience and work well in conjunction to significantly boost the resilience of a network. Here we provide background on both of these areas.

Byzantine Resilient Network Diversity. As a first major step towards byzantine resilience, we want to ensure a surviving network remains after sophisticated router compromise attempts by attackers. A router in the network has many attack surfaces such as the routing code, the operating system of the router, the administration, or the cloud service provider providing the router. If an attack is found against one of these attack surfaces, and the network consists of homogeneous routers, then the entire network can be compromised. In this case, any resilience gained by a byzantine tolerant protocol are meaningless despite the use of a sophisticated byzantine tolerant protocol.

For these types of diversity, its common to have only a few diverse variants available for use. Networks have numerous routing nodes which need a variant, so there is a problem of how to assign this diversity. By spreading it evenly, you are ensuring a portion of the network survives an attack. This is not sufficient for good connectivity, since the remaining structure of the network may not be well connected. So,

we study how to assign diversity such that the network remains well connected even after portions of the network are compromised due to vulnerable variants.

Byzantine Resilient Protocols with Network Coding. In scenarios of attack where routers are compromised, our diversity techniques ensure a strong surviving set of nodes is not compromised. For a holistic defense, we must also be able to perform routing over a set of honest nodes despite malicious nodes actively attempting to disrupt routing. As opposed to byzantine resilience with traditional best path routing, we focus on byzantine resilience with network coding for two main reasons. First, through encoding data at the routers, network coding achieves higher performance than simple best path routing by leveraging the coding capabilities of intermediate routers. Second, network coding has inherent resilience advantages compared with typical best path routing as data can be delivered along multiple paths, so if one path is blocked by an adversary then the remaining paths still deliver data without interruption.

In network coding systems, intermediate routers are supposed to alter packets to perform their coding which makes various security tasks much more difficult. An attacker can subvert a coding system by crafting packets to ensure ensure the final decoding results are corrupted data (pollution attack) or the crafted packets provide no information for decoding (entropy attack). More research is needed to defend against these attacks without imposing significant overhead.

1.1 Our Contributions

This dissertation advances the state-of-the-art in byzantine tolerant networks in both aspects of network diversity and byzantine resilient network coding.

Existing work offers numerous techniques to diversity systems [1–4] by creating different versions that fail in different ways. However, no existing work addresses how to spread such diversity across a network ensuring important networking goals such as

connectivity. We call this spreading of diversity an *assignment* of diversity to routing nodes. We motivate and investigate this new area with the following contributions:

- Define specific diversity assignment optimization problems matching different networking needs.
- Show how to express these problems as Mixed Integer Programs to solve these problems optimally. For large-scale problems, we also provide heuristics to solve at scale achieving close to an optimal assignment.
- Demonstrated practical benefits on a real global cloud network topology.

Existing work has demonstrated performance benefits of disseminating data with network coding [5–14] which is why we envision such protocols for the network demands of critical services. We aim to leverage these performance benefits while addressing key security concerns of using network coding. Namely, the difficulties in preventing intermediate routers from modifying packets to result in incorrect decoding (pollution attacks) or useless information (entropy attacks). Preventing modification is straight-forward in traditional networks without network coding as classical cryptographic integrity measures can be applied ensuring untrusted routers do not alter data in packets. However, in networks with network coding, the untrusted routers must alter packet data to perform coding operations.

We advance defenses against those pollution attacks that aim to ensure decoding fails which have been well-studied problem in network coding [15–26]. Many of these works provide valid defenses, but they impose such great overhead in terms of communication and computation that network performance is worse than a more simplistic best path routing protocol that enforces digital signatures. As the aim of utilizing network coding is increased performance, it is our goal to maintain that performance improvement over traditional best path routing protocols even in secure settings. Thus, our focus is to ensure a practical overhead while still ensuring strong security guarantees. We meet this goal by creating a defense based on properties of null spaces. We contribute a pollution defense with the following contributions:

- We overcome limitations of existing work by using cheap dot product computations for verifying packets, fixed overhead independent of the number of insiders, and do not enforce any delays on forwarders to verify packets.

- We provide detailed security analysis to prove our scheme is resilient to any number of insiders that can overhear any traffic sent through the network.

- Through simulation experiments, we are able to demonstrate the performance benefits of our scheme compared to existing work.

We also provide needed research for the other important attack against network coding, entropy attacks. These do not cause decoding to fail instead just delay decoding by crafting packets with useless information. To our knowledge one work mentions such an attack [27] while another work aims to lessen the computational overhead during such an attack [28]. Both works do not study the potential impact of an attack on the overall network performance nor explore the full potential of attacks. We explore this area from both an attack and defense perspective:

- Through simulations we show the severity of entropy attacks on the performance of a network coding system.

- We demonstrate how an attacker can create packets that look useful locally, but somewhere downstream they are actually useless to a network. We call this a global entropy attack, and it subverts most straight-forward defenses that only use local information.

- We propose two defenses with different constraints and needs that are capable of mitigating the stealthier global entropy attacks. We analyze the effectiveness and overhead these two defenses which shows the important trade-offs for choosing an appropriate defense for a network.

1.2 Organization

We organize this dissertation as follows. Chapter 2 contains our work on network diversity which first appeared in [29]. Chapter 3 contains our proposed pollution defense which first appeared in [30]. Chapter 4 contains our work on entropy attacks and defenses which first appeared in [31]. We cover all related work to this dissertation in Chapter 5. We conclude the dissertation in Chapter 6.

2 INCREASED NETWORK RESILIENCE BY ASSIGNING DIVERSE VARIANTS TO ROUTING NODES

Networks with homogeneous routing nodes are constantly at risk as any vulnerability found against a single routing node could be used to compromise all nodes. Diversity can be employed at various levels on the routing nodes to address this problem by improving resiliency against different classes of attacks. In this chapter, we base resiliency on the number of surviving client-to-client connections offered by the network when under attack. Diversifying the operating system provides protection against common types of attacks that target operating system vulnerabilities [1]; utilizing multi-variant programming protects against programming vulnerabilities or logical programming errors [2, 3]; using different administrative personnel mitigates social engineering or insider attacks [4] However, there are only a limited number of operating systems, software versions, and personnel to utilize as diverse variants. So then, how does one assign these limited number of diverse variants to the routing nodes in the network to achieve optimal resiliency?

Initially, we assumed that a random assignment of a few diverse variants would perform well. However, we found that a random assignment performs rather poorly, in many cases providing less resiliency than using the best single variant at all routing nodes, and occasionally even less resiliency than using the worst single variant at all routing nodes. Clearly, a better approach is necessary to realize the benefits of diversity.

Our interest in the assignment of few diverse variants to networks arose from constructing a cloud service over a global network of data centers [32]. We needed to have an intrusion-tolerant infrastructure in order to monitor and control the cloud even in the case of sophisticated intrusions. While designing intrusion-tolerant protocols for messaging and maintaining consistent state, we realized that without diversity all

the nodes could be compromised by a single vulnerability. Inspired by prior work on diversity [1], we were especially interested in diversifying the operating system (e.g., Linux, MacOS, and FreeBSD). The additional overhead of managing multiple operating systems within the cloud infrastructure led us to consider only a small number of variants to create diversity.

In this chapter, we demonstrate that the way diverse variants are assigned across the network (i.e., which variant is assigned to which routing node) is of utmost importance to the overall network resiliency when the number of variants is smaller than the number of routing nodes in the network. To our knowledge, our work on diversity is the first to study the impact of variant assignment to routing nodes on overall network resiliency.

We define a new problem, the Diversity Assignment Problem (DAP), which specifies how to optimize overall network resiliency when placing diverse variants that are compromised independently at routing nodes, and we present novel solutions to solve this problem. While DAP is NP-Hard, we show that it is feasible to solve it optimally on a variety of medium-size random network graphs. We also show an efficient algorithm that approximates DAP well for larger graphs, incurring a relatively small resiliency cost compared with the optimal solution.

To check the applicability of our approach in a real-world setting, we obtained a network graph representative of the global overlay topology used by LTN. Even though this topology was constructed with high availability as the goal (rather than intrusion-tolerance), the optimal variant assignment solution to the DAP ensures a system resiliency that is significantly higher than the resiliency achieved by any of the individual variants.

In real-life settings, routing nodes may be added from time to time to meet increasing system demands. Calculating an optimal solution for the extended network is certainly feasible. However, that solution is likely to require variant re-assignment for many of the existing routing nodes, which may not be feasible in a 24/7 service as downtime for re-configuring nodes is unacceptable. We present an online version of

DAP that finds the optimal incremental assignment. When applied to the mentioned global topology, we discover that an important trade-off exists between the resiliency the system achieves and how often the network changes.

We initially choose an application agnostic metric for network resiliency that captures the expected client-to-client connectivity between all pairs. We investigate the advantages of considering the specific resiliency needs defined by the nature of a distributed application running at the clients. Specifically, we show how to find the optimal assignment for the underlying network supporting either the Paxos [33] or Byzantine Fault-Tolerant (BFT) [34] protocols. When applied to the mentioned global topology, we found that an assignment that is tailored to these application requirements can provide higher resiliency than an assignment that focuses on general network resiliency obtained by maximizing the expected client-to-client connectivity. Furthermore, we show how to optimize for a weighted resiliency metric for use as a resiliency metric. Such an optimization offers the ability to designate specific client pairs with higher importance such that the assignment prioritizes these client pairs by offering them higher resiliency.

Our assignments are based on assumptions of correct information about compromise probabilities, so we finally investigate the effect of performing an assignment based on incorrect information. We observe two bad effects of using incorrect information. First, an assignment based on incorrect information will have worse resiliency than an assignment based on correct information. Second, a network operator's confidence in the overall resiliency of a network is incorrect if information about compromises is incorrect.

When studying DAP, we learned three key points that are relevant to any application of limited diversity that aims to increase network resiliency:

1. A high level of overall network resiliency can be obtained even from variants that are weak on their own. Despite the variants being compromised (independently) with a relatively high probability, they are compromised in different

ways. Carefully assigning variants to routing nodes allows surviving subsets of the network to still remain highly connected.

2. The simplest and seemingly practical approach of just assigning variants randomly offers very low resiliency compared to the optimal assignment. Additionally, in many random placements we found that the resiliency of the network is actually worse than if no diversity assignment were used at all. To provide further understanding we investigated the proportion of search space that has solutions near the optimal, finding that a very small fraction of the search space has resilience values within a few factors of the optimal solution.

3. While optimizing expected client-to-client connectivity provides a good measure for the resiliency of the network to intrusions, considering application-specific connectivity requirements may lead to a different assignment that maximizes overall system resiliency (as opposed to network resiliency) for that application on that network.

The contributions described in this chapter are as follows:

- We introduce DAP and formulate it using Mixed Integer Programming (MIP) [35] and find the optimal solution on random graphs constructed in a manner reminiscent of real overlay topologies. To support larger graphs, we extend this formulation to a fast greedy approximation and demonstrate results that are relatively close to the optimal solution in such larger graphs.

- We analyze the impact of diversity on a real cloud overlay topology and extend our approach to support adding routing nodes to the graph in an online manner to address increased client demand.

- We extend our approach to optimize network resiliency for a given application's demands, rather than for overall expected client-to-client connectivity, to maximize system resiliency.

- We extend our approach to prioritize optimization for specific client-to-client connections such that the assignment offers higher resiliency to certain connections while sacrificing resiliency of other connections.

- We analyze the loss in resiliency when optimally assigning variants based on incorrect information about compromises.

The rest of the chapter is organized as follows. Section 2.1 describes our network and adversarial models. Section 2.2 presents the general DAP along with an optimal solution. Section 2.3 describes and evaluates a greedy approximation algorithm to solve DAP in larger topologies. Section 2.4 shows how resiliency is affected in dynamic topology scenarios. Section 2.5 shows the increased advantage of performing diversity assignment with client application knowledge. Section 2.6 shows how to convert DAP to prioritize specific client pairs. Section 2.7 analyzes how incorrect compromise information affects assignment. Section 2.8 summarizes this chapter.

2.1 Network Diversity Model

We describe the model of the network and attacker which we consider in this work. These models are quite general as our approaches can be applied in a variety networking contexts with various of diversity techniques. Our motivation started with a scenario of cloud services being provided over a global network of datacenters while diversifying operating systems for improved resilience, but we noticed that the core problem is general to any network.

2.1.1 Network Model

We assume a network topology of routing *nodes* that provide communication to *clients*. We assume no control over the structure of the network topology as this is fixed based on the constraints of the networking context. In an overlay routing context, network links impose overhead to continuously monitor their latency and

loss characteristics, thus the degree at each node must be limited while ensuring the entire network is still well connected. Alternatively, in a wireless context, network links are limited by the physical broadcast range of each node. We assume that we have a set of diverse variants and can configure each routing node with a single variant. Our network goals are to maximize the number of client connections or an application-specific communication requirement of the clients.

2.1.2 Adversarial Model

We assume that there is no way to configure a routing node that meets our network needs while being completely invulnerable to attacker attempts of compromise. Thus, we adopt a probabilistic adversarial model that captures the following important resilience property of diversity we wish to leverage: even though we do not have access to a variant that cannot be compromised, we do have access to variants that are compromised in different ways. We assign a probability that an attacker is able to both find a vulnerability and create a successful exploit against a variant within a given time period, and then any routing node in the network with this variant will become compromised. As our probabilities are with respect to a certain time frame, a full long-term system needs mechanisms to detect and recover compromised variants, and we consider such mechanisms as outside the scope of this current work. Our probabilistic model of compromise offers a useful way to reason about an attacker's capabilities and measure a network's resilience. Even in realistic scenarios where an attacker is not modeled well probabilistically, we are still raising the bar for the attacker to ensure the attacker must find vulnerabilities and create exploits for different variants of routing nodes.

We do assume a byzantine tolerant routing protocol is used for routing to ensure that communication can occur between two clients as long as an honest path of routing nodes exists.

2.2 Diversity Assignment

In this section we present the Diversity Assignment Problem (DAP). DAP describes how to assign diversity to routing nodes in order to maximize the probability of each client pair being connected. We then describe existing Mixed Integer Programming (MIP) techniques and how these can be used to solve DAP. Lastly, we show the effectiveness of this technique on a realistic case study topology when compared with randomly assigning diversity.

2.2.1 Diversity Assignment Problem (DAP)

We consider a network consisting of a set of nodes N and a set of clients M. We use the terms *node* and *routing node* interchangeably throughout this chapter as they both correspond to routers in a network. A set of connections are defined among nodes and clients, so we can represent a network as a graph such as the one in Figure 2.1 and Figure 2.2 where clients are squares and nodes are circles in the graph. We assume no connections between clients which is the typical case in network topologies. However, such connections would pose no serious issue to our model or the solutions we present later. Each node is assigned a variant from the set of variants V, so there are $|V|^{|N|}$ possible assignments. We denote an assignment of one variant for each node as A. Note that $|V| < |N|$. Each variant $v_k \in V$ is associated with a compromise event e_k in the set of all compromise events E, so $|E| = |V|$. The probability of e_k occurring is $P(e_k)$. These events of compromise are independent,[1] so for any two compromise events $e_{k'}$ and $e_{k''}$ the following holds $P(e_{k'} \cap e_{k''}) = P(e_{k'}) * P(e_{k''})$.

We measure the goodness of an assignment of variants with the metric *expected client connectivity*. This metric is the expected value of the proportion of client pairs that are connected. To compute this value we consider the set of all possible

[1]We make an assumption of independence among compromise events in this work as this simplifies the presentation of the fundamental ideas in this work. However, as long as compromise events are not highly positively correlated (i.e., when one compromise event occurs then others are highly likely to happen), then all of our techniques and results still hold even though compromise events may not be completely independent.

combinations of compromise events C where $|C| = 2^{|E|}$ (C is the powerset of E). An element $c \in C$ is a subset of the compromise events, E, and corresponds to those compromise events occurring while any other compromise events do not occur. We can compute the proportion of clients connected given that those variants are compromised. We consider two clients to be connected if a path of non-compromised nodes exists between them.

Our goal is to maximize the expected client connectivity of a graph by strategically assigning variants. We call this problem the Diversity Assignment Problem.

Definition 2.2.1 *The Diversity Assignment Problem is to find the assignment of variants to nodes which maximizes the expected client connectivity. First, for a given assignment A and set of compromised variant events $c \in C$, we define a connectivity function $f_{A,c}(a,b)$ between two clients a and b as:*

$$f_{A,c}(a,b) = \begin{cases} \binom{|M|}{2}^{-1} & \textit{if clients } a \textit{ and } b \textit{ are connected} \\ & \textit{by a set of non-compromised nodes} \\ 0 & \textit{otherwise} \end{cases}$$

Then, the expected client connectivity is:

$$E\left[\sum_{\{a,b \in M : a<b\}} f_{A,c}(a,b)\right] = \sum_{c \in C}\left(\prod_{e_k \in c} P(e_k) \prod_{e_k \notin c}(1 - P(e_k))\right) * \left(\sum_{\{a,b \in M : a<b\}} f_{A,c}(a,b)\right)$$

The Diversity Assignment Problem is:

$$argmax_A\left(E\left[\sum_{\{a,b \in M : a<b\}} f_{A,c}(a,b)\right]\right)$$

As an illustrative example of the meaning of DAP we show Figures 2.1 & 2.2 as two ways to assign variants to an example topology. The three clients (squares) are connected through nodes (circles) by the shown lines. There are three client pairs for the three clients, and we describe how the expected client connectivity is calculated in each assignment. The variants are red and blue which have probabilities

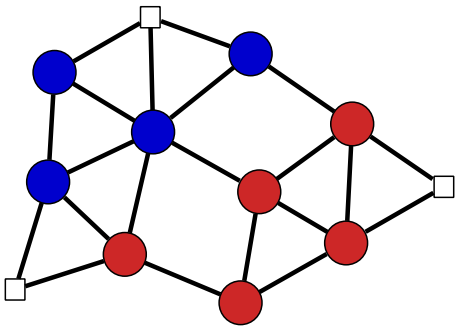

Figure 2.1.: Diversity assignment with 0.838 expected client connectivity on example topology.

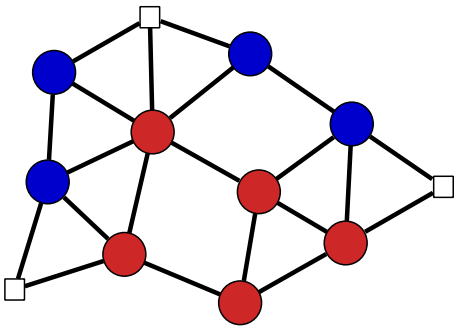

Figure 2.2.: Diversity assignment with 0.957 expected client connectivity on example topology.

of being compromised of 0.1 and 0.15 respectively. For the assignment in Figure 2.1, 3 client pairs are connected when neither variant is compromised which happens with probability $(1 - 0.1 * 0.15)$, 1 client pair is connected by the blue variant when just the red variant is compromised with probability $(1 - 0.1)$, and 1 client pair is connected by the red variant when just the blue variant is compromised with probability $(1 - 0.15)$. The resulting expected client connectivity is the following weighted sum $\frac{3}{3}(1-0.1*0.15)+\frac{1}{3}(1-0.1)+\frac{1}{3}(1-0.15) = 0.838$. For the assignment in Figure 2.2, 3 client pairs are connected when neither variant is compromised which happens with probability $(1 - 0.1 * 0.15)$, 3 client pairs are connected by the blue variant when just the red variant is compromised with probability $(1 - 0.1)$, and 2 client pairs is connected by the red variant when just the blue variant is compromised with probability $(1-0.15)$. The resulting expected client connectivity is the following weighted sum $\frac{3}{3}(1-0.1*0.15)+\frac{3}{3}(1-0.1)+\frac{2}{3}(1-0.15) = 0.957$. The second assignment has an increased expected client connectivity since more client pairs are connected in the scenarios where just one variant is compromised.

Theorem 2.2.1 *The Diversity Assignment Problem is NP-Hard with two or more variants.*

Proof We show that 3-SAT is polynomial-time Turing-reducible to the Diversity Assignment Problem. We will show that 3-SAT is solvable in polynomial-time if both

the DAP is used as a subroutine and the DAP is solvable in polynomial-time. Our proof is split into three parts (1) description of our transformation from a 3-SAT problem to the DAP, (2) showing a 3-SAT solution implies a DAP solution, and (3) showing a DAP solution implies a 3-SAT solution.

Transformation from 3-SAT to DAP. First we denote the variables for the input boolean expression of the 3-SAT as $\beta_1, \beta_2, ..., \beta_s$. Then, we denote the boolean expression as $(\beta_{\gamma_{1,1}}^{\lambda_{1,1}} + \beta_{\gamma_{1,2}}^{\lambda_{1,2}} + \beta_{\gamma_{1,3}}^{\lambda_{1,3}})(\beta_{\gamma_{2,1}}^{\lambda_{2,1}} + \beta_{\gamma_{2,2}}^{\lambda_{2,2}} + \beta_{\gamma_{2,3}}^{\lambda_{2,3}})...(\beta_{\gamma_{t,1}}^{\lambda_{t,1}} + \beta_{\gamma_{t,2}}^{\lambda_{t,2}} + \beta_{\gamma_{t,3}}^{\lambda_{t,3}})$. For all i and j, $\gamma_{i,j}$ is an index value, so $1 \leq \gamma_{i,j} \leq s$. For all i and j, $\lambda_{i,j} \in \{T, F\}$ where F denotes the complement of the boolean variable while T does not. The 3-SAT problem has s distinct variables and t clauses.

Let the DAP have two variants v_1 and v_2. Create $2s$ nodes denoted by $x_1^T, x_2^T, ..., x_s^T$ and $x_1^F, x_2^F, ...x_s^F$. These nodes will correspond to true and false boolean assignments for the 3-SAT variables $\beta_1, \beta_2, ..., \beta_s$.

Create $2t$ clients denoted by $a_1, a_2, ..., a_t$ and $b_1, b_2, ..., b_t$. For all i and j add the following two edges $(a_i, x_{\gamma_{i,j}}^{\lambda_{i,j}})$ and $(b_i, x_{\gamma_{i,j}}^{\lambda_{i,j}})$.

Create $2s(t+1)$ more clients denoted by $a_{1,j}, a_{2,j}, ..., a_{s,j}$ and $b_{1,j}, b_{2,j}, ..., b_{s,j}$ for all j such that $1 \leq j \leq t+1$. For all i and j add the following four edges $(a_{i,j}, x_i^T)$, $(b_{i,j}, x_i^T)$, $(a_{i,j}, x_i^F)$, and $(b_{i,j}, x_i^F)$. Note that for a given i the clients $a_{i,j}$ and $b_{i,j}$ for all j are equivalent in terms of their connections, and each i corresponds to variable in the 3-SAT problem.

Create nodes denoted by $y_1^T, y_2^T, ..., y_{\binom{|M|}{2}-\frac{|M|}{2}}^T$ and $y_1^F, y_2^F, ..., y_{\binom{|M|}{2}-\frac{|M|}{2}}^F$. So far, all clients have been created in pairs ($\frac{|M|}{2}$ of these pairs), and we use these y nodes to connect all remaining pairs. We add the following four edges for every client pair a, a' that is not meaningful, (a, y_i^T), (a', y_i^T), (a, y_i^F), and (a', y_i^F) such that a different i is used for each pair a, a'.

The last step in the construction is the selection of the compromise event probabilities $P(e_1)$ and $P(e_2)$ along with the the desired minimum expected client connectivity that must be found by the DAP to ensure a 3-SAT solution exists. We consider three types of client pairs which together make up all possible client pairs. We consider the

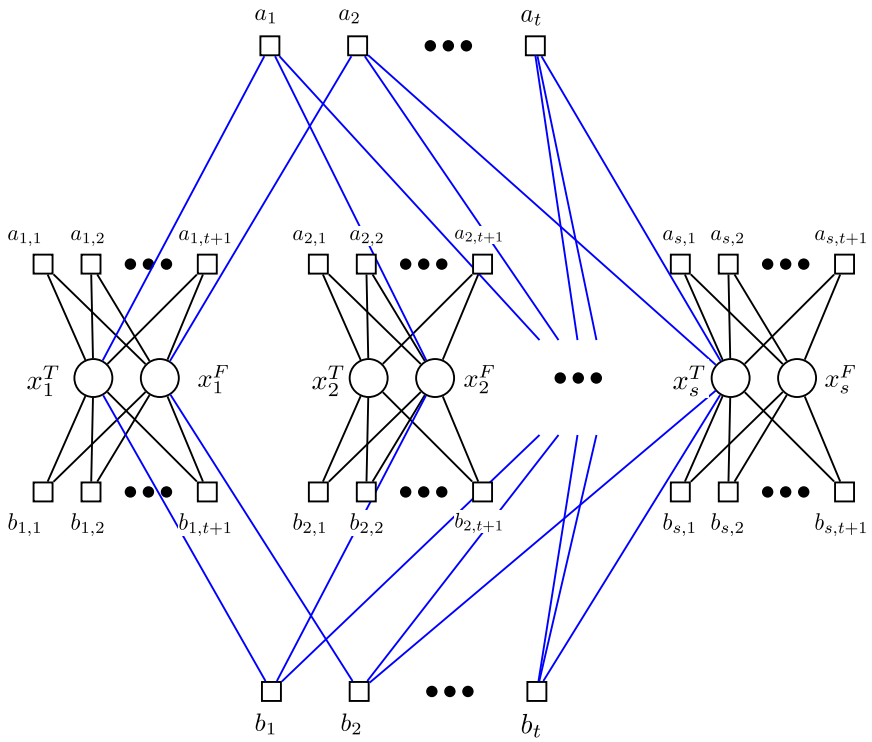

Figure 2.3.: DAP construction to solve the 3-SAT problem, $(\beta_1 + \bar{\beta}_2 + \beta_4)(\bar{\beta}_1 + \beta_7 + \beta_s)...(\bar{\beta}_8 + \beta_9 + \beta_s)$. Note that the y nodes are not included in this diagram.

DAP solved if we find an ECC greater or equal to the following desired value Γ where $\Gamma = \frac{\binom{|M|}{2} - \frac{|M|}{2}}{\binom{|M|}{2}}(1 - P(e_1)P(e_2)) + \frac{s(t+1)}{\binom{|M|}{2}}(1 - P(e_1)P(e_2)) + \frac{t}{\binom{|M|}{2}}(1 - P(e_1))$. Also, we must ensure that e_1 is significantly stronger than e_2 in terms of resilience by ensuring $P(e_1) < \frac{P(e_2)}{t + P(e_2) - tP(e_2)}$.

3-SAT solution implies a DAP solution. Given an assignment of true and false to the 3-SAT variables $\beta_1, \beta_2, ..., \beta_s$, we show there is a DAP assignment with ECC greater than or equal to Γ. For each $1 \leq i \leq s$, if β_i is true let x_i^T be assigned e_1 while x_i^F be assigned e_2, and for β_i switch this assignment. Assign e_1 to $y_1^T, y_2^T, ..., y_{\left(\binom{|M|}{2} - \frac{|M|}{2}\right)}^T$ and e_2 to $y_1^F, y_2^F, ..., y_{\left(\binom{|M|}{2} - \frac{|M|}{2}\right)}^F$.

Now we sum up different portions of the total ECC. First, since each 3-SAT variable is true or false, those $2s(t+1)$ client pairs $a_{1,j}, a_{2,j}, ..., a_{s,j}$ and $b_{1,j}, b_{2,j}, ..., b_{s,j}$ for all $1 \leq j \leq t+1$ are connected by both variants attributing $\frac{s(t+1)}{\binom{|M|}{2}}(1 - P(e_1)P(e_2))$ to the ECC. Second, each clause of 3-SAT is satisfied, those client pairs $a_1, a_2, ..., a_t$

and $b_1, b_2, ..., b_t$ are connected by at least the variant e_1 attributing $\frac{t}{\binom{|M|}{2}}(1-P(e_1))$ to the ECC. Finally, every remaining client pair is connected by both variants through the y nodes attributing $\frac{\binom{\binom{|M|}{2}}{2}-\frac{|M|}{2}}{\binom{|M|}{2}}(1-P(e_1)P(e_2))$ to the ECC. The sum of those three expressions is Γ which is the minimum desired ECC for a correct DAP solution.

DAP solution implies a 3-SAT solution. Given an assignment to DAP with an ECC greater than or equal to Γ we show a 3-SAT assignment exists. There are a total of $\binom{|M|}{2}$ client pairs where $|M| = 2s(t+1) + 2t$. View ECC as a sum of values from each client pair, and that value for a pair is $\frac{1-P(e_1)}{\binom{|M|}{2}}$ when connected by just variant 1, $\frac{1-P(e_2)}{\binom{|M|}{2}}$ when connected by just variant 2, or $\frac{1-P(e_1)P(e_2)}{\binom{|M|}{2}}$ when connected by both variant 1 and 2.

We start by showing show that each 3-SAT boolean variable β will have only one of $\beta, \bar{\beta}$ be true. From our transformation, this is equivalent to all those $2s(t+1)$ client pairs $a_{1,j}, a_{2,j}, ..., a_{s,j}$ and $b_{1,j}, b_{2,j}, ..., b_{s,j}$ for all $1 \leq j \leq t+1$ being connected by both variants. We show this with a contradictory argument by assuming some $(a_{i,j}, b_{i,j})$ pair which is only connected by a single variant. The existence of this single pair implies that at least $t+1$ pairs $(a_{i,1}, b_{i,1}), (a_{i,2}, b_{i,2}), ..., (a_{i,t+1}, b_{i,t+1})$ are only connected by a single variant as these $t+1$ client pairs have the same links to the same nodes. At best, these $t+1$ pairs contribute $\frac{(t+1)(1-P(e_1))}{\binom{|M|}{2}}$ which imposes an upper-bound on the final ECC of $\frac{\binom{|M|}{2}-(t+1)}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{t+1}{\binom{|M|}{2}}(1-P(e_1))$. This upper-bound is strictly less than Γ which contradicts our assumptions. Thus, we conclude that those $2s(t+1)$ client pairs are indeed connected by both variants.

It remains for us to show that each clause has at least one true value. From our transformation, the 3-SAT clause constraints are equivalent to those t client pairs $a_1, a_2, ..., a_t$ and $b_1, b_2, ..., b_t$ being connected by at least variant 1. We show this with a contradictory argument by assuming some (a_i, b_i) pair which is connected by only variant 2. This puts an upper-bound on the ECC of $\frac{\binom{|M|}{2}-1}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{1}{\binom{|M|}{2}}(1-P(e_2))$ from those t client pairs resulting in an upper-bound on the total

ECC of $\frac{\binom{|M|}{2}-t}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{t}{\binom{|M|}{2}}(1-P(e_2)4)$. We have the following from our selection of probabilities values:

$$\begin{aligned} P(e_1) &< \frac{P(e_2)}{t+P(e_2)-tP(e_2)} \\ \Rightarrow (t-1)(1-P(e_1)P(e_2)) + (1-P(e_2)) &< t(1-P(e_1)) \\ \Rightarrow \frac{t-1}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{1}{\binom{|M|}{2}}(1-P(e_2)) &< \frac{t}{\binom{|M|}{2}}(1-P(e_1)) \end{aligned}$$

With the above inequality, we can show the following inequality between the upper-bound ECC and Γ.

$$\begin{aligned} & \frac{\binom{|M|}{2}-1}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{1}{\binom{|M|}{2}}(1-P(e_2)) \\ = & \frac{\binom{|M|}{2}-t}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{t-1}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{1}{\binom{|M|}{2}}(1-P(e_2)) \\ < & \frac{\binom{|M|}{2}-t}{\binom{|M|}{2}}(1-P(e_1)P(e_2)) + \frac{t}{\binom{|M|}{2}}(1-P(e_1)) \\ = & \Gamma \end{aligned}$$

Thus, we have a contradiction and must assume that all client pairs are connected by variant 1.

Proof summary. We have shown that a correct 3-SAT assignment exists if and only if a resulting DAP exists with with sufficient expected client connectivity. Thus, the existence of an algorithm to solve DAP optimally in poly-time would imply an algorithm to solve 3-SAT in poly-time. ■

2.2.2 MIP Approach to DAP

Despite DAP being NP-Hard, many real-world network topologies are of limited size, so finding the optimal solution is of practical interest. To find the optimal solution, we chose to formulate the problem as a MIP and utilize an existing commercial solver, CPLEX [36]. A MIP is a linear program with the addition of integer constraints. The important implication of these integer constraints is that a MIP is not

solvable in polynomial time (while a linear program can be), but these integer constraints allow for formulations of many difficult combinatorial problems. Problems from other domains have also resorted to MIP to find optimal solutions to practical problems in the area of operations research [37–39]. MIP formulations are good for problems where the optimal is desired and no efficient algorithm is known as many MIP solvers [36,40,41] employ a variety of techniques to avoid exhaustively searching the entire space of feasible solutions.

Our MIP formulation is seemingly more complex than the mathematical formulation in Definition 2.2.1 mainly due to the expression of the function $f_{A,c}(a,b)$ as a MIP. This function's output depends on whether two clients are connected given an assignment and set of compromise events. In the MIP formulation we capture the same connectivity by setting up flow variables on each edge. When considering a specific source client, we count the number of other clients that are connected to this source client with the following constraints on these flow variables. The source client has no incoming flow and unbounded outgoing flow, each other client accepts at most one unit of incoming flow and has no outgoing flow, and each non-compromised node has equivalent incoming and outgoing flow. Compromised nodes have no incoming or outgoing flow, and a node is compromised when the node's variant assignment is included in the set of compromised events being considered. With these flow variables, $\sum_{\{a,b\in M:a<b\}} f_{A,c}(a,b)$ is equivalent to $\frac{1}{2} * \binom{|M|}{2}^{-1}$ multiplied by the total outgoing flow of the given clients for $|M|$ copies of the same graph and flow variables where each graph considers a different source client. Then we must copy these variables again, once for each possible set of compromise events.

Table 2.1 describes each symbol that we use in our MIP formulation. We present the objective function (Equation 2.1) followed by each constraint (Equations 2.2-2.10).

DAP objective:

$$\text{maximize}_{s,f} \quad \frac{1}{2} * \binom{|M|}{2}^{-1} * \sum_{c\in C, a\in M, x\in N} \left(\prod_{e_i \in c} P(e_i) \prod_{e_i \notin c} (1 - P(e_i)) \right) f_{c,a,a,x} \qquad (2.1)$$

Table 2.1: Notation for network diversity

Symbol	Description
N	Set of routing nodes. As our notation, these are x, y, z, etc.
M	Set of client nodes. As our notation, these are a, b, etc.
V	Set of variants.
E	Set of all compromise events. We index elements of E and V by k as their elements are related such that each e_k corresponds to the compromise event of the variant v_k.
C	Set of all possible compromise event sets, so $\|C\| = 2^{\|E\|}$. Each element $c \in C$ is a set of compromise events ($e \in E$) that are compromised.
$w_{i,j}$	Constants designating that edge $\{i,j\}$ exists. i and j can be either routing nodes or client nodes. Note that clients should not connect directly to other clients, so $i, j \in M \Rightarrow w_{i,j} = 0$
$f_{c,a,i,j}$	Measures the amount of flow that starts at client node a and travels on edge $\{i,j\}$ in compromise event set c. i and j can be either routing nodes or client nodes. Also, $c \in C$. This must be a non-negative value.
$s_{v,x}$	The variant assignment of routing node x. $s_{v,x}$ is 1 if x is variant v and 0 otherwise.

We maximize the expected client connectivity of the graph, over all compromise events. The first term $(\frac{1}{2} * \binom{|M|}{2})$ ensures that the result will be out of 1, rather than out of the number of possible connections between clients. The two products ensure that each possible compromise event is weighted by the probability that it happens. The f term is a measure of how much flow the given client a can push out onto the

network (specifically, $f_{c,a,i,j}$ measures the amount of flow that started at source client a that travels on edge $\{i,j\}$ in compromise case c). Because of all the constraints below, this is exactly a measure of how many other clients client a can connect to.

Variant constraints (I):

$$s_{v_i,x} = \{0,1\}, \quad v_i \in V, \quad x \in N \tag{2.2}$$

Routing nodes must be either entirely of a variant or entirely not of that variant. Fractional assignments are not allowed.

Variant constraints (II):

$$\sum_{v_i \in V} s_{v_i,x} = 1, \ x \in N \tag{2.3}$$

Routing nodes must be exactly one variant.

Node flow constraints:

$$\sum_{i \in N \cup (M - \{a\})} f_{c,a,x,i} - \sum_{i \in N \cup \{a\}} f_{c,a,i,x} = 0, c \in C, \ a \in M, \ x \in N \tag{2.4}$$

The flow (originating at source client node a) entering routing node x must equal the flow (originating at source client node a) exiting routing node x. This is enforced for each of the $|M|$ clients and for each of the $|N|$ nodes, separately. In other words, flow cannot get stuck in the middle of the network; it has to end at client nodes.

Client flow constraints (I):

$$\sum_{x \in N} f_{c,a,x,b} \leq 1, \ c \in C, \ a, b \in M, \ a \neq b \tag{2.5}$$

A client cannot accept more than one unit of flow from another client. This is so that we can count the total flow out of the source client to get the number of connected clients. Despite this constraint being ≤ 1, it can only take a value of 0 or 1 due to the other constraints and the objective. For the CPLEX solver [36], it is more efficient to enforce fewer integer constraints whenever possible.

Client flow constraints (II):

$$f_{c,a,x,a} = 0, \ c \in C, \ a \in M, \ x \in N \tag{2.6}$$

Traffic cannot start and end at the same client. In other words, a client cannot send to itself. Note that $\{x, a\}$ is any incoming edge into a.

Client flow constraints (III):

$$f_{c,a,b,x} = 0,\ c \in C,\ a, b \in M,\ x \in N,\ a \neq b \tag{2.7}$$

A destination client cannot send out flow. So, flow cannot use a client to reach other clients.

Topology constraints:

$$f_{c,a,i,j} \leq (|M| - 1) * w_{i,j},\ c \in C,\ a \in M,\ i, j \in (N \cup M) \tag{2.8}$$

Any pair of nodes with no edge between them (i.e., $w_{i,j} = 0$) cannot have any flow directly between them. It also underlines the fact that up to $|M| - 1$ units of flow originating at the same client can share the same edge.

Variant flow constraints (I):

$$\begin{aligned} &f_{c,a,x,i} \leq (|M| - 1) * \min_{e_i \in C}(1 - s_{v_i,x}), \\ &c \in C,\ a \in M,\ x \in N,\ i \in N \cup M \end{aligned} \tag{2.9}$$

The amount of flow out of a routing node must be 0 if that node is compromised. It also underlines the fact that no edge can carry more than $|M| - 1$ units of flow from any source client node a.

Variant flow constraints (II):

$$\begin{aligned} &f_{c,a,i,x} \leq (|M| - 1) * \min_{e_i \in c}(1 - s_{v_i,x}), \\ &c \in C,\ a \in M,\ i \in N \cup M,\ x \in N \end{aligned} \tag{2.10}$$

The amount of flow into a node must be 0 if that node is compromised. It also underlines the fact that no edge can carry more than $|M| - 1$ units of flow from any source client node a.

2.2.3 DAP on the Case Study Topology

We investigate the benefit of optimal diversity assignment on a realistic overlay network topology. The topology and compromise scenario are detailed in Table 2.2. Then, various assignments of diversity are shown on the case study topology with their corresponding expected client connectivity. We show assignments for DAP with increasing number of variants being used, and we investigate random assignments as a comparison with the optimal solution.

For a case study topology, we took a connectivity graph from a cloud network provider [32]. The nodes of the graph represent data centers located around the globe. Each node is assigned a single variant which means that the overlay routing element at that data center will utilize the selected variant. The edges of the graph represent overlay connectivity used on that cloud to connect the different data centers. This connectivity is provided by a number of Internet Service Providers at each data center. The clients in the graph represent either clients external to the cloud or infrastructure components of the cloud. Each client has multiple connections to the cloud to avoid a single point of failure. In this example we use three connections as that level of connectivity was quite prevalent in that network. This connectivity graph was designed with resiliency in mind, and without any consideration for diversity.

We assume some hypothetical scenario with three diverse variants represented by blue (darkest), yellow (lightest), and red (medium) having a 0.1, 0.15, and 0.2 probability of being compromised over some arbitrary period of time, respectively. Note that this example, while simplistic, provides an interesting insight into the benefits and risks of diversity.[2]

Figure 2.4 shows the optimal solution when only a single variant can be used. All the nodes are assigned with the least vulnerable variant. This corresponds to

[2]The purpose of these values is to give preference to one variant over another and to quantify an estimate of the system resiliency with diversity. While we select numbers to illustrate the main concepts, the resulting assignment would not be significantly different if other values were selected.

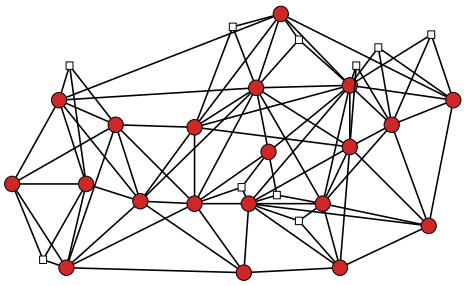

Figure 2.4.: Optimal assignment of one variant on case study topology achieving 0.9 expected client connectivity.

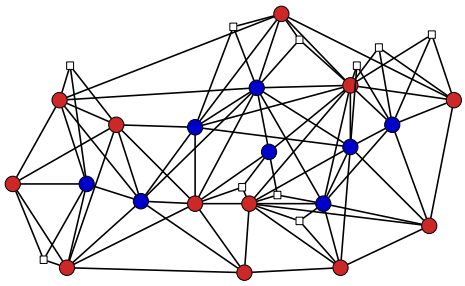

Figure 2.5.: Optimal assignment of two variants on case study topology achieving 0.985 expected client connectivity.

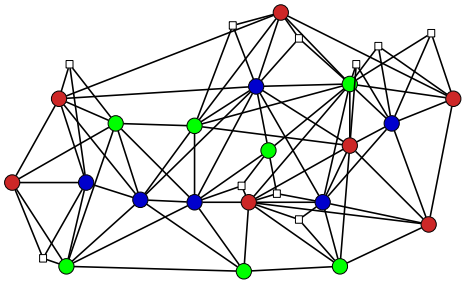

Figure 2.6.: Optimal assignment on case study topology of three variants achieving 0.997 expected client connectivity.

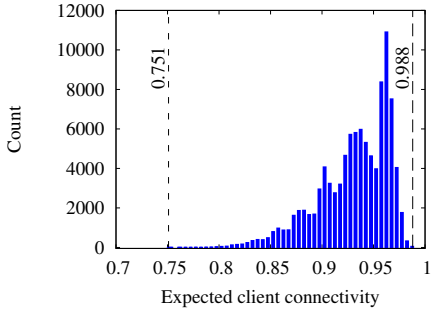

Figure 2.7.: Histogram of expected client connectivity of 100,000 random assignments on case study topology.

the situation where no diversity is used. The resulting network achieves an expected client connectivity of 0.9.

Figure 2.5 shows the optimal solution when two variants can be used. Each node is assigned with either of the two least vulnerable variants. The resulting network achieves an expected client connectivity of 0.985. Note that this is better than either variant by itself.

Figure 2.6 shows the optimal solution when three variants can be used. The resulting network achieves an expected client connectivity of 0.997. Notice that the optimal solution finds an assignment where any single variant is capable of connecting all clients. By adding a third more vulnerable variant actually makes the system significantly more resilient.

Table 2.2: Network characteristics for case study topology.

Symbol	Description
N	Set of 20 overlay nodes, shown in the figures as colored circles.
M	Set of 10 client nodes, shown in the figures as white squares.
V	Set of variants. v_1 represented by blue, v_2 represented by yellow, and v_3 represented by red. In Figure 2.4: $\{v_1\}$. In Figure 2.5: $\{v_1, v_2\}$. In Figure 2.6 and Figure 2.8: $\{v_1, v_2, v_3\}$.
C	Set of compromise event sets. In Figure 2.4: $\{\{\}, \{v_1\}\}$. In Figure 2.5: $\{\{\}, \{v_1\}, \{v_2\}, \{v_1, v_2\}\}$. In Figure 2.6 and Figure 2.8: $\{\{\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_1, v_2, v_3\}\}$.
$w_{i,j}$	Constants designating that edge $\{i, j\}$ exists. These are too numerous to be listed here, but can be observed from the figures.
E	The probability of compromise events for each variant are $P(e_1) = .1$, $P(e_2) = .15$, $P(e_3) = .2$.

As stated before, in this example, each client is connected to three routing nodes. If clients do not have at least three potential entry points into the network, then the availability of the connection is limited by the variants of the routing nodes that they are connected to. For example, if each client only connects to a single routing node, that connection would fail if either of the entry-point routing nodes is compromised. This is much more likely to occur than if there are three such entry-point routing nodes for each client, requiring at least three routing nodes to be compromised to cut the connection.

In this example, including variants *that have a higher but independent probability of being compromised* improves the overall system resiliency. This may be counter-intuitive, as adding weaker components to a system usually makes it weaker, not stronger. The independence of the different variants and the overall robustness of the network mean that adding additional, more vulnerable variants makes a system more resilient.

As discussed earlier, random assignment could be used instead of the optimal MIP approach. One might expect this approach to do well, since randomness often helps in adding diversity to systems. However, this does not necessarily lead to a good result. An example graph can be seen in Figure 2.8. This graph achieves an expected client connectivity of only 0.811, much worse than any of the other three graphs. In fact, it barely outperforms the worst of the three variants. This example graph comes from the bottom 1% of possible assignments and is given as an example of what could occur if the diversity assignment is not considered carefully.

Figure 2.7 is a histogram created with data from 100,000 random assignments of this graph. For this data set, the minimum and maximum are 0.751 and 0.988 respectively. The mean is 0.931 and the median is 0.937. As can be seen, most of the random assignments perform better than if the best variant is used by itself ($0.937 > 0.9$). However, very few of the random assignments come close to performing as well as the optimal assignment found by MIP.

The optimal solution of 0.997 expected client connectivity exists while the best random solution out of the 100,000 random assignment shown in Figure 2.7 was 0.988 expected client connectivity. Thus, even the best random solution out of numerous trials does not achieve the optimal solution. We define *expected client disconnectivity* to be the expected probability that communication between a client pair is broken, and this value is equivalent to (expected client disconnectivity) = 1 - (expected client connectivity). In terms of expected client disconnectivity the best random solution is 0.012 while the optimal solution is 0.003, so a client-to-client connection is broken four times less often with the optimal assignment.

Interestingly, the difference between what the optimal solution provides and the probability that at least one of the variants is non-compromised provides a metric for the quality of the connectivity resiliency of the graph.Ideally, we would want this distance to be zero, as in Figure 2.5 and Figure 2.6 of the provided example.

2.2.4 Near-optimal Assignments on Case Study Topology

Here, we aim to further understand why the problem is difficult and an optimal solution is several factors better than random assignments from the perspective of the expected client disconnectivity. To achieve this, compute the set of all assignments near the optimal solution in terms of expected client disconnectivity. The number of assignments found compared to the size of the search space further supports our claim that random assignments are typically much worse than the optimal assignment. Thus, applying techniques of this work to search for optimal assignments is important for any network aiming to achieve high resilience through diversity.

We search for solutions within a *disconnectivity_factor* of the optimal solution. This value is computed from a given expected client disconnectivity as follows:

$$disconnectivity\_factor = \frac{expected\_client\_disconnectivity}{OPT}$$

OPT is the optimal expected client disconnectivity. Intuitively a disconnectivity factor of two for an assignment implies that clients on average are disconnected twice as much as the optimal assignment.

An exhaustive search of the entire search space is prohibitively expensive for the three variant case, and we could not use this strategy to find all near-optimal solutions. However, we were able to find all solutions within a factor of optimal by leveraging advanced features of MIP solvers. After finding an optimal solution the solver can be set to continue searching for solutions. The solver avoids exhaustively searching the entire space by eliminating large portions of the search space through its branch and bound techniques. Given that the number of solutions found is small, then this procedure is quite efficient.

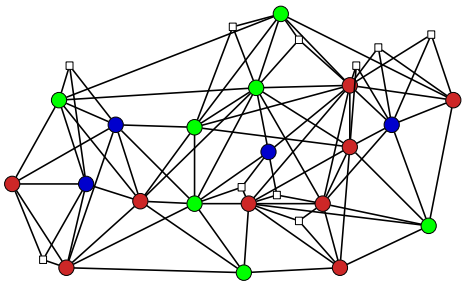

Figure 2.8.: Random assignment on case study topology achieving 0.881 expected client connectivity

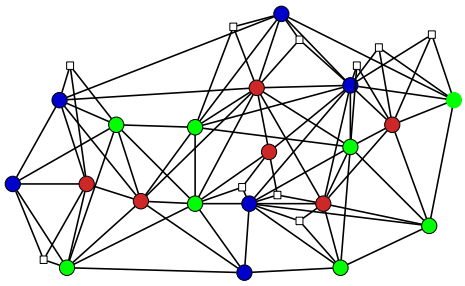

Figure 2.9.: Greedy assignment on case study topology achieving 0.992 expected client connectivity

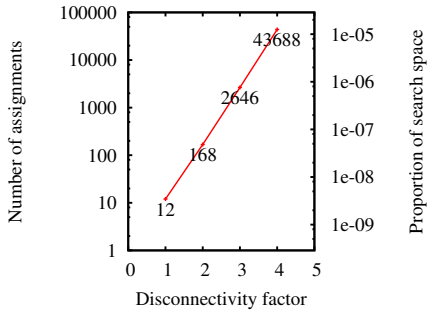

Figure 2.10.: Number of solutions within a given disconnectivity factor bound of optimal assignment on case study topology.

Figure 2.10 shows the number of solutions within a small factor of the optimal solution for the three variant scenario. Note the log-scale of the y-axis, and we also show the proportion of the search space that these solutions represent. The proportion of the search space indicates the probability that a random assignment has of achieving an assignment within a small factor of the optimal solution. Thus, a random assignment has a probability of $3 * 10^{-9}$ to achieve optimal, so that would require on the order of a billion topologies to be assigned and evaluated to find an optimal solution. The visually linear trend in this figure implies an exponential trend in the data due to the logscale of the y-axis. Thus, the number of solutions within a factor of optimal decreases exponentially with respect to decreasing factor, and this implies searching exponentially more assignments to expect to find such a solution.

2.3 Scaling Diversity Assignment

DAP is not tractable for large topologies since DAP is NP-Hard (see Theorem 2.2.1). To scale to larger topologies, we sacrifice optimality in order to ensure the algorithm completes within a polynomially-bounded time. In this section we present the Approximate DAP (A-DAP), a greedy approach to A-DAP, an example on the case study topology, and an evaluation on random topologies.

2.3.1 Approximate DAP (A-DAP)

A-DAP is similar to DAP, but A-DAP does not require that the problem be solved optimally. By relaxing this condition, we aim to find algorithms that run in polynomial time which are able to find large values of expected client connectivity. We do not formally define any restrictions on the goodness of the approximations as it is an open problem of whether a reasonable bound can be placed on the expected client connectivity achieved by a deterministic polynomial time algorithm. Instead we used random topologies to validate the goodness of expected client connectivities achieved by a greedy approach to A-DAP when compared with the optimal.

2.3.2 Greedy Approach to A-DAP

Our greedy approach incrementally assigns nodes to variants. At each incremental assignment the algorithm considers several candidate assignments and selects the one which provides the best immediate results. For a candidate set of incremental assignments we consider sets of nodes which can connect a client pair by a variant, so we consider at most $\binom{|M|}{2} * |V|$ candidate variant assignments. For a given client pair and variant, we compute the minimal number of unassigned nodes which must be assigned that variant to connect those clients by that variant. After this computation we have two values: the increase in expected client connectivity α and the number of newly assigned nodes β.

Given a set of candidate assignments that each have an α and β value, we select the one which maximizes $\frac{\alpha}{\beta}$. It is obvious why we want to find large α values, but it is equally important to ensure the β value is small as well. Smaller values of β allow for more nodes to remain unassigned and to be used to connect more client pairs by other variants in future assignments. This approach is analogous to the greedy choice in bin packing, as we select items with the highest payoff versus weight ratio to ensure that items are selected that increase overall payoff while allowing for more items to be picked in the future. Note that $\beta = 0$ is a trivial case where the candidate is simply removed from consideration as the client pair is already connected via the considered variant.

The pseudo-code of the algorithm is shown in Algorithm 1. Each iteration of the while loop (Line 3-15) creates a set of candidate variant assignments (Line 7), then selects the best candidate (Line 11-14), and lastly applies the assignment of that candidate to the topology (Line 15). This algorithm completes when no further client pairs can be connected by a variant, and the algorithm is guaranteed to complete in a bounded number of iterations since each step connects at least one new client pair via a variant (at most $|C| * |M|^2$ iterations).

2.3.3 A-DAP on the Case Study Topology

We consider the same scenario as in Section 2.2.3 with three variants. Figure 2.9 shows the assignment found by our greedy solution which achieves 0.992 expected client connectivity. Notice that all clients are connected via just the blue or yellow variants. However, two clients remain unconnected from the rest if only the red variant is uncompromised. The optimal solution found with the MIP formulation finds an assignment which connects all clients as long as any single variant is uncompromised. This loss of expected client connectivity is due to the greedy algorithm making choices in the early steps of the algorithm to connect clients via blue and yellow variants (the more resilient variants) which leaves fewer choices to connect clients via the

Algorithm 1 Greedy assignment heuristic

Variables

CPVC: Client Pair and Variant Combinations

VA: Variant Assignment

DVA: Delta Variant Assignment

CG: Connectivity Gain

BCG: Best Connectivity Gain

DVA: Delta Variant Assignment

BDVA: Best Delta Variant Assignment

α: Tunable parameter which affects the trade-off between increasing connectivity and minimizing the size of the DVA set

Functions

$f(\cdot,\cdot)$: Minimal set of unassigned overlay nodes that must be assigned a particular variant to connect a particular client pair

$g(\cdot)$: Overall connectivity for a particular variant assignment

Algorithm

1: CPVC := $M \times M \times V$
2: VA := $\emptyset$
3: **while** CPVC $\neq \emptyset$ **do**
4: BCG := 0
5: BDVA := $\emptyset$
6: **for all** $x \in$ CPVC **do**
7: DVA := $f(x, \text{VA})$
8: **if** DVA = $\emptyset$ **then**
9: CPVC := CPVC $- x$
10: **else**
11: CG := $\frac{g(\text{VA} \cup \text{DVA}) - g(\text{VA})}{|\text{DVA}|^{\alpha}}$
12: **if** CG > BCG **then**
13: BDVA := DVA
14: BCG := CG
15: VA := VA $\cup$ DVA

red variants. The greedy approach for the A-DAP took 0.38 seconds to complete while the MIP approach for the DAP took 396.13 seconds to complete. With far less computational requirements, the greedy heuristic does outperform the best of

the 100,000 random assignments (0.988 client connectivity) and comes close to the optimal solution.

2.3.4 A-DAP on Random Topologies

We present a methodology followed by results to answer the following questions of interest about the performance of the greedy heuristic for the A-DAP:

1. How does the goodness of the assignment of the greedy algorithm compare to other algorithms (random assignment and optimal) for the DAP on typical topologies?

2. How does the running time of the greedy heuristic for the A-DAP and the MIP approach for the DAP vary with typical topologies created with different parameters?

3. What are trends in the expected client connectivity over all the assignment algorithms when varying topology parameters?

We select expected client connectivity and running time to measure for each algorithm. Expected client connectivity is a measure of how well the algorithm performs, which can be compared with MIP's optimal value. Running time is a measure of how quickly the algorithm will terminate with an expected client connectivity.

We generate random topologies in a similar way to random wireless topologies. That is, we place the desired number of nodes and clients randomly inside a two-dimensional box. Then based on a density parameter, we give each node and client a range. All nodes and client within the range have an edge between them. The density parameter is the average number of connections for each node or client. Note that client to client edges are not added. We can create many random topologies given a number of nodes and a density value. We chose to limit the number of nodes in order to ensure that the optimal value could be calculated for comparison. Topologies constructed in this way are obviously representative of wireless contexts, but they are

also quite similar to overlay topologies because overlay topologies include many short, well-behaved links.

Given topology parameters, we create 30 random topologies and run the three algorithms on these topologies. We average the expected client connectivity and running times obtained for each algorithm over the 30 runs. For the running time values of the MIP formulation, it is important to note that we use the software package CPLEX with a quad-core 3.4 Ghz Intel processor which does leverage all cores.

We describe how they answer each of the initial questions that we proposed.

Question 1. The goodness of an algorithm's assignment is the expected client connectivity. This is upper-bounded by the optimal value (which the MIP approach always achieves). The greedy heuristic outperformed the random assignment and was quite close to the optimal value, independent of varying either density (Figure 2.11) or the number of nodes (Figure 2.13).

Question 2. The running time of the greedy heuristic is on the order of milliseconds, which is barely visible when compared to the running time of the MIP-based approach. Figure 2.12 shows the MIP approach running time for varying density values. The running time is low for small density values since most variant assignments result in poor expected client connectivity, allowing the branch-and-bound algorithm of CPLEX to avoid searching the majority of variant assignments. The running time is also low for high density values since a dense graph has many possible optimal assignments and the branch-and-bound algorithm can terminate early after finding any of them. Thus, the problem is hardest for moderate density values. The running time of both algorithms when varying the size of the network is shown in Figure 2.14. The MIP approach running time grows nearly linearly over these input parameters, but this relationship is potentially exponential according to Theorem 2.2.1. The MIP approach running time is still significantly greater than the greedy approach.

Question 3. The trend of expected client connectivity is similar among all three algorithms. The expected client connectivity increases as density increases (Figure 2.11), which is expected since more edges allow more possibilities for clients to

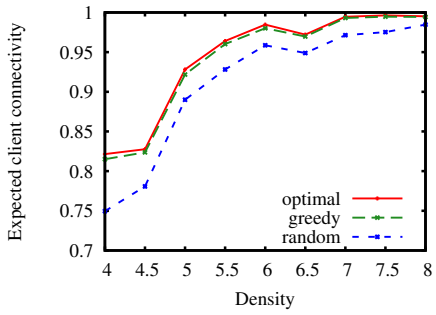

Figure 2.11.: Expected client connectivity of random, optimal, and greedy assignments on random topologies for 25 nodes and varied density.

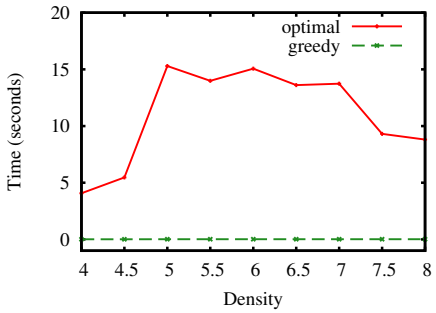

Figure 2.12.: Time of optimal and greedy assignments on random topologies for 25 nodes and varied density.

become connected. The expected client connectivity decreases as the number of nodes increases (Figure 2.13). By keeping the density constant and increasing the number of nodes, the graph becomes less connected and therefore less resilient.

From these results we see that the greedy heuristic outperforms the random algorithm while being quite close to the optimal solution, and the greedy heuristic is far more efficient in terms of running time and is polynomially-bounded while the MIP formulation is not. Hence, on larger topologies where the MIP formulation cannot be computed, the greedy heuristic is a decent substitute. Another interesting result is that the expected client connectivity decreases with more nodes when keeping the density constant. So, the density or node degree must increase to retain high levels of expected client connectivity when the number of nodes increases in the topology.

2.4 Diversity Assignment for Dynamic Topologies

In practice, networks typically do not remain static throughout their lifetime. Instead an initial setup is deployed and over time nodes are dynamically added. One trivial way to leverage diversity in an online scenario is to solve DAP every time a change in the topology occurs. However, for many classes of diversity it is highly expensive or even prohibitive to reassign an existing node of one variant to a different

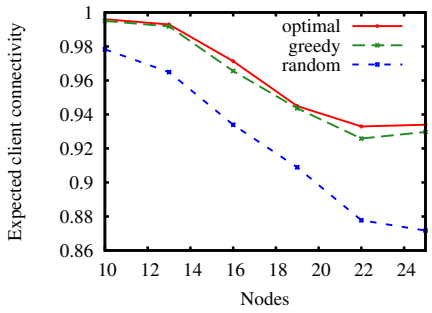

Figure 2.13.: Expected client connectivity of random, optimal, and greedy assignments on random topologies for 6 density and varied number of nodes.

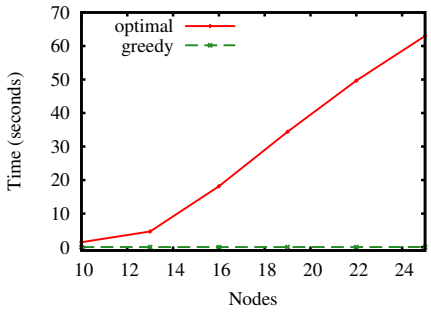

Figure 2.14.: Time of optimal and greedy assignments on random topologies for 6 density and varied number of nodes.

variant as it may be difficult to revoke access from an administrator or expensive to reinstall a new diverse software. A more realistic solution is to always keep the existing variant assignment and just assign variants to the newly added nodes.

We next describe the specific model which captures our assumptions. Then we describe an approach to solve this problem and evaluate this approach for an online scenario.

2.4.1 Online DAP (O-DAP)

We assume that there is some variant assignment that exists for a set of nodes which we denote by A'. A new set of nodes are added to the topology with given links to existing nodes in the network. We assume that there is no knowledge of future topology changes, so we cannot anticipate where new nodes may be added, which is an assumption that is realistic in practice. We seek a variant assignment, A, which retains all of the variant assignments of A'. We denote this problem as the Online Diversity Assignment Problem (O-DAP) with formal details in Definition 2.4.1.

Definition 2.4.1 *The Online Diversity Assignment Problem extends DAP by adding additional constraints. There exists some set of nodes which have already been assigned variants, and this existing assignment is denoted by A'. We are using the*

notation $A' \subset A$ to convey that the assignment A must retain the assignment of A'. The assignment A does have the freedom to assign variants in any way to those new nodes added to the network which are not contained in the assignment A'. Reusing notation from Definition 2.2.1, we can define the Online Diversity Assignment Problem as:

$$\begin{array}{ll} argmax_A & \left(E\left[\sum_{\{a,b \in M: a<b\}} f_{A,c}(a,b)\right]\right) \\ subject\ to & A' \subset A \end{array}$$

Theorem 2.4.1 *The Online Diversity Assignment Problem is NP-Hard with two or more variants.*

Proof Let $A' = \emptyset$, and then O-DAP is equivalent to DAP. Theorem 2.2.1 states that DAP is NP-Hard. ■

2.4.2 MIP Approach to O-DAP

We detail a MIP approach for O-DAP as it is typically easy to solve O-DAP optimally because the number of nodes which are added to a network at once is usually small. Given that x nodes are added to the network and x is small, then the search space, $|V|^x$, is reasonably small as well. Exhaustive search by checking all possible variant assignment combinations of the x new nodes could be used. However, as we already have a MIP formulation available to us, it is simple to reformulate the MIP that optimally solves DAP to optimally solve O-DAP. Specifically, we add the following constraint to the same MIP formulation for DAP from Section 2.2.2.

Online variant constraints:

$$s_{v_i,x} = 1, \quad \langle x, v_i \rangle \in A' \tag{2.11}$$

Nodes that have been assigned previously by A' (elements in A' are two tuples denoting a node and its corresponding variant assignment) must keep that variant assignment.

Theorem 2.4.1 states that O-DAP is NP-Hard. In scenarios where the number of nodes being added dynamically is large, it is possible to extend the greedy approach for A-DAP into an online version that approximates O-DAP.

2.4.3 O-DAP on the Case Study Topology

The expected client connectivity of DAP is always greater than or equal to that of O-DAP for the same topology because O-DAP only adds constraints to DAP. For a real deployment this means that reconfiguring all of the variants to be optimal when each dynamic change occurs always results in equal or better expected client connectivity compared with an online version where existing variants cannot be reconfigured. We measure this degradation in expected client connectivity for this evaluation.

The size of the incremental node additions influences the resulting expected client connectivity. A network which does many additions of just a few nodes per topology change will suffer more in expected client connectivity than a network which adds many nodes per topology change. Networks which add many nodes at once allow O-DAP to consider more combinations of variant assignment choices. We consider the following two scenarios for dynamic topologies in our evaluation:

- **reconfigure**: DAP is solved and that solution is applied to all nodes in the network. As reconfiguring is typically an unreasonable approach in practice, we use this as a baseline for comparison with the online approach.
- **online-x**: Nodes are added to the network x at a time, and O-DAP is solved where the variants of existing nodes in the network cannot be changed.

We evaluate these strategies with the following scenario on our case study topology. We initialize a scenario topology from our case study topology by selecting 8 of the 20 nodes. Next, we solve the DAP for the scenario topology. Then, we add nodes to the scenario topology based on the strategy being used (i.e., x for online-x) until all 20 nodes are in the scenario topology. We keep the order in which nodes are added

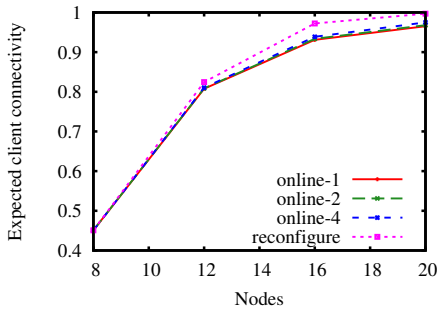

Figure 2.15.: Expected client connectivity achieved by reconfigure and online assignments as case study topology grows.

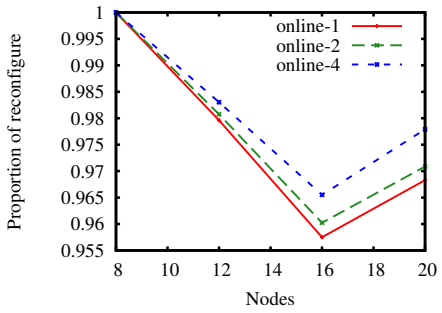

Figure 2.16.: Proportion of reconfigure's expected client connectivity achieved by online assignments as case study topology grows.

to the scenario topology consistent across different strategies. For example, the first four nodes added one at a time in online-1 will be the same nodes added all at once in the first iteration of online-4. Note that while the topologies will match, the variant assignments may differ. We repeat this process for 30 different scenarios, randomly varying which nodes are in the initial topology and the order in which the remaining nodes are added. Finally we show averages over these 30 scenarios.

Figure 2.15 shows the results for the evaluated strategies. From this figure it is evident that the online strategies achieve less expected client connectivity than the reconfigure strategy. To better compare these strategies we show Figure 2.16, which instead of showing absolute expected client connectivity, shows the proportion of the expected client connectivity achieved by the online versions to the expected client connectivity achieved by the reconfigure strategy, which is optimal. The online strategies always achieve at least 95% of the reconfigure strategy. More dynamic strategies reduce client connectivity, but in our experiment this degradation was never more than 1% when comparing online-1 and online-4. The downward then upward trend (V-shape) of Figure 2.16 is due to the following: the initial downward trend is due to the online strategies diverging more and more from reconfigure strategy at larger node values, the latter upward trend is due to the general high connectivity

in the topology which results in any online assignment strategy being close to the reconfigure's optimal assignment as the network becomes fully assigned.

These results indicate that diversity is not limited to static deployments, but that diversity can also be applied effectively when networks are dynamic. However, a trade-off exists; reconfiguring the entire network is costly but it yields the optimal expected client connectivity. It is up to the system designer to judge the correct balance between resilience and reconfiguration cost. For systems with very high resiliency goals, this reconfiguration may be necessary. When the highest resiliency is not necessary, the O-DAP approach can be utilized to eliminate the costs of reconfiguring nodes while sacrificing resiliency. In our experiments, we observed that the O-DAP approach achieved resiliency no worse than 95% of optimal.

2.5 Diversity Assignment for Specific Applications

Certain distributed systems that maintain consistent state pride themselves on their ability to tolerate part of the system failing. State machine replication protocols with this property include Paxos [33], Byzantine Fault Tolerance (BFT) [34], Prime [42], and Aardvark [43], where Prime and Aardvark give additional performance guarantees even while the system is under attack. These protocols explicitly state their assumptions about the proportion of replicas that must be correct for safety and liveness properties to hold. However, an equally important consideration is that a sufficient number of correct replicas must be able to communicate with each other via the underlying network. If we view the state machine replicas as clients of the underlying network, then applying diversity to the network improves the resiliency of the overall system.

We use these state machine replication protocols as an example of how to customize DAP for a specific client application. The state machine replication protocols have specific connectivity needs among replicas that must be satisfied to ensure safety and liveness. We show how DAP is customized to better ensure the network meets

these requirements, and we show how such customization can be helpful in a realistic scenario. The steps we take here to customize DAP can be followed to create other versions that meet the specific connectivity needs of other distributed systems.

The expected client connectivity from DAP maximizes the expected value of the proportion of client pairs that are connected. This is a reasonable metric for resiliency of many applications, and it could even work well for state machine replication in certain scenarios. However, an approach that takes into account the connectivity requirements of the specific application (in this case, state machine replication) may result in higher overall resiliency. We refine DAP to exactly match the needs of a replicated state machine protocol by maximizing the probability that a specific sized connected component exists among the replicas.

2.5.1 Connected Component DAP (CC-DAP)

The goal of this algorithm is to optimize the probability that g clients can communicate with each other. The connected component size g can be derived from the specific state machine replication protocol, we demonstrate this later with BFT. We denote this problem as the Connected Component Diversity Assignment Problem (CC-DAP) with formal details in Defintion 2.5.1 (notation comes from Table 2.1). Unsurprisingly, this problem is also NP-Hard as stated in Theorem 2.5.1.

Definition 2.5.1 *The Connected Component Diversity Assignment Problem is to find the assignment of variants to nodes which maximizes the probability of a component of clients being connected. First, we define the random variable X_A which is the size of the largest connected component of clients given a variant assignment A. This variable is random as it depends on the random events E. Then, the Connected Component Diversity Assignment Problem is:*

$$argmax_A \left(P(X_A \geq g)\right)$$

Theorem 2.5.1 *The Connected Component Diversity Assignment Problem is NP-Hard with two or more variants.*

Proof We show that a variant of 3-SAT which is denoted as Not-All-Equal 3-SAT [44] is polynomial-time Turing-reducible to CC-DAP. Not-All-Equal 3-SAT has the same setup as 3-SAT except clauses where all variables are true is not allowed; there must be a mixture of true and false variables. We will show that Not-All-Equal 3-SAT is solvable in polynomial-time if both the CC-DAP is used as a subroutine and the CC-DAP is solvable in polynomial-time.

Assume the same network setup as in the proof for NP-Hardness of DAP which is visualized in Figure 2.3. This proof differs as we replace the last step of assigning $P(e_1)$ and $P(e_2)$ and use CC-DAP instead of DAP.

In this proof, we can let $P(e_1)$ and $P(e_2)$ take on any value in the range $(0, 1)$ as opposed to requiring certain constraints on these values.

For the CC-DAP algorithm, we aim to maximize the probability of a connected component of $|M|$ clients, i.e., all clients in a connected component.

If and only if CC-DAP finds a probability of $1 - P(e_1) * P(e_2)$ for a connected component of $|M|$ clients, then we have also found a solution to Not-All-Equal 3-SAT due to the following: CC-DAP with a probability of $1 - P(e_1) * P(e_2)$ implies each client pair is connected by both variants v_1 and v_2. The connections between client pairs $a_{i,j}$ and $b_{i,j}$ ensures that $\beta_i \neq \bar{\beta}_i$ for each β_i in Not-All-Equal 3-SAT. The connections between client pairs a_i and b_i ensure that each clause i in the Not-All-Equal 3-SAT problem is connected by at least one true value and at least one false value which is the requirement for Not-All-Equal 3-SAT. Having at least one false value for a clause is a special condition that distinguishes it from standard 3-SAT, and this is the reason we reduce from Not-All-Equal 3-SAT in this proof. ■

2.5.2 MIP Approach to CC-DAP

For the MIP formulation we keep the constraints in Equations 2.2-2.10 from Section 2.2.2, reformulate the objective function, and add new constraints. Our new objective and constraints include new variables which are used to keep track of which

subset of clients are used for a connected component $\beta_{c,a}$ as well as variables to check if the connected component α_c is large enough. We describe the purpose of the new objective and each new constraint in detail to show how it captures the CC-DAP problem.

CC-DAP objective:

$$\text{maximize}_{s,f,\alpha,\beta} \quad \sum_{c \in C} \left(\prod_{e_i \in c} (P(e_i)) \prod_{e_i \notin c} (1 - P(e_i)) \right) \alpha_c \tag{2.12}$$

We maximize the probability that a g-sized connected component exists, over all compromise events. The two products ensure that each possible compromise event is weighted by the probability that it happens. α_c is 1 if a connected component of size g is present under compromise event c and 0 otherwise.

Component constraint (I):

$$\alpha_c = \{0, 1\},\ c \in C \tag{2.13}$$

A g-sized connected component either exists under compromise event c, or it does not.

Component constraint (II):

$$\beta_{c,a} = \{0, 1\},\ c \in C,\ a \in M \tag{2.14}$$

$\beta_{c,a}$ is 1 if client a is in the g-sized connected component under compromise event c, and 0 otherwise.

Component constraint (III):

$$g = \sum_{a \in M} \beta_{c,a},\ c \in C \tag{2.15}$$

A valid connected component under compromise event c must be of size g. In any other case, this constraint will not be met. Note, if a larger connected component could exist, this constraint ensures that only g clients are considered, which is required for other constraints.

Component flow constraint (I):

$$f_{c,a,x,b} \leq \beta_{c,b},\ c \in C,\ a,b \in M,\ x \in N, a \neq b \tag{2.16}$$

A client b, in the connected component under compromise event c, cannot accept more than one unit of flow from another client a. If b is not in the connected component, it will not accept any flow.

Component flow constraint (II):

$$f_{c,a,a,x} \leq (g-1) * \beta_{c,a},\ c \in C,\ a \in M,\ x \in N \tag{2.17}$$

A client a, in the connected component under compromise event c, cannot send more than $g-1$ units of flow, enough for every other client in the connected component. If a is not in the connected component, it will not send any flow.

Component satisfaction constraints:

$$g * (g-1) * \alpha_c = \sum_{a \in M, x \in N} f_{c,a,a,x},\ c \in C \tag{2.18}$$

If there exists a g-sized connected component under compromise event c, then there are a total of $g * (g-1)$ units of flow in the network. If no such connected component exists, the total flow is 0.

2.5.3 CC-DAP on Example Ring Topology

We provide a quick example on a topology which is contrived to show the advantage of using CC-DAP for applications such as Paxos and BFT. In the following subsections we show this on the case study topology as well.

Figure 2.19 shows the configuration as well as the optimal assignment for DAP. The optimal solution connects all clients by the strongest variant and is able to connect one additional client pair by the second strongest variant.

Figure 2.17 shows with the same configuration the optimal assignment for CC-DAP with a connected component size of 9. For 16 replicas, 9 is the smallest required

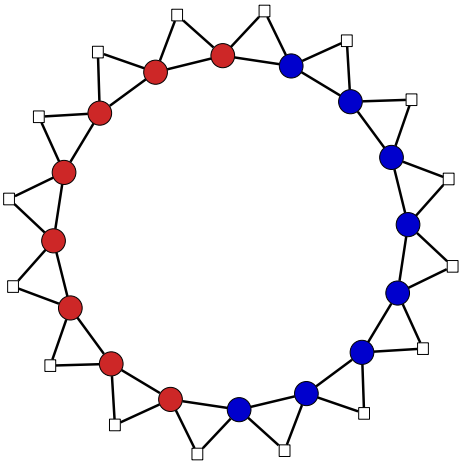

Figure 2.17.: Assignment from optimizing probability of Paxos progress on ring topology.

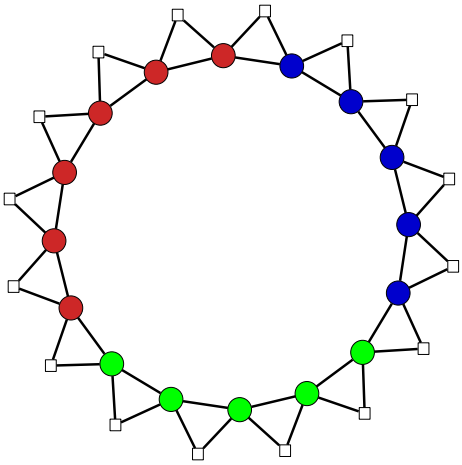

Figure 2.18.: Assignment from optimizing probability of BFT progress on ring topology.

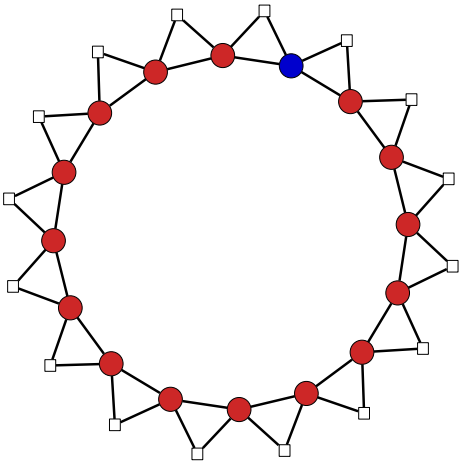

Figure 2.19.: Assignment from optimizing expected client connectivity on ring topology.

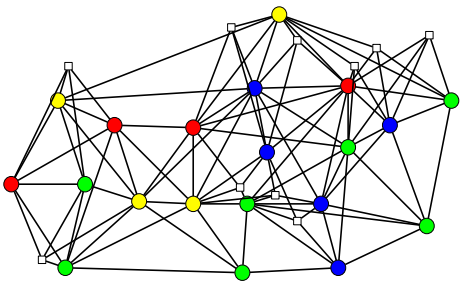

Figure 2.20.: Four variant CC-DAP assignment for connected components of size 8 for BFT.

connected component for Paxos to make progress. The optimal assignment is able to ensure a connected component of 9 with the strongest variant and second strongest variant independently. That is, as long as either the red or blue variant are not compromised Paxos will make progress.

Table 2.3: Values of three metrics for ring topology for three assignments that each maximize their own metric

Assignment	Expect client connectivity	Paxos probability of progress	BFT probability of progress
Figure 2.19	**0.9004**	0.9	0.9
Figure 2.17	0.831	**0.985**	0.765
Figure 2.18	0.787	0.941	**0.941**

Figure 2.18 shows the optimal assignment for CC-DAP with a connected component size of 11 which is appropriate for BFT with 16 replicas. Here, the optimal assignment ensures 11 clients are connected when any single variant is compromised. That is, if just red, just blue, or just green variants are compromised, then BFT will make progress.

Table 2.3 shows the values of each metric for each assignment. Its important to notice how poor certain metrics are when they are not being optimized. Thus, this example shows the value a network can provide when knowing the application being run among the clients.

2.5.4 CC-DAP for Paxos on the Case Study Topology

We show compelling examples of using CC-DAP for assignment in the context of Paxos in this subsection and BFT in the next subsection. For a non-trivial comparison between DAP and CC-DAP, we seek scenarios where DAP cannot connect all client pairs by every variant individually. These scenarios are trivial for CC-DAP since an optimal DAP assignment is also an optimal CC-DAP assignment. We slightly change the setup from Section 2.2.3 to ensure these interesting scenarios. In this Paxos scenario, we add a new variant v_4 where $P(e_4) = 0.25$ represented in the figures by the color green (second lightest). In the BFT scenario, we had this same new variant in addition to adding new connections between the clients and the routing nodes.

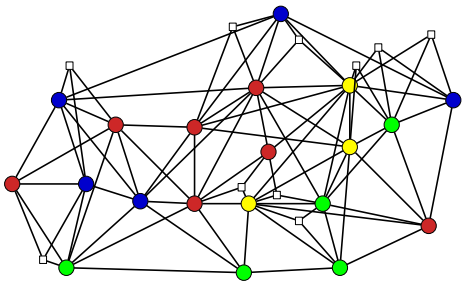

Figure 2.21.: Three variant CC-DAP assignment for connected components of size 6 for Paxos.

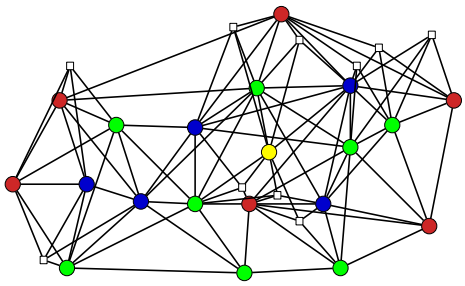

Figure 2.22.: Four variant DAP assignment.

BFT requires this extra modification of including new connections since the nature of BFT's assumptions requires larger connected components.

Paxos maintains consistent state given that there are at most f_s fail-stop failures when using a total of $n = 2f_s + 1$ replicas. In this Paxos scenario, we assume replicas may be partitioned from each other due to attacks on the routing nodes. A client being partitioned from the others is equivalent to a fail-stop failure. We assume these are the only types of fail-stop failures, i.e., the network may fail but the replicas themselves do not fail. Given that we have 10 replicas in total, implies that $f_s = 4$. As a result, the required connected component size is $g = n - f_s = 6$.

Figure 2.21 shows the assignment when using the MIP approach for CC-DAP while Figure 2.6 from our earlier explanation shows the assignment when using the MIP approach for DAP. In Figure 2.21, the probability that 6 of the clients will be able to communicate is 0.99925 with an expected client connectivity of 0.9675. In contrast, in Figure 2.6, the probability that 6 of the clients will be able to communicate is only 0.997 while having an expected client connectivity of 0.997 as well. In essence, CC-DAP is able to sacrifice some of the expected client connectivity to increase the probability that a connected component of the desired size will be present.

2.5.5 CC-DAP for BFT on the Case Study Topology

BFT tolerates up to f byzantine failures when using a total of $n = 3f + 1$ replicas. We will view these f failures as a combination of f_b, byzantine replicas, and f_s, fail-stop replicas (indistinguishable from replicas that have been partitioned away). The choice of values for f_b and f_s are left to the system designer. There is trade-off between f_b and f_s, governed by the trustworthiness of the replicas vs. the trustworthiness of the network routing nodes, but further details are beyond the scope of this research. For our example, we choose $f_b = 1$. Given that we have 10 replicas in total, implying that $f = 3$, the system can tolerate two replicas being partitioned away ($f_s = 2$) and still tolerate one byzantine fault. As a result, the required connected component size is $g = n - f_s = 8$.

For the results of assignments for BFT, we observe a similar trend to the results of the Paxos scenario. Figure 2.20 shows the assignment when using the MIP approach for CC-DAP that achieves a probability of 0.99925 that 8 of the clients communicate. Figure 2.22 shows the assignment when using the MIP approach for DAP which has only a probability of 0.997 that 8 of the clients communicate.

2.6 Optimizing Client Traffic Patterns

Specific client connectivity requirements were considered in the previous section. Those connectivity requirements were based on ensuring cliques of communication exist even after router compromise occurs. Here we offer optimization based on client traffic patterns. This offers a selection of importance for certain client pairs based on the amount of traffic or monetary value of the traffic. The choice between these two types of utility is based on the type of network and information available to the network. Instead of finding cliques in CC-DAP or treating all client pairs equally in DAP, we allow arbitrary selection of value for each client pair.

The network can understand the utility gained for offering communication between each pair of clients by observing the quantity of traffic between client pairs or based

on payment the network may receive for delivering data between client-pairs. By performing diversity assignment based on this information, the network can better maximize utility by ensuring more important client pairs have communication with higher tolerance to router compromise.

2.6.1 Weighted DAP (W-DAP)

This problem has a similar definition to DAP with the exception that traffic weights are included. Each client pair has a given traffic weight which allows us to compute a weighted expected client connectivity for a given assignment. W-DAP is to maximize this weighted expected client connectivity instead of expected client connectivity. We formulate this problem in Definition 2.6.1.

Definition 2.6.1 *The Weighted Diversity Assignment Problem is to find the assignment of variants to nodes which maximizes the weighted expected client connectivity. Let $T_{a,b}$ be the chosen weight value by the network operator for a client pair a, b. Let T^{sum} be the sum of all weight values for each pair of clients $T^{sum} = \sum_{\{a,b \in M : a<b\}} T_{a,b}$, and this is used for normalizing weighted expected client connectivity between 0 and 1. Let $g_{A,c}(a,b)$ be a weighted connectivity function defined as follows:*

$$g_{A,c}(a,b) = \begin{cases} \frac{T_{a,b}}{T^{sum}} & \text{if clients } a \text{ and } b \text{ are connected} \\ & \text{by a set of non-compromised nodes} \\ 0 & \text{otherwise} \end{cases}$$

The weighted expected client connectivity is (same as expected client connectivity from Definition 2.2.1 with the exception of using function g instead of f):

$$E\left[\sum_{\{a,b \in M : a<b\}} g_{A,c}(a,b)\right] = \sum_{c \in C}\left(\prod_{e_k \in c} P(e_k) \prod_{e_k \notin c}(1 - P(e_k))\right) * \left(\sum_{\{a,b \in M : a<b\}} g_{A,c}(a,b)\right)$$

The Weighted Diversity Assignment Problem is:

$$argmax_A\left(E\left[\sum_{\{a,b \in M : a<b\}} g_{A,c}(a,b)\right]\right)$$

We know that W-DAP is also hard to solve given that DAP is a special case of W-DAP, and we proved DAP is hard to solve. This special case occurs when all traffic weights are equal.

Theorem 2.6.1 *The Weighted Diversity Assignment Problem is NP-Hard with two or more variants.*

Proof In the case where $\forall a, b, c, d \in M, T_{a,b} = T_{c,d}$, solving W-DAP is equivalent to DAP. ■

2.6.2 MIP Approach to W-DAP

For the MIP formulation we keep the same formulation of DAP from Section 2.2.2 while reformulating the objective and certain constraints to correctly include traffic weights between clients. The following modifications to DAP will allow W-DAP to be solved.

W-DAP objective:

$$\text{maximize}_{s,f} \quad \frac{1}{2 * T^{sum}} * \sum_{c \in C, a \in M, x \in N} \left(\prod_{e_i \in c} P(e_i) \prod_{e_i \notin c} (1 - P(e_i)) \right) * f_{c,a,a,x} \tag{2.19}$$

We maximize the weighted expected client connectivity of the graph, over all compromise events. The first normalizing term changes to $\frac{1}{2 * T^{sum}}$ instead of the number of client pairs. This is just a detail that normalizes the solutions of W-DAP between 0 and 1, and the 2 in the denominator is to handle a small discrepancy where T^{sum} only sums over each client-pair once while here we sum over each client pair twice (both directions). The connectivity values $f_{c,a,a,x}$ will be weighted correctly due to the following changes in constraints.

Weighted client flow constraints (I):

$$\sum_{x \in N} f_{c,a,x,b} \leq T_{a,b}, \ c \in C, \ a, b \in M, \ a \neq b \tag{2.20}$$

This is the main change that ensures flow variables optimize for weights. Clients accept an amount of flow according to the traffic weights. This forces the flow from a client a to a client b summed over all paths to take on a value of $T_{a,b}$ if and only if at least one path from a to b exists given the compromise event c.

Weighted topology constraints:

$$f_{c,a,i,j} \leq \sum_{b \in M, a \neq b} (T_{a,b}) * w_{i,j},\ c \in C,\ a \in M, \\ i, j \in (N \cup M) \tag{2.21}$$

Here, $w_{i,j}$ denotes whether an edge exists (value of 1) or does not exist (value of 0) between node i and j. We must change the previous value of $(|M| - 1)$ with this summation over $T_{a,b}$ values since this is the largest possible amount of flow that may be needed for a given edge.

Weighted variant flow constraints (I):

$$f_{c,a,x,i} \leq \sum_{b \in M, a \neq b} (T_{a,b}) * \min_{e_i \in C}(1 - s_{v_i,x}), \\ c \in C,\ a \in M,\ x \in N,\ i \in N \cup M \tag{2.22}$$

Weighted variant flow constraints (II):

$$f_{c,a,i,x} \leq \sum_{b \in M, a \neq b} (T_{a,b}) * \min_{e_i \in c}(1 - s_{v_i,x}), \\ c \in C,\ a \in M,\ i \in N \cup M,\ x \in N \tag{2.23}$$

These two weighted variant flow constraints also change the maximum amount of flow for an edge.

2.6.3 W-DAP on the Case Study Topology

We provide the following example of how W-DAP is useful in a realistic scenario. We consider the case with four variants from Section 2.5. This scenario is selected as optimal assignments must make interesting choices because the topology cannot

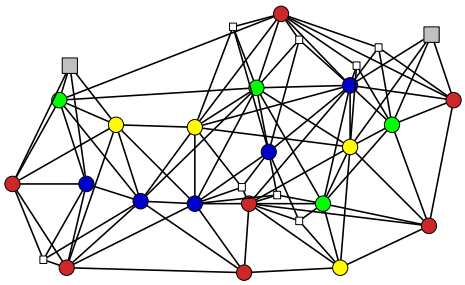

Figure 2.23.: Assignment with W-DAP with dominant weights between gray-filled client-pair on case study topology.

connect all clients by each of the four variants independently. We let one client-pair be a high priority client pair such that any possible connectivity is more important than connecting other client-pairs. To do this for the important client pair a, b we let $T_{a,b} = T_{b,a} = 100000$ (exact value unimportant, just needs to be very large) while all other traffic weights are 1. The optimization will treat this client pair a, b as a primary client pair and ensure assignments ensure highest connectivity for that pair. Then, the other client pairs are assigned to maximize connectivity as long as it does not hinder the connectivity between a and b.

Figure 2.23 illustrates the optimal assignment to W-DAP in this scenario. The weighted expected client connectivity is approximately 0.99925, and this value is significant since it is the probability that at least any one variant is not compromised, computed by: $1 - 0.1 * 0.15 * 0.2 * 0.25 = 0.99925$. This is a close approximation of the client connectivity value because the dominant client pair is connected when any single variant exists as you can see the four node-disjoint paths between these two clients. This solution is more difficult than just finding any set of four node-disjoint paths between these two clients as that is an easily solvable problem. Out of all possible node-disjoint paths between these two clients, this one maximizes the expected client connectivity of the remaining nodes. Thus, the network finds the best scenario for this primary client pair, and it can still aim to satisfy the other client pairs as well. Due to the constraint of having to connect this primary client pair, the general expected client connectivity does suffer as it is 0.99482 compared to the

0.9975 value that was found in Figure 2.22 which did not optimize for a high priority client pair.

2.7 Errors in Variant Compromise Estimation

Up to now, we have assumed the true assignment compromise values are known and independent with each other. In a realistic scenario, these assignment values could be selected based on expert opinion or extracted from real-world statistics. Both techniques cannot be perfectly accurate. In this section we investigate what occurs when assignment is based on imperfect information.

2.7.1 Methodology to Investigate Erroneous Information

We establish certain parameters and values that we use to investigate the effects of errors in information.

We define three scenarios for obtaining an ECC from solving DAP:

- AVAIL_ECC_AVAIL_INFO is the ECC value based on available information for an assignment solved with the available information. This is the connectivity that a network operator expects when using an assignment based on solving DAP with available information.
- REAL_ECC_AVAIL_INFO is the ECC value based on real information for an assignment solved with available information. This is the realistic connectivity that a network operator will actually achieve when using an assignment based on solving DAP with available information.
- REAL_ECC_REAL_INFO is the ECC value based on real information for an assignment solved with real information. This is the connectivity that could have been achieved if the network operator had perfect information.

We consider two types of discrepancies between available and real information. First, some compromise events have incorrect values, that is, $P'(e_i) = P(e_i)+\Delta_i$ where

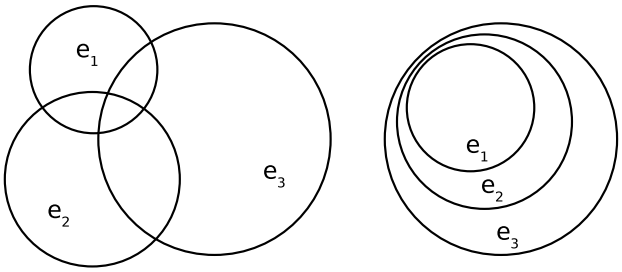

Figure 2.24.: Depiction of the difference between independent events on the left ($\alpha = 0$) and full depedence on the right ($\alpha = 1$).

P' is the available probability distribution, P is the actual probability distribution, and Δ_i is the error for a particular compromise event. Second, the compromise events are not fully independent, that is, $P'(E) = (1 - \alpha) * P(E) + \alpha * D(E)$ where E is a set of compromise events, $D(\cdot)$ is the probability distribution if there is complete dependence among the events, and α is a parameter determining how correlated the variants actually are ($\alpha = 0$ is complete independence while $\alpha = 1$ is the most extreme dependence). We show in Table 2.4 an example of the probability distribution of $P'(\cdot)$ for differing values of α with three variants $P(e_1) = 0.1, P(e_2) = 0.15, P(e_3) = 0.2$. We also illustrate the independent and full dependence scenarios with Venn diagrams in Figure 2.24.

With a discrepancy between the available and real information and letting $x =$ AVAIL_ECC_AVAIL_INFO, $y =$ REAL_ECC_AVAIL_INFO, and $z =$ REAL_ECC_REAL_INFO we observe the following two types of errors.

- CONFIDENCE_ERROR $= \frac{|x-y|}{y}$ is the error in how confident a network operator is with the created assignment.
- CONNECTIVITY_ERROR $= \frac{|y-z|}{z}$ is the error in how much worse an assignment based on available information is versus an assignment based on the real information.

Table 2.4: Probability distributions for different α values.

e_3	e_2	e_1	$\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 1.0$
			0.612	0.659	0.706	0.800
		1	0.153	0.127	0.102	0.050
	1		0.108	0.094	0.079	0.050
1			0.068	0.051	0.034	0.000
	1	1	0.027	0.020	0.014	0.000
1		1	0.017	0.013	0.009	0.000
1	1		0.012	0.009	0.006	0.000
1	1	1	0.003	0.027	0.052	0.100

2.7.2 Error Analysis on Random Topologies

We show the effect of a discrepancy in the compromise probability of a single variant. Then, we show the effect of discrepancy in the assumption of complete independence among variants. We use random topologies with similar settings to the random topologies in Section 2.3.4. Each topology had 5 clients and 3 variants with compromise probabilities $P(e_1) = 0.1, P(e_2) = 0.15, P(e_3) = 0.2$. In that section we showed results when varying density and number of nodes. For varying density we fixed the number of nodes at 25, and for varying the number of nodes we fixed the density at 6. These values were chosen as these parameters produced interesting topologies, that is, the topologies were connected but not too connected that assignment was trivial. Thus, in this section we fix the number of nodes to 25 and density to 6 for interesting topologies to investigate the effects on assignment when there are discrepancies between available and real information.

Figure 2.25 shows the CONFIDENCE_ERROR and CONNECTIVITY_ERROR when the $P(e_2)$ used for assignment is different from the real $P(e_2)$. We show the errors when the available information has a compromise probability greater than the actual compromise probability ($\Delta_2 < 0$) and less than the actual compromise

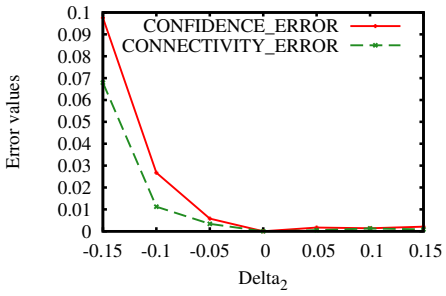

Figure 2.25.: Error values with a discrepency between real and available information in Δ_2.

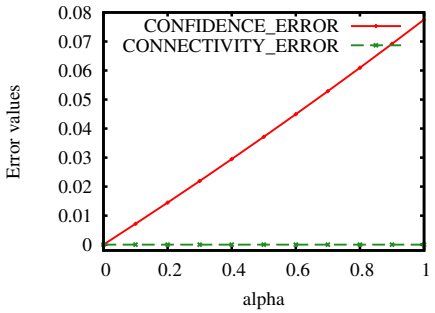

Figure 2.26.: Error values with a discrepency between real and available information in α.

probability ($\Delta_2 > 0$). We see the greatest errors for both types of errors when $P(e_2)$ is believed to be a weaker variant than it truly is, that is, $\Delta_2 < 0$. This is due to the assignment algorithm preferring to select v_3 over v_2 when forced to make a choice between these two. We observe little errors when $\Delta_2 > 0$ which is the case when the available information indicates v_2 is a stronger variant than actuality. This is due to the random topologies having many client pairs that can be connected by two paths. Therefore it is not as detrimental for the assignment to prefer v_1 over v_2. There is some error which indicates that the preference of v_2 over v_1 is slightly detrimental.

Figure 2.26 shows the errors when the assignment selected is based on the assumption of complete independence. We note in this case that REAL_ECC_AVAIL_INFO = REAL_ECC_REAL_INFO, as the assignments are actually the same despite the change in independence information, and thus CONNECTIVITY_ERROR is always equal to zero in this case. However, CONFIDENCE_ERROR is nonzero since any connectivity believed to be found by a network operator is less than the realistic connectivity since dependence among variants is detrimental to diversity. We see that this error increases linearly with α.

2.8 Summary

This chapter has illustrated the resiliency benefits gained when shifting from homogeneous networks with potential vulnerabilities shared across all routing nodes to networks that leverage optimally-assigned diversity. We summarize our key findings. First, randomly assigning diversity to a realistic network demonstrated surprisingly poor results, which motivated the need to formulate and solve the Diversity Assignment Problem (DAP). Second, we propose an algorithm that solves DAP optimally and show the results on medium-sized random networks as well as a realistic network. Third, we propose an algorithm that approximates the optimal solution, scaling well to large networks, and show that on random networks the resulting resiliency is close to that of the optimal solution. Fourth, we show how to optimize for the specific resiliency needs of an application running on the network. We applied this to Paxos and BFT, finding that the probability of making progress can be significantly increased. Lastly, as it is difficult to exactly estimate compromise probabilities we showed how discrepancies between compromise probabilities used for assignment and the real compromise probabilities affect assignment and resilience.

3 NULL SPACE BASED DEFENSE FOR POLLUTION ATTACKS IN WIRELESS NETWORK CODING

In the previous chapter we demonstrated how to ensure a surviving network of nodes under sophisticated attacks which may be able to compromise certain variants of routers. After such an attack, the network is partially compromised with a set of honest nodes that provide a high amount of connectivity between clients. For this chapter and the next chapter we shift focus to the problem of routing data among the set of honest nodes using network coding while preventing any attempts by compromised nodes to disrupt this routing. This chapter focuses on mitigating the most well-known attack which threatens the use of network coding in byzantine environments, namely, pollution attacks.

Network coding routing deviates from traditional store-and-forward routing by allowing intermediate nodes to code packets together. Network coding is particularly applicable in wireless networks where the broadcast nature and opportunistic reception of the wireless medium allows network coding to surpass traditional routing protocols by taking advantage of any overheard packets. Theoretical results [45] have shown that network coding achieves higher network capacity than traditional networks with little coordination [46]. In the context of wireless networks, network coding has emperically achieved increased throughput [5], increased reliability [6, 7], and reduced power consumption [8]. Numerous practical systems [9–14] have been proposed to achieve these improvements.

Network coding systems are vulnerable to pollution attacks [47] in which adversaries acting as intermediate nodes inject bogus packets into the network. The injection of polluted packets can also occur in traditional store-and-forward routing protocols. In this case, since intermediate nodes just forward packets, any scheme that provides data source authentication (such as digital signatures) is an effective

defense in detecting packets that were not created by the source. In random network coding, intermediate nodes code new packets by computing random linear combinations of the packets received from upstream nodes. In this case, traditional data source authentication mechanisms are not applicable. Such authentication schemes need to have homomorphic properties in order to allow intermediate nodes to verify that the packets they are coding are in turn linear combinations of packets that originated at the source. Even more, while pollution attacks require little resources from the attacker they have an epidemic effect in network coding systems as honest intermediate nodes unknowingly amplify the attack by creating new packets based on the received infected packets and forwarding the resulting new malformed packets in the network.

Several pollution defenses exist for pollution attacks relying on cryptographic, information theory, or algebraic mechanisms. *Cryptographic-based schemes* [15–19] create homomorphic digital signatures and hashes. These techniques impose prohibitive communication and computation overhead in wireless networks [48]. A cryptographic solution based on MACs [21] imposes prohibitive overhead in the presence of multiple byzantine adversaries, which we show in Section 3.4. *Information theoretical-based schemes* [25,26] code redundant information into packets, allowing receivers to recover correct packets when some packets are polluted. Such approaches hinder the system performance as they limit the throughput of the network coding system based on the adversary's available bandwidth or impose restrictions on broadcasts of intermediate nodes.

Algebraic-based schemes verify that packets received by forwarders belong to the space defined by the original packets sent by the source. Two representative approaches are the schemes in [23] and [24]. The scheme in [23] creates non-cryptographic checksums and relies its security on the difference between the time when a packet was received and the time when the checksum used to verify the packet was created. The scheme is effective but requires time synchronization and delays packets before forwarding them. The scheme in [24] uses null space properties to provide nodes with

vectors (referred to as null keys) belonging to the null space of source packets that are used to algebraically verify that the packets belong to the same space as the source packets. Because the defined null keys are large and would impose high load on the source and high communication in the network, the source distributes null keys only to first hop neighbors and relies on the homomorphic property of the null keys to have intermediate nodes create null keys for their downstream nodes. Thus, the scheme relies its security on path diversity, to ensure that each node will have a null key that spans a space much larger than any one adversary can know about. Path diversity is possible in peer-to-peer networks because links can be inserted and deleted easily. However, this assumption is not valid in wireless networks where there is less path diversity and where the topology is optimized based on the wireless link qualities, making the scheme insecure in wireless networks.

We propose a new defense against pollution attacks based on the null space properties and without relying on any assumptions about the network topology or time synchronization. Specifically:

- We propose Split Null Keys (SNK), a new defense against pollution attacks that splits a null key in two components, a small generation dependent one and a larger generation independent one chosen randomly[1]. As a result, after an initialization phase when the generation independent component is distributed, only a small portion of a null key (160 bytes) that is dependent on the data from each generation must be updated for each generation. The small communication overhead allows the source to securely distribute the update individually to each forwarder. Since each forwarder receives its own update securely an attacker cannot exploit the knowledge of the null key, and thus no path diversity is required. SNK has a smaller communication cost per generation than previous work and a very small computation cost which consists of inexpensive matrix

[1]In a network coding scheme the source disseminates the entire sequence of packets in sub-sequences called *generations*.

multiplications. Our scheme also does not delay packets for verification and scales with the number of colluding adversaries in the network.

- We formally prove that a probabilistic polynomial time adversary that can control any set of byzantine forwarder nodes and overhear all communication in the network, cannot pollute a target victim node. The intuition is that the large, generation independent portion of the null key serves as a secret between the source and forwarder, so keeping this portion constant across multiple generations does not deteriorate security as long as it remains secret. Even if forwarders collude the adversary cannot know how the null key of the target victim node and the null keys known by the adversary overlap due to the fact that all null keys are generated independently and randomly.

- We validate the performance and overhead of our scheme with extensive simulations using a well-known network coding system for wireless networks (MORE [9]) and realistic link quality measurements from the Roofnet [49] experimental testbed. Our results show that SNK imposes little communication overhead with an average of 25 kbps to distribute null keys. SNK outperforms previous defenses against pollution attacks in both benign and adversarial networks achieving better throughput and latency. Finally, SNK retains the benefits of network coding by performing better than a traditional, secure, store-and-forward routing protocol ARAN [50] (a secure version of the well-known shortest path routing protocol AODV).

The rest of the chapter is organized as follows. We present the system and attacker model in Section 3.1 and our approach in Section 3.2. We analyze the security of our scheme in Section 3.3 and evaluate its performance and overhead in Section 3.4. We conclude the chapter in Section 3.5.

Table 3.1: Notation for network coding systems and the SNK protocol

n	Number of plain packets per generation
m	Number of symbols per plain packet
q	Field for a symbol, the symbol size is $\log_2(q)$
$\mathbf{X}$	Data matrix of plain packets, size n by m
$\mathbf{I}$	Identity matrix
$\mathbf{A}$	Augmented data matrix of size n by $n+m$ $\mathbf{A} = [\mathbf{I}\|\mathbf{X}]$
$\mathbf{c}$	Coded packet, it is an element of the row space of $\mathbf{A}$
$\mathbf{V}$	Coding header at a destination used for decoding
$\mathbf{B}$	Null space matrix of $\mathbf{A}$ which has size $n+m$ by m
$\mathbf{0}$	Matrix of all zeros
$\mathbf{K}_i$	Null key for forwarder i which is subspace of the column space of $\mathbf{B}$
ω	Security parameter for the number of null keys a forwarder uses for verification
$\tilde{\mathbf{K}}_i$	Generation dependent null key, first n rows of $\mathbf{K}_i$
$\bar{\mathbf{K}}_i$	Generation independent null key, last m columns of $\mathbf{K}_i$
$\mathbf{S}$	First n rows of $\mathbf{B}$
$\mathbf{T}$	Last m rows of $\mathbf{B}$
$\mathbf{G}_i$	Null key generator for null key $\mathbf{K}_i$, $\mathbf{B} * \mathbf{G}_i = \mathbf{K}_i$
θ	Number of possible forwarders for a source
β	Number of forwarders for a flow

3.1 Pollution Attack Model

We describe the network coding system and adversarial model. The notation we use is presented in Table 3.1.

3.1.1 Network Coding System

We assume an intra-flow network coding system with one *source* that sends data via *forwarders* to one or more *destinations*. The source sends data in *generations*. A

generation represents a subsequence of packets from the total number of packets and consists of n *plain packets*. A plain packet consists of m *symbols* which are elements of the finite field $\mathbb{F}_q$ (each symbol is of size $\log_2(q)$ bits). The plain packets are encoded in a *data matrix* $\mathbf{X}$ of size n by m such that each row is a plain packet. The matrix $\mathbf{X}$ is augmented by the identity matrix $\mathbf{I}$ to form an *augmented data matrix* $\mathbf{A} = [\mathbf{I}|\mathbf{X}]$. The identity matrix is inserted to serve as a *coding header* for decoding the coded packets at a destination.

The source creates *coded packets* $\mathbf{c}$ by generating random vectors that belong to the row space of $\mathbf{A}$ and sends these coded packets to forwarders and destinations that store them in a *coding buffer*. Forwarders create coded packets by generating random vectors from their coding buffer. Destinations eventually obtain a coding buffer spanning the same space as the row space of $\mathbf{A}$, i.e., each destination has $[\mathbf{V}|\mathbf{V} * \mathbf{X}]$ where $\mathbf{V}$ is the coding header and has full rank. The destination decodes the packets by computing $\mathbf{V}^{-1} * [\mathbf{V}|\mathbf{V} * \mathbf{X}] = [\mathbf{V}^{-1} * \mathbf{V}|\mathbf{V}^{-1} * \mathbf{V} * \mathbf{X}] = [\mathbf{I}|\mathbf{X}]$ to obtain the original data matrix $\mathbf{X}$.

Parameter selection. The selection of parameters n and m impacts performance of a network coding system. The parameter n must be set to ensure a sufficient number of packets are coded together to obtain network coding gains. However, n affects *coding overhead* which is the overhead for distributing the coding header. Each generation contains a data matrix, $\mathbf{X}$, that is $n * m * \log_2(q)$ bits, and each generation the source distributes a larger, augmented data matrix, $\mathbf{A}$, that is $n^2 * \log_2(q) + n * m * \log_2(q)$ bits. The extra $n^2 * \log_2(q)$ bits distributed are coding overhead. Thus, the selection of n and m must ensure that $n << m$ to minimize coding overhead.

3.1.2 Adversarial Model

We assume that an attacker mounts pollution attacks by injecting *polluted coded packets* in the network. A polluted packet is a coded packet that is not an element of the row space of $\mathbf{A}$. Nodes downstream from the attacker accept this packet as

valid and store it in their coding buffer. Forwarders with polluted coded packets in their coding buffers will create new coded packets that are also polluted. Thus, the forwarders unknowingly act as pollution attackers themselves, and the attack propagates epidemically throughout the network. Destinations with polluted coded packets in their coding buffers will not obtain the data sent by the source upon decoding. An attacker could be either a rogue node without the credentials to be part of the network or a node with the credentials to be in the network. We assume that multiple attackers exist and can collude.

3.2 Split Null Keys (SNK)

We first overview the null space properties that our scheme relies on and then describe our scheme. In the following, the term *forwarders* also refers to destinations.

3.2.1 Null Space Properties

Let the null space of the row space of the matrix $\mathbf{A}$ (of size n by $n+m$) be the column space of $\mathbf{B}$, then we have:

$$\mathbf{A} * \mathbf{B} = \mathbf{0}$$

and $\mathbf{B}$ is a basis for the null space of the row space of $\mathbf{A}$.

According to the rank nullity theorem

$$r(\mathbf{A}) + r(\mathbf{B}) = n + m \Rightarrow r(\mathbf{B}) = m$$

so the rank of the column space of $\mathbf{B}$, $r(\mathbf{B})$, is m. Thus, $\mathbf{B}$ is a matrix of size $(n+m)$ by m.

Definition 3.2.1 *A* null key *is a matrix that spans a subspace of the column space of* $\boldsymbol{B}$*. We denote a null key by* $\boldsymbol{K}$*.*

We now show three properties for null keys in relation to valid coded packets and polluted coded packets.

Lemma 3.2.1 *A valid coded packet multiplied by a null key always equals a zero vector.*

Proof By definition, any vector of the row space of $\mathbf{A}$ multiplied with any vector of the column space of $\mathbf{B}$ results in a zero. ■

Lemma 3.2.2 *A randomly generated coded packet $\boldsymbol{c}$ has a probability of $(\frac{1}{q})^{\omega}$ to satisfy $\boldsymbol{c} * \boldsymbol{K} = \boldsymbol{0}$ where $\boldsymbol{K}$ is a null key with rank ω and q is the symbol size.*

Proof Let K be the column space of $\mathbf{K}$. The probability that a randomly chosen coded packet $\mathbf{c}$ yields $\mathbf{c} * \mathbf{K} = \mathbf{0}$ is equivalent to the probability that a randomly chosen coded packet is a vector that is in the null space K. The null space of K is the space of all vectors $\mathbf{c}'$ such that $\mathbf{c}' * \mathbf{K} = \mathbf{0}$. The rank of K is ω, so the rank of the null space of K is $n + m - \omega$ according to the rank nullity theorem. The number of vectors in the null space of K is $q^{n+m-\omega}$, and the number of possible coded packets is q^{n+m}. Thus, the probability that a randomly chosen coded packet is a vector that is in the null space K is $\frac{q^{n+m-\omega}}{q^{n+m}} = (\frac{1}{q})^{\omega}$. ■

Lemma 3.2.3 *Let $\boldsymbol{K}'$ be a matrix that represents a subspace of the column space of the null key $\boldsymbol{K}$ where $\boldsymbol{K}'$ has rank ω' and $\boldsymbol{K}$ has rank ω ($\omega' \leq \omega$). Then, a randomly selected coded packet $\boldsymbol{c}$ from the set of coded packets that satisfy $\boldsymbol{c} * \boldsymbol{K}' = \boldsymbol{0}$ has a probability of $(\frac{1}{q})^{(\omega-\omega')}$ to satisfy $\boldsymbol{c} * \boldsymbol{K} = \boldsymbol{0}$.*

Proof Let the column space of $\mathbf{K}$ and $\mathbf{K}'$ be denoted by K and K' respectively. The ranks of the null spaces of K and K' are $n + m - \omega$ and $n + m - \omega'$ respectively. The number of vectors in the null spaces of K and K' are $q^{n+m-\omega}$ and $q^{n+m-\omega'}$ respectively. Given that $\mathbf{K}'$ is a linear combination of the vectors of $\mathbf{K}$ we have that any coded packet $\mathbf{c}'$ that satisfies $\mathbf{c}' * \mathbf{K} = \mathbf{0}$ also satisfies $\mathbf{c}' * \mathbf{K}' = \mathbf{0}$, so null space of K is a subset of the null space of K'. A randomly selected coded packet $\mathbf{c}$ from the null space of K' has a probability of $\frac{q^{n+m-\omega}}{q^{n+m-\omega'}} = (\frac{1}{q})^{(\omega-\omega')}$ to satisfy $\mathbf{c} * \mathbf{K} = \mathbf{0}$. ■

Using null keys to detect pollution. Based on Lemma 3.2.1 and Lemma 3.2.2, polluted packets can be identified as follows. A forwarder i having the null key $\mathbf{K}_i$

and receiving a coded packet $\mathbf{c}$ will compute $\mathbf{c} * \mathbf{K}_i$. If the result is a zero vector then the coded packet is accepted, otherwise the coded packet is dropped. In the case the packet is accepted, there is low probability that the packet may still be polluted, $(\frac{1}{q})^{\omega}$ if the packet is chosen randomly according to Lemma 3.2.2 which is controlled by the column rank of the null key, ω. We show in Section 3.3 that an attacker cannot do better than generating polluted coded packets randomly when attempting to pass a victim node's verification test.

Impact on security when dimensions of null keys overlap. If the dimensions of the null keys at two forwarders overlap, and a malicious forwarder knows the dimensions that overlap, then the malicious forwarder can pollute the other forwarder with a high probability given in Lemma 3.2.3. The higher the overlap, the higher the success of crafting a polluted packet. Thus, it is essential that an attacker does not know the dimensions that overlap between their null key and the null keys at other honest forwarders in the network.

3.2.2 SNK Overview

As a null key is a matrix that is a subspace of the column space of $\mathbf{B}$ (which is an $(n+m)$ by m matrix), the size of a null key for a forwarder is $(n+m)$ by ω, where ω is the column rank of $\mathbf{K}_i$. As stated in Lemma 3.2.3 the dimensions of null keys should not overlap, so if the source distributes all the null keys, this results in a very high communication overhead. Typical settings for wireless networks are $q = 256$ (1 byte symbols), $n = 32$, and $m = 1468$ ($n + m = 1500$ typical wireless packet size), $w = 5$ to prevent random guessing (Lemma 3.2.2) so even with few forwarders in the network, the source will spend more time sending null keys than data.

In previous work [24] it was proposed to reduce this communication overhead by having the source send null keys only to the first hop nodes and rely on forwarder nodes to generate null keys for downstream nodes by combining null keys from upstream nodes. Such an approach scales well with large networks, but it makes a

critical assumption, that there is *enough path diversity* such that a malicious forwarder cannot know the dimensions that overlap with null keys at other forwarders. An attacker that knows which dimensions overlap can easily craft a polluted packet that passes a legitimate forwarder's verification test with high probability or even 1 according to Lemma 3.2.3. Such path diversity cannot be guaranteed in wireless networks.

Given the lack of path diversity in wireless network, we cannot rely on forwarders to create new null keys for downstream nodes. At the same time, the size of a null key is large preventing a source from sending individual keys to each forwarder. Our scheme, Split Null Keys (SNK), is based on the observation that only a small portion of a null key for a generation is dependent on the data for that generation while the remaining, larger portion of the null key is chosen randomly. The large, random portion serves as a secret between the source and forwarder, so keeping this portion constant across multiple generations does not deteriorate security as long as it remains secret. SNK splits a null key in two components: a component that is generation independent which is sent only at system initialization, and a component that is generation dependent and sent every generation. Specifically, for each null key, the generation dependent component has a size of $\omega * n * \log_2(q)$ bits, and the generation independent has a size of $\omega * m * \log_2(q)$ bits. Thus, every generation, for a forwarder, our scheme needs to send only $\omega * n * \log_2(q)$ bits, while if the scheme [24] is used, the entire null key of size at least $(n + m) * \log_2(q)$ bits needs to be updated (null keys for this scheme are sometimes larger based on the topology). The source generates and distributes the null keys for each forwarder, in a secure manner[2], so SNK does not rely on path diversity of the network topology.

At a high level, SNK works as follows (see also Algorithm 2). In the initialization step which is performed only once, the source creates and distributes the generation independent null keys to forwarders. In the update step which is performed every

[2]Note that nodes should share symmetric keys with the source in order to have end-to-end data integrity and confidentiality; a basic service for any communication protocol.

Algorithm 2 SNK

Initialization (generation independent): Source initializes a network with forwarders $f_1, ..., f_\theta$

1: Randomly select null key generators $\mathbf{G}_1, ..., \mathbf{G}_\theta$

2: Calculate $\bar{\mathbf{K}}_i = \mathbf{G}_i$ for $i = 1, ..., \theta$

3: Distribute $\bar{\mathbf{K}}_i$ to forwarder i for $i = 1, ..., \theta$

Null key update (generation dependent): Source generates update keys for a generation consisting of data matrix $\boldsymbol{X}$ *for a flow with forwarders* $f_1, ..., f_\beta$

1: Calculate $\tilde{\mathbf{K}}_i = -\mathbf{X} * \mathbf{G}_i$ for $i = f_1, ..., f_\beta$

2: Distribute $\tilde{\mathbf{K}}_i$ to forwarder i for $i = f_1, ..., f_\beta$

Verification (per packet): Forwarder f *verifies a coded packet* $\boldsymbol{c}$

1: Form null key $\mathbf{K}_f$ from $\tilde{\mathbf{K}}_f$ and $\bar{\mathbf{K}}_f$, $\mathbf{K}_f = \begin{bmatrix} \tilde{\mathbf{K}}_f \\ \bar{\mathbf{K}}_f \end{bmatrix}$

2: Verify that $\mathbf{c} * \mathbf{K}_f = \mathbf{0}$

generation, the source calculates and distributes the generation dependent null keys for a new generation represented by the data matrix $\mathbf{X}$ for each forwarder in the flow. In the verifying step which is performed every time a packet is received, a forwarder forms its null key from the received null key parts and verifies a received coded packet.

3.2.3 Null Keys Splitting Procedure

We split each null key into two parts $\mathbf{K}_i = \begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix}$, a *generation dependent null key* ($\tilde{\mathbf{K}}_i$) and a *generation independent null key* ($\bar{\mathbf{K}}_i$). The first n rows of the null key are the generation dependent portion, while the remaining m rows are the generation independent portion. Generation independent null keys are updated once for multiple generations while generation dependent null keys are updated every generation. As

in a typical network coding system $n \ll m$, ensuring that the generation dependent portion of a null key has n rows reduces the overhead significantly.

In order to ensure that the generation independent null key component remains constant across multiple generations we split $\mathbf{B}$ as follows. Let $\mathbf{S}$ be an n by m matrix and $\mathbf{T}$ be an m by m matrix such that $\mathbf{B} = \begin{bmatrix} \mathbf{S} \\ \mathbf{T} \end{bmatrix}$. The source keeps $\mathbf{T}$ constant for each generation and computes a new $\mathbf{S}$ in order to satisfy the null space property that $\mathbf{A} * \mathbf{B} = \mathbf{0}$. Let $\mathbf{T} = \mathbf{I}$ for each generation:

$$\begin{aligned} \mathbf{A} * \mathbf{B} = \mathbf{0} \;\Rightarrow\; [\mathbf{I}|\mathbf{X}] * \begin{bmatrix} \mathbf{S} \\ \mathbf{T} \end{bmatrix} = \mathbf{0} \;\Rightarrow\; \mathbf{I} * \mathbf{S} + \mathbf{X} * \mathbf{T} = \mathbf{0} \\ \Rightarrow\; \mathbf{S} + \mathbf{X} * \mathbf{T} = \mathbf{0} \quad \Rightarrow\; \mathbf{S} = -\mathbf{X} * \mathbf{T} \\ \Rightarrow\; \mathbf{S} = -\mathbf{X} * \mathbf{I} \quad \Rightarrow\; \mathbf{S} = -\mathbf{X} \end{aligned}$$

Thus, by choosing $\mathbf{T} = \mathbf{I}$, we obtain $\mathbf{B} = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix}$.

A null key is a random subspace of the column space of $\mathbf{B}$. To ensure that a generation independent null key remains constant for multiple generations, a *null key generator* is selected for each null key, and the null key generator remains constant for multiple generations. The null key generator is a random matrix $\mathbf{G}_i$ of size m by ω with full column rank. A null key is computed as $\mathbf{K}_i = \mathbf{B} * \mathbf{G}_i$.

We show that if we choose $\mathbf{T} = \mathbf{I}$, which in turn means $\mathbf{B} = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix}$, then only the generation dependent null key is dependent on $\mathbf{X}$:

$$\begin{aligned} \begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix} &= \mathbf{K}_i = \mathbf{B} * \mathbf{G}_i = \begin{bmatrix} -\mathbf{X} \\ \mathbf{I} \end{bmatrix} * \mathbf{G}_i = \begin{bmatrix} -\mathbf{X} * \mathbf{G}_i \\ \mathbf{G}_i \end{bmatrix} \\ &\Rightarrow\; \tilde{\mathbf{K}}_i = -\mathbf{X} * \mathbf{G}_i \quad \text{and} \quad \bar{\mathbf{K}}_i = \mathbf{G}_i \end{aligned}$$

We summarize the splitting algorithm. For each forwarder i the source generates a random matrix $\mathbf{G}_i$ of size m by ω with full column rank. Then, the source computes the generation independent null key $\mathbf{K}_i$ as $\bar{\mathbf{K}}_i = \mathbf{G}_i$ and the generation dependent null key $\mathbf{K}_i$ computed for each generation with data $\mathbf{X}$ as $\tilde{\mathbf{K}}_i = -(\mathbf{X} * \mathbf{G}_i)$. Thus, the null key remains constant for multiple generations and the generation dependent

null key depends on the data matrix of each generation. Each forwarder recreates its null key as $\begin{bmatrix} \tilde{\mathbf{K}}_i \\ \bar{\mathbf{K}}_i \end{bmatrix} = \mathbf{K}_i$.

3.2.4 Null Key Distribution and Verification

Distribution. An adversary that knows the null keys for a legitimate node can form coded packets that pass the legitimate node's verification test. Thus, the source distributes null keys to each forwarder over confidential and authenticated channels. We justify the use of these secure channels as we will show in Section 3.4 that our approach incurs significantly less overhead than previous cryptographic approaches which do not require secure channels with forwarders. Each forwarder shares a unique symmetric key with the source which can be set up before distributing generation independent null keys. For each generation the source generates a *null key packet* with the contents $\langle i||GID||Enc(\tilde{\mathbf{K}}_i)||MAC(i||GID||Enc(\tilde{\mathbf{K}}_i))\rangle$ where GID is an identifier for the generation, $Enc()$ is a block cipher encryption such as AES [51] in CBC mode, and $MAC()$ is a message authentication code such as HMAC [52] with SHA-1. [53]. A null key packet is sent on a multi-hop best path from the source to each forwarder of the flow for each generation.

Motivation for encrypting $\tilde{\mathbf{K}}_i$. When distributing the generation dependent null key $\tilde{\mathbf{K}}_i$ from the source to a forwarder it is necessary to encrypt it such that no other forwarder can decrypt the value. If the generation dependent null key were not encrypted, then a subtle attack exists where an attacker can exploit the knowledge of $\tilde{\mathbf{K}}_i$ over many generations. Given that $\tilde{\mathbf{K}}_i$ is sent in the clear, the attacker obtains n unique equations in a system of m unknowns, $\tilde{\mathbf{K}}_i * \bar{\mathbf{K}}_i = -\mathbf{X}$, each generation. After $\lceil \frac{m}{n} \rceil$ generations, the attacker obtains enough equations to compute the value $\bar{\mathbf{K}}_i$. With the entire contents of a victim node's null key, the attacker can craft polluted coded packets that pass a victim node's verification test. However, it is easy to prevent this subtle attack by encrypting each $\tilde{\mathbf{K}}_i$, and we do this as part of our protocol.

Verification. Given that a forwarder i has null key $\mathbf{K}_i$, the forwarder verifies packet $\mathbf{c}$ by checking if $\mathbf{c} * \mathbf{K}_i = \mathbf{0}$. According to Lemma 3.2.1, a valid packet will pass verification. Without the knowledge of $\mathbf{K}_i$ or any dimensions of $\mathbf{K}_i$, an attacker can conduct an attack only by randomly generating polluted coded packets. The probability that such a random polluted packet does pass the verification test is negligible for typical wireless network coding settings and is given in Lemma 3.2.2. In our scheme, because the source (which is trusted by all nodes) is the only one generating null keys and because these null key components are disseminated in a secure manner, the attacker cannot gain any knowledge of $\mathbf{K}_i$ or any dimensions of $\mathbf{K}_i$ and thus cannot improve their probability of polluting node i.

3.3 Security Analysis

We assume a strong adversary that can overhear all communication in the network for multiple generations and compromise all forwarders in the network except a victim node that is the target of the attack.[3] We formalize the security of our scheme in the form of Game 1. If the attacker wins the game, the scheme is insecure as the attacker is able to craft a polluted packet that passes the target node's verification. The previous work on null keys [24] is insecure under our assumption since the adversary will win Game 1 because the knowledge of null keys upstream of the target node allows those nodes to pollute the target node.

Theorem 3.3.1 computes the probability that an adversary will win Game 1. The theorem states that if both the cipher used to encrypt the null keys and the PRG used to generate the null keys cannot be broken then the probability that an attacker will win the game is the probability that an attacker will just guess a coded packet that will pass verification. The probability of guessing is the value given by Lemma 3.2.2 which is negligible for typical parameters.

[3]We do not consider an adversary that attempts to modify null key packets sent to the target node from the source. We assume that the MAC that is attached to each null key packet is sufficient to protect against such modification. This can be formally shown with additional attack games and theorems, but we omit this for space considerations.

We prove Theorem 3.3.1 by using Lemma 3.3.1 and intermediate games. Lemma 3.3.1 proves that an adversary's ability to pollute a target node is not improved by knowing other nodes' null keys. This lemma is true despite the dimensions of the target node's null key possibly overlapping with dimensions of other nodes' null keys because an adversary cannot know how they overlap due to the fact that all null keys are generated independently and randomly. Game 2 describes a game similar to Game 1 with the exception that the adversary cannot break the PRG. Game 3 describes a game similar to Game 1 with the exception that the adversary can neither break the cipher nor the PRG. Games 4 and 5 describe what is means for an attacker to break the PRG and cipher respectively. We show that an adversary wins Game 3 with a negligible probability. If strong PRG and cipher are selected, Games 4 and 5 are won with negligible probability. Given that Game 3, Game 4, or Game 5 must be won to win Game 1, we can deduce that Game 1, the game the attacker plays against our SNK protocol, is won with negligible probability. This security analysis is inspired by and is similar to the one in [21].

Definition 3.3.1 *The advantage PA-Adv*$[\mathscr{A}, SNK]$ *of adversary* $\mathscr{A}$ *against SNK is the probability that* $\mathscr{A}$ *wins Game 1.*

Definition 3.3.2 *SNK is secure if for all polynomial time adversaries* $\mathscr{A}$*, the value PA-Adv*$[\mathscr{A}, SNK]$ *is negligible.*

Lemma 3.3.1 *Consider a Probabilistic Polynomial-Time (PPT) adversary* $\mathscr{A}_1$ *that knows the data for a generation* $\boldsymbol{X}$ *and must produce a polluted coded packet* $\boldsymbol{c}$ *such that* $\boldsymbol{K}_i * \boldsymbol{c} = \boldsymbol{0}$*. Given that* $\boldsymbol{G}_i$ *are chosen truly randomly for each forwarder,* $\mathscr{A}_1$ *gains no advantage if it has knowledge of* z *null keys* $\boldsymbol{K}_j$ *where* $j \neq i$ *and* z *is bounded by a polynomial.*

Proof Assume an adversary $\mathscr{A}_2$ is equivalent to $\mathscr{A}_1$ with the exception that $\mathscr{A}_2$ has knowledge of z null keys $\mathbf{K}_j$ where $j \neq i$. The adversary $\mathscr{A}_1$ has $\mathbf{X}$ and can generate

Game 1 Pollution attack game for SNK

Game between a challenger $\mathscr{C}$ and an adversary $\mathscr{A}$. Parameters are $(\theta, \Omega, E, R, q, n, m, \omega)$ where θ is the number of forwarders, Ω is the number of generations, E is a cipher, R is a PRG, and the other parameters are the same as in SNK. **Setup:**

1: $\mathscr{C}$ generates a random key k and a random seed s.

2: $\mathscr{C}$ computes $\mathbf{G}_l$ for each forwarder l using $R(s)$.

3: $\mathscr{C}$ computes $\bar{\mathbf{K}}_l = \mathbf{G}_l$ for each forwarder l.

4: $\mathscr{C}$ computes $\tilde{\mathbf{K}}_l(j) = -\mathbf{X}(j) * \mathbf{G}_l$ for each forwarder l and each generation j.

5: $\mathscr{C}$ chooses some target forwarder i.

Queries:

1: $\mathscr{A}$ can request the target forwarder i. $\mathscr{C}$ responds with i.

2: $\mathscr{A}$ can request the encrypted generation dependent key for a given generation j. $\mathscr{C}$ responds with $E_k(\tilde{\mathbf{K}}_i(j))$.

3: $\mathscr{A}$ can request the encrypted generation independent key. $\mathscr{C}$ responds with $E_k(\bar{\mathbf{K}}_i)$.

4: $\mathscr{A}$ can request the null key $\mathbf{K}_l(j)$ of any node l s.t. $l \neq i$ and any generation j. $\mathscr{C}$ responds with $\mathbf{K}_l(j)$.

5: $\mathscr{A}$ can request the data for generation j. $\mathscr{C}$ responds with $\mathbf{X}(j)$.

Output:

1: $\mathscr{A}$ must output a generation identifier j and coded packet $\mathbf{c} = [\mathbf{v}|\mathbf{x}]$. $\mathscr{A}$ wins the game if $\mathbf{v} * \mathbf{X}(j) \neq \mathbf{x}$ and $\mathbf{c} * \mathbf{K}_i(j) = \mathbf{0}$.

Game 2 Pollution attack game without a PRG

Intermediate game with parameters $(\theta, \Omega, E, q, n, m, \omega)$. This game is identical to Game 1 with the exception that the PRG R is replaced with a truly random bit generator. Step 2 of the setup is the only change.

Game 3 Pollution attack game without a PRG or cipher

Intermediate game with parameters $(\theta, \Omega, q, n, m, \omega)$. This game is identical to Game 3 with the exception that the cipher E is replaced with a theoretically secure one-time pad. Steps 2 and 3 of the queries are the only steps changed.

z null key generators $\mathbf{G}_j$'s where $j \neq i$. With these values, $\mathscr{A}_1$ can calculate z null keys $\mathbf{K}_j$ where $j \neq i$. Since $\mathscr{A}_1$ can calculate z null keys $\mathbf{K}_j$ where $j \neq i$ in polynomial time, $\mathscr{A}_1$ and $\mathscr{A}_2$ are equivalent. Thus, our initial assumption is false, and we must

Game 4 Cipher attack game

Game for a cipher E with parameters (t_1, t_2, Ω) between a challenger $\mathscr{C}$ and an adversary $\mathscr{B}_1$.

Setup:

1: $\mathscr{C}$ generates a random key k.

2: $\mathscr{C}$ generates random t_1-bit messages M_i for $1 \leq i \leq \Omega$.

3: $\mathscr{C}$ generates a random t_2-bit message $\hat{M}$.

Queries:

1: $\mathscr{B}_1$ can request $E_k(M_i)$ for some i. $\mathscr{C}$ responds with $E_k(M_i)$.

2: $\mathscr{B}_1$ can request $E_k(\hat{M})$. $\mathscr{C}$ responds with $E_k(\hat{M})$.

Output:

1: $\mathscr{B}_1$ must output i, M', and M''. $\mathscr{B}_1$ wins the game if $M_i = M'$ and $\hat{M} = M''$.

Game 5 PRG attack game

Game for a PRG R with parameters (t, θ) between a challenger $\mathscr{C}$ and an adversary $\mathscr{B}_2$.

Setup:

1: $\mathscr{C}$ generates a random seed s.

2: $\mathscr{C}$ computes $t * \theta$-bit message M from $R(s)$.

3: $\mathscr{C}$ groups splits M into sets of t-bit messages $M_1, M_2, ..., M_\theta$.

4: $\mathscr{C}$ chooses an i such that $1 \leq i \leq \theta$.

Queries:

1: $\mathscr{B}_2$ can request i. $\mathscr{C}$ responds with i.

2: $\mathscr{B}_2$ can request M_j s.t. $j \neq i$. $\mathscr{C}$ responds with M_j.

Output:

1: $\mathscr{B}_2$ must output M'. $\mathscr{B}_2$ wins the game if $M' = M_i$.

accept that no adversary $\mathscr{A}_2$ obtains an advantage over $\mathscr{A}_1$ by having knowledge of z null keys. ■

Definition 3.3.3 *The advantage C-Adv$[\mathscr{B}_1, E]$ of adversary $\mathscr{B}_1$ against E is the probability that $\mathscr{B}_1$ wins Game 4.*

Definition 3.3.4 *The advantage PRG-Adv$[\mathscr{B}_2, R]$ of adversary $\mathscr{B}_2$ against R is the probability that $\mathscr{B}_2$ wins Game 5.*

Theorem 3.3.1 *SNK with parameters $(\theta, \Omega, E, R, q, n, m, \omega)$ is secure as long as the cipher E with parameters $(\omega * n * log_2(q), \omega * n * log_2(q), \Omega)$ is a secure cipher and the PRG R with parameters $(n * m * log_2(q), \theta)$ is a secure PRG. More specifically, for any $\mathscr{A}$ there is a cipher adversary $\mathscr{B}_1$ and a PRG adversary $\mathscr{B}_2$ such that*

$$PA\text{-}Adv[\mathscr{A}, SNK] \leq C\text{-}Adv[\mathscr{B}_1, E] + PRG\text{-}Adv[\mathscr{B}_2, R] + \left(\frac{1}{q}\right)^{\omega}$$

Proof We prove the theorem using Game 1, Game 2, and Game 3 defined above. For $i = 1, 2, 3$ let W_i be the events that $\mathscr{A}$ wins the pollution defense game in Game i.

$$P[W_1] = \text{PA-Adv}[\mathscr{A}, SNK] \tag{3.1}$$

In Game 2, we replace R in SNK with a truly random generator. Thus, every $\mathbf{G}_i$ is truly random instead of being generated with a PRG. The rest is the same as Game 1.

$$|P[W_1] - P[W_2]| = \text{PRG-Adv}[\mathscr{B}_1, R] \tag{3.2}$$

In Game 3, we replace the cipher E in SNK with a truly random generator. Thus, every response to queries 2 and 3 are random instead of encrypted null keys. Everything else is the same as Game 2.

$$|P[W_2] - P[W_3]| = \text{C-Adv}[\mathscr{B}_2, E] \tag{3.3}$$

In Game 3, the adversary learns nothing by querying the challenger for encrypted versions of i's generation dependent or independent null keys. The adversary cannot infer the value of $\mathbf{G}_i$ from the values of $\mathbf{G}_l$ for $i \neq l$ (note that $\mathbf{G}_l$ is part of each $\mathbf{K}_l(j)$) because these values are truly random along with the result of Lemma 3.3.1. We show that $P[W_3] = (\frac{1}{q})^{\omega}$. To win adversary must choose a $\mathbf{c}$ and j such that:

$$\begin{aligned}
\mathbf{c} * \mathbf{K}_i(j) = \mathbf{0} \quad &\Rightarrow \quad [\mathbf{v}|\mathbf{x}] * [\tilde{\mathbf{K}}_i^t(j)|\bar{\mathbf{K}}_i^t]^t = \mathbf{0} \\
&\Rightarrow \quad \mathbf{v} * \tilde{\mathbf{K}}_i(j) + \mathbf{x} * \bar{\mathbf{K}}_i = \mathbf{0} \\
&\Rightarrow \quad \mathbf{v} * (-\mathbf{X}(j) * \mathbf{G}_i) + \mathbf{x} * \mathbf{G}_i = \mathbf{0} \\
&\Rightarrow \quad \mathbf{x} * \mathbf{G}_i - \mathbf{v} * \mathbf{X}(j) * \mathbf{G}_i = \mathbf{0} \\
&\Rightarrow \quad (\mathbf{x} - \mathbf{v} * \mathbf{X}(j)) * \mathbf{G}_i = \mathbf{0}
\end{aligned}$$

Let $\mathbf{a} = \mathbf{x} - \mathbf{v} * \mathbf{X}(j)$. The adversary can freely choose any $\mathbf{a}$ by choosing an arbitrary $\mathbf{v}$, and then setting $\mathbf{x}$ to $\mathbf{a} + \mathbf{v} * \mathbf{X}(j)$. We have simplified the adversaries problem to choosing a $\mathbf{a}$ such that $\mathbf{a} * \mathbf{G}_i = 0$.

The trivial solution for choosing $\mathbf{a}$ is to let $\mathbf{a} = \mathbf{0}$, but this violates the $\mathbf{v} * \mathbf{X} \neq \mathbf{x}$ condition of winning the game. Thus, the adversary must choose an $\mathbf{a}$ such that $\mathbf{a} \neq \mathbf{0}$ and $\mathbf{a} * \mathbf{G}_i = \mathbf{0}$. Given that $\mathbf{G}_i$ is unknown to the adversary and is completely random in Game 3, any choice of $\mathbf{a}$ by the adversary will result in $\mathbf{a} * \mathbf{G}_i$ being a random vector of ω elements. Each element of this vector takes a random element from a field of q elements. Thus, the probability that $\mathbf{a} * \mathbf{G}_i = \mathbf{0}$ is $(\frac{1}{q})^\omega$ which is also the probability of winning Game 3.

$$P[W_3] = \left(\frac{1}{q}\right)^\omega \tag{3.4}$$

By combining Equations 3.1, 3.2, 3.3, and 3.4 we have:

$$\text{PA-Adv}[\mathscr{A}, SNK] \leq \text{C-Adv}[\mathscr{B}_1, E] + \text{PRG-Adv}[\mathscr{B}_2, R] + \left(\frac{1}{q}\right)^\omega$$

■

3.4 Evaluation

In this section, we compare the performance and overhead of SNK with other pollution defenses and a secure, store-and-forward routing protocol.

3.4.1 Simulation Methodology

Our experiments are conducted using the Glomosim [54] simulator with an implementation of the MORE [9] wireless network coding system. We use 802.11 [55] with a raw link bandwidth of 5.5 Mbps. For our topology we use the link quality measurements from Roofnet [49], a 38-node 802.11b/g mesh network. For each simulation, we set up a random flow in the network by selecting two random nodes as the source and the destination; the source transmits for 400 seconds. We select 200 random flows and conduct the simulation once for each flow and protocol.

Metrics. We measure *throughput* as the rate (in kbps) of data being decoded at the destination. We measure *latency* as the time between the start of the source transferring the first generation to decoding of the generation at the destination. We measure *communication overhead* as the total summed rate (in kbps) of overhead data broadcasted by all nodes. Data that does not belong to a standard network coding system is overhead data which are checksums, MACs, and null keys.

To demonstrate the efficacy of our scheme, we compare it with previous defenses against pollution attacks, all implemented in the MORE system. We compare SNK with the insecure MORE system, two representative cryptographic schemes KFM [16] and HOMOMAC [21], and two algebraic schemes DART [23] and EDART [23]. To show that SNK is practical, we also compare it with a secure traditional routing protocol, ARAN [50]. We do not compare with the scheme in [24] since as described in Section 3.2 such a scheme will not be secure in wireless networks.

We consider communication and computation overhead of each scheme. SNK sends generation dependent null key packets each generation, DART and EDART send checksum packets during generations, HOMOMAC appends MACs to coded packets, and KFM requires heavy computations.

Parameter selection. We select the network coding parameters to match the default settings for MORE in [9]. The size of a generation is $n = 32$, a symbol size of 1 byte $q = 2^8$, and the size of a coded packet is 1500 bytes. These parameters are the

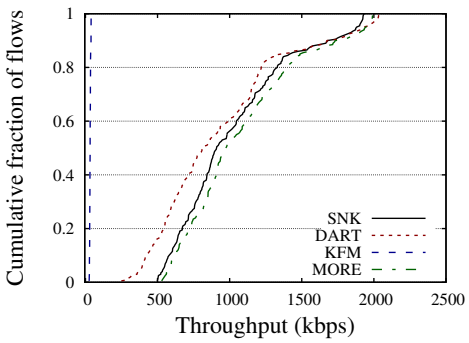

Figure 3.1.: Throughput of SNK, DART, KFM, and MORE.

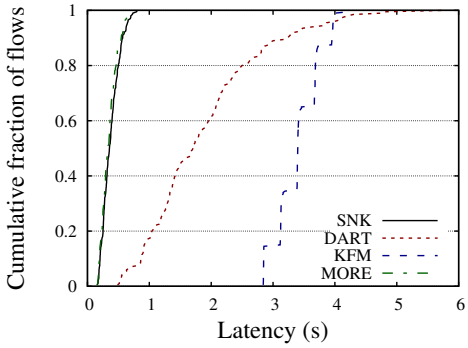

Figure 3.2.: Latency of SNK, DART, KFM, and MORE.

same for each scheme with the exception of KFM which requires a larger symbol size to ensure the intractability of the discrete logarithm problem. DART and EDART are configured to ensure their best performance, as in [23].

Attack settings. We select defense parameters for each defense scheme to ensure the same strength of $(\frac{1}{2})^{40}$ where the strength corresponds to the probability that the verification mechanism accepts a polluted coded packet. The rank of null keys, size of checksums, and number of MACs are selected appropriately for SNK, DART/EDART, and HOMOMAC respectively. We cannot ensure such strength for the adaptive defense scheme EDART as it purposely forwards some coded packets without verifying to reduce the delay imposed by DART. A pollution attacker broadcasts a polluted coded packet for every 5 coded packets it receives. The polluted coded packets are generated randomly as there is no better strategy for selecting polluted coded packets for these schemes given that the underlying security assumptions hold.

3.4.2 Performance Evaluation

We compare with two other proactive defenses KFM and DART. The proactive schemes verify every coded packet independent of the number of forwarders, so their overhead is the same in adversarial and benign networks. We include the insecure system MORE as a baseline for comparison.

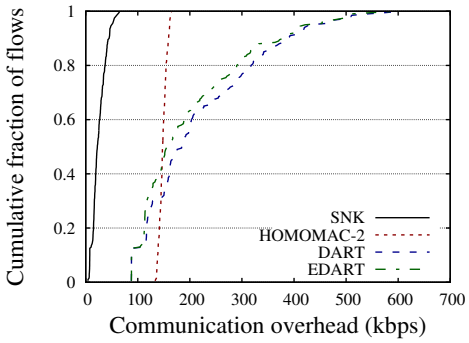

Figure 3.3.: Communication overhead of SNK, HOMOMAC-2, DART, and EDART.

From the results shown in Figure 3.1, SNK outperforms DART by over 100 kbps in the lowest 50% of flows (according to throughput) due to the lowest flows having larger number of hops from the source to destination. This pattern shows that DART's performance diminishes more than SNK as more hops exist between the source and destination due to the delaying of each coded packet for verification. Despite the similar throughput of SNK and DART, Figure 3.2 shows that DART imposes 5 times the latency compared to SNK. The increased latency of DART is due to the pipelining of 5 generations that is necessary to mitigate the delayed verification of packets and achieve high throughput. KFM only maintains roughly 50 kbps for all flows since it suffers from large computational overhead like many homomorphic signature schemes.

3.4.3 Scalability with Multiple Adversaries

We compare SNK with EDART and HOMOMAC whose performance depends on the number of adversaries.

Comparison with EDART. DART delays every packet to wait for the checksum that verifies that packet, and EDART differs by forwarding packets before they are verified. As a result, some polluted packets travel multiple hops causing more damage, and EDART responds by forcing affected nodes to delay packets for verification. Unlike EDART, our scheme, SNK, verifies all packets without delaying them.

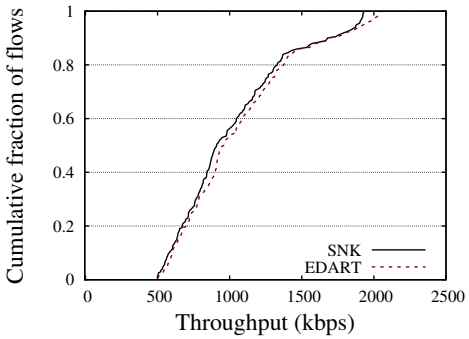

Figure 3.4.: Throughput of SNK and EDART with 0 attackers.

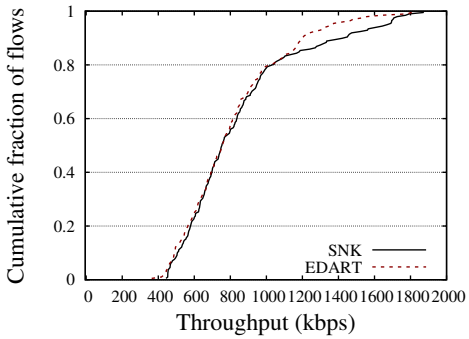

Figure 3.5.: Throughput of SNK and EDART with 5 attackers.

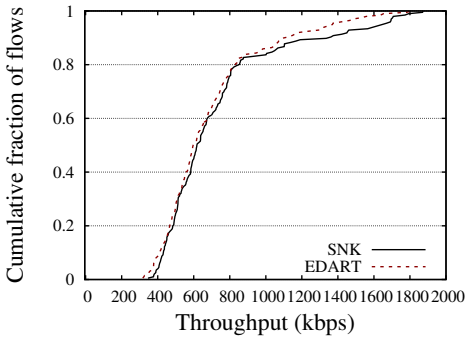

Figure 3.6.: Throughput of SNK and EDART with 10 attackers.

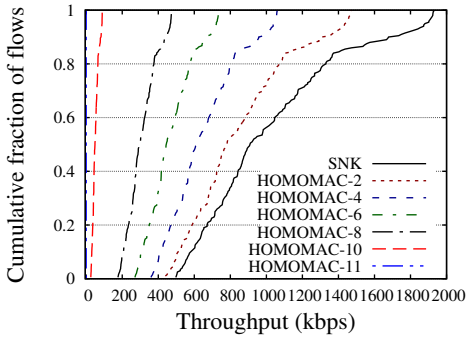

Figure 3.7.: Throughput of SNK which provides defense against any number of adversaries and HOMOMAC-x that is configured to provide defense against x adveraries.

Therefore the performance of SNK relative to the performance of EDART improves when attackers are present.

The performance of SNK and EDART with varying numbers of attackers are shown in Figures 3.4, 3.5, and 3.6. EDART outperforms SNK slightly in the benign scenario because packets are forwarded without being delayed for verification (verification is done later when a valid checksum is received), while SNK always verifies every packet. When attackers are present, SNK outperforms EDART which is most visible in the top 15% of flows where the difference ranges from 100 kbps to 300 kbps. These two schemes perform similarly for the rest of the 85% of flows with SNK gaining

relative throughput to EDART as the number of attackers increases. Averaged over all flows, the increases in throughput of SNK relative to EDART are 4.8% and 6.2% for cases of 5 and 10 attackers respectively. As the number of attackers increases, the EDART scheme will have a lower performance because it may either delay coded packets to verify them or allow polluted coded packets to be forwarded.

Comparison with HOMOMAC. We use HOMOMAC-x to denote the HOMOMAC scheme configured to defend against x adversaries. HOMOMAC utilizes redundant MACs and a special key distribution to ensure that an adversary cannot forge a coded packet. To remain resilient, the number of MACs per coded packet must increase as the number of colluding adversaries increases. We configure HOMOMAC to be resilient to varying numbers of colluding adversaries. In Figure 3.7, we compare the different HOMOMAC variants with SNK which is resilient to any number of adversaries. The severe degradation in performance is due to the increased communication cost of appending MACs to each packet, keeping in mind the number of MACs increases quadratically with respect to the number of colluding adversaries.

3.4.4 SNK vs. Traditional Secure Routing

We showed that SNK outperforms other pollution defenses in a network coding system. However, for a secure scheme to be practical it must preserve network coding benefits. In other words, the secure network coding scheme should still have better performance than a secure traditional store-and-forward routing protocol. To demonstrate that SNK is a practical defense, we compare it with ARAN, a secure version of the well-known AODV wireless routing protocol which signs packets to ensure packets modified by routers are dropped.

Figure 3.8 shows that SNK retains most of the throughput of MORE. The throughput of SNK is roughly 50 kbps lower than MORE in all flows, and the degradation is consistent among all flows due to consistent overhead in distributing null keys. SNK outperforms ARAN in nearly the same fraction of flows that MORE outperforms

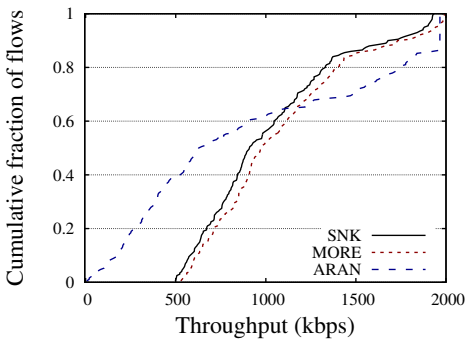

Figure 3.8.: Throughput of SNK, MORE, and ARAN.

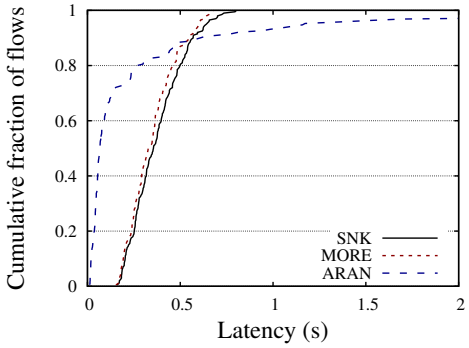

Figure 3.9.: Latency of SNK, MORE, and ARAN.

ARAN, 65%, and this is due to the advantages of network coding. The 35% of flows where ARAN outperforms MORE and SNK are flows that have few hops, and few network coding advantages exist. The latency of network coding systems is generally higher than traditional routing because an entire generation is transferred before the first byte of data is decoded at the destination. SNK only imposes up to 10 ms of additional latency over MORE as seen in Figure 3.9. For 90% of flows, network coding imposes higher latency on the network. However, at the highest 10% of flows, the latency of ARAN is significantly higher due to the fact that shortest path routing suffers in flows with long paths in wireless mesh networks.

3.4.5 Overhead Results

We further evaluate the overhead of the protocols with better performance. We compare the overhead of SNK, with DART, EDART, and HOMOMAC. We did not include KFM in the overhead comparison because its low performance (Figure 3.1 and Figure 3.2) does not make it a good candidate for a pollution defense in wireless networks.

Communication overhead. Figure 3.3 presents the communication overhead for SNK, DART, EDART, and HOMOMAC. The median communication overhead of SNK is 25 kbps which is an insignificant amount given the median throughput rate of

900 kbps for SNK. The other pollution defenses have larger communication overheads. HOMOMAC has a consistent communication overhead between 130-170 kbps for all flows. DART and EDART have lower communication overhead than HOMOMAC in the lowest 40% of flows, but DART and EDART have communication overhead as high as 600 kbps in the highest flows. The large variations in overhead for DART and EDART are a result of the variations in the number of forwarders in each flow, and DART and EDART periodically disseminate checksums to all forwarders.

Computation overhead. We measure the computation overhead that takes place at the source to generate null keys, checksums, or MACs and at the forwarders to verify incoming coded packets. On average in our topology, a flow has 4.57 forwarders, so SNK's time is 4.57 multiplied by the time to create a null key packet. A null key packet requires the creation of one generation dependent null key, encrypting it with AES, and computing an HMAC of the packet with SHA-1. For DART/EDART at least one checksum packet is required per generation due to the checksum interval of 32 and $n = 32$, so we give DART/EDART an advantage by only benchmarking for one checksum per generation. A checksum packet requires the creation of 5 checksums for the 5 pipelined generations and 1 RSA signature. HOMOMAC's time is for creating the required homomorphic MACs for each generation. Benchmarking results are average times of 1000 runs of a computation on a 2.4 Ghz processor with cryptographic computations from the OpenSSL library [56].

Figure 3.10 and 3.11 present computational overhead at a forwarder and source respectively for each scheme. Note that the computational overhead at the destination is comparable to that at a forwarder. In both cases, SNK imposes the least computational overhead while DART/EDART imposes slightly more computational overhead. Due to the generation and verification of multiple MACs, HOMOMAC requires 4 times the overhead of both SNK and DART/EDART.

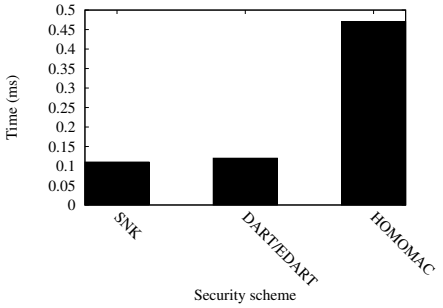

Figure 3.10.: Time for forwarder computation of verifying a packet for SNK, DART/EDART, and HOMOMAC-2.

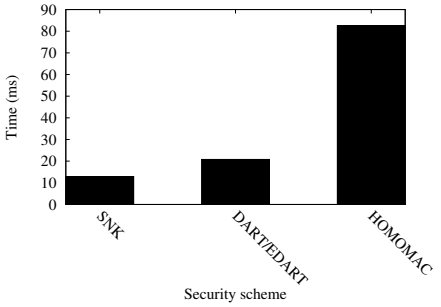

Figure 3.11.: Time for source computation of generating null keys, checksums, and homomorphic MACs for SNK, DART/EDART, and HOMOMAC-2 for a generation.

3.5 Summary

We present the Split Null Keys (SNK) protocol, a pollution defense for wireless network coding that relies on null space properties. We show that our scheme is secure against a strong adversary that overhears all communication and compromises multiple forwarders. We evaluate our defense through simulations in a typical network coding system scenario. Our evaluation shows that SNK maintains the network coding gains of MORE and outperforms previous pollution defenses.

4 ENTROPY ATTACKS AND COUNTERMEASURES IN WIRELESS NETWORK CODING

We now move onto entropy attacks against network coding which has been given less attention in the literature. The entropy attack enables a small set of malicious routers to significantly diminish the throughput of a network even if proper defenses are deployed against pollution attacks. The pollution attack defenses are useless against entropy attacks as the malicious packets are valid, they just happen to offer no useful coding information. We demonstrate the effectiveness of entropy attacks on a network while emphasizing that sophisticated attackers can perform such attacks in a significantly stealthy way. We propose defenses which work under simplistic forms of the attack and also intricate defenses which can also detect those sophisticated, stealthier versions of attacks.

A node creates *correct* coded packets by computing a random linear combination of packets stored in its *coding buffer*. The coding buffer is the set of coded packets received correctly. A receiver is able to eventually recover the original packets if it obtains n linearly-independent coded packets generated based on original plain packets from the source. A malicious node can deviate from the standard coding procedure and conduct two types of attacks that are specific to network coding systems. The first, well-known attack is called a pollution attack [27, 57] where a node creates an *invalid* coded packet which is not a valid combination of coded packets. The second, less studied attack is called an entropy attack [27, 28] where a node creates a *non-innovative* coded packet which is a non-random linear combination of coded packets such that the coded packet is *linearly dependent* with the coded packets stored at a downstream node. A linearly dependent coded packet wastes resources since it adds no useful information to help the receivers decode the original packets. We clas-

sify entropy attacks into two categories which require different capabilities from an attacker:

- A *local entropy attack* corresponds to an attacker that produces coded packets that are non-innovative to local neighboring nodes.
- A *global entropy attack* corresponds to a more capable attacker that produces coded packets that are seemingly innovative to local neighboring nodes but are non-innovative to at least one distant downstream node.

Many defenses for pollution attacks were proposed [15–19, 21, 23–27, 58–60], and they are all designed to defend against invalid coded packets but they provide no defense against attacks that use valid, but non-innovative coded packets. Cryptographic-based defenses [15–19, 21, 23, 24, 58] use homomorphic cryptography to detect and drop coded packets that are not valid linear combinations of the source data. Non-innovative packets will pass such verifications since they are valid combinations of the source data. Information theoretic defenses [25, 26, 59] rely on sending additional redundant information to correct invalid coded packets at the receiver. This additional information does not provide any benefit against an attacker that creates non-innovative packets because the added redundancy will only help recover the non-innovative packets. Lastly, existing monitoring defenses for pollution attacks [60, 61] focus on detecting invalid coded packets by comparing the packets received and sent by a node. While such schemes can detect simple types of entropy attacks in which the attacker for example sends the same valid packet repeatedly, they can not detect global entropy attacks.

Previous work [27, 28] showed that receivers waste resources to process non-innovative packets. However, entropy attacks cause more damage to a network than just occupying these resources. An entropy attack disrupts routing the same way selective forwarding attacks [62] disrupt routing in a network. In both cases, the routing algorithm chooses an optimal route or multiple routes to send data on, but the attacking node refuses to participate correctly in the forwarding of packets which

prevents information transfer along one or more paths. While the effect of the entropy and selective forwarding attacks are similar, defenses against selective forwarding attacks [62–64] can not be directly applied to entropy attacks because entropy attackers actually send packets and, in the case of global entropy attacks, the attack can not be locally detected and global information is needed. Multi-path defenses [62] rely on sending redundant information along multiple paths. Network coding routing inherently sends on multiple paths, but performance can still significantly be degraded by entropy attacks since a compromised node can still deny flow on a fraction of the paths. ACK-based defenses [63] are not applicable for entropy attacks since they require nodes to acknowledge that they have received packets and do not provide mechanisms to detect if those packets were innovative or not. Existing monitoring defenses [64] require that watchdog nodes receive a fraction of traffic in and out of a suspected node to make an accurate decision of misbehavior. The approach will not work for entropy attacks since the watchdog must receive all packets that the watched node receives, otherwise it cannot determine if the newly created packets are innovative.

In this chapter we study entropy attacks and their impact on wireless network coding systems. Specifically:

- We classify entropy attacks based on attacker's capabilities into *local entropy attacks* and *global entropy attacks*. We introduce a new attack, the *global entropy attack* in which the attacker generates coded packets that seem to be innovative to immediate neighbors, but are non-innovative for nodes that are further downstream. We demonstrate via simulation the negative impact of entropy attacks on network performance.

- We propose a defense scheme against local entropy attacks. The scheme, Non-innovative Link Adjustment (NLA), routes around attackers by adjusting the link quality for each link based on the percentage of non-innovative packets

they carry. We show that while NLA works well for local entropy attacks, is not effective for global entropy attacks.

- We propose two defenses to address global entropy attacks. In the first defense, Upstream Buffer Propagation (UBP), downstream nodes share information about received coded packets with upstream nodes such that immediate downstream neighbors of an attacker can detect the attack. In the second defense, Buffer Monitoring (BM), watchdog nodes monitor forwarder nodes to ensure that broadcast coded packets are random linear combinations of all received coded packets, and the coefficients of this linear combination are chosen according to a publicly known pseudo-random function. BM is essentially different from typical monitoring techniques since watchdogs need to know every packet received by a forwarder. The defenses differ in terms of detection efficacy and overhead cost.

- We analytically compare the security strength of the UBP and BM defense schemes. We are able to quantify the capabilities of an attacker under each global entropy defense.

- We use a real network topology to analyze the applicability of the BM monitoring-based global entropy defense since not every flow in a topology may have sufficient wireless links to monitor traffic. We find that for the real network topology we used the monitoring-based defense can be applied to only 84.7% of flows due to constraints of the topology.

This chapter is organized as follows. Section 4.1 defines the model we assume for this chapter. Section 4.3 describes local attacks and defenses while Section 4.4 describes global attacks and defenses. Section 4.5 analyzes our defenses emphasizing their effectiveness and limitations. Section 4.6 analyzes the imposed overhead of our defenses. Section 4.7 summarizes this chapter.

4.1 Entropy Attack Model

In this section, we describe the entropy attack model. The network coding system is the same as presented in the previous Chapter within Section 3.1.1. From the notation in Table 3.1, the following notation is re-used here: n packets per generation, m data symbols per packet, q field size of a symbol, $\mathbf{A}$ coding buffer, and $\mathbf{c}$ coded packet. The remaining notation in that table is specific to the pollution defense.

To formally define entropy attacks we present some additional notation for network coding operations which includes a time variable t. The source continuously broadcasts *coded packets* $\mathbf{c}(t)$ that are random linear combinations of the source's coding buffer $\mathbf{A}$:

$$\mathbf{c}(t) = \mathbf{r}(t) * \mathbf{A} \tag{4.1}$$

The vector $\mathbf{r}(t)$ is the random vector used to create $\mathbf{c}(t)$ at time t at the source. Each forwarder i has a coding buffer $\mathbf{A}_i(t)$ which is a matrix such that the first rows are the set of all overheard coded packets at node i that are linearly independent when the forwarder i broadcasts a coded packet at time t, and the rest of the rows are zero such that the total number of rows is n. The t component is necessary because the number of non-zero rows in the coding buffer grows over time as more coded packets are received. Forwarders have a condition that defines when to forward a packet. For example, in MORE [9] a forwarder will forward when it has received a sufficient (depending on the topology) number of coded packets. When this condition is met at time t for forwarder i, the forwarder generates and broadcasts the coded packet $\mathbf{c}_i(t)$:

$$\mathbf{c}_i(t) = \mathbf{r}_i(t) * \mathbf{A}_i(t) \tag{4.2}$$

The vector $\mathbf{r}_i(t)$ is the random vector used to create the coded packet $\mathbf{c}_i(t)$ at time t at node i.

We define two classes of entropy attacks, local and global. A global entropy attacker is capable of overhearing traffic on a link that is located several hops downstream. Such overhearing is possible if the attacker has a more advanced antenna for

reception or cooperates with another wireless device that is located near the link that must be eavesdropped. A global entropy attacker requires a much more sophisticated defense to deal with. We will motivate via simulation in Section 4.4 the need to create sophisticated defenses for detecting the most capable entropy attackers.

4.1.1 Local Entropy Attacks

A local entropy attacker creates coded packets that are non-innovative to neighboring nodes. Such an attacker creates non-innovative coded packets by refusing to code optimally as the optimal is creating a coded packets that is a random linear combinations of all received coded packets. Specifically, we define a local entropy attacker as a forwarder i that deviates from the protocol at some time t by creating a coded packet $\bar{\mathbf{c}}_i(t)$:

$$\bar{\mathbf{c}}_i(t) = \bar{\mathbf{r}}_i(t) * \mathbf{A}_i(t) \tag{4.3}$$

The vector $\bar{\mathbf{r}}_i(t)$ is an arbitrary vector chosen by the attacker. A random linear network coding protocol dictates that coded packets are combined randomly, but the attacker deviates by choosing a non-random vector $\bar{\mathbf{r}}_i(t)$. Specifically, a non-random vector is a vector such that at least one element is not chosen randomly, while a random vector has every element chosen randomly.

4.1.2 Global Entropy Attacks

A global entropy attacker uses global information about what coded packets have been sent in the network to create coded packets that are seemingly innovative to local nodes but are non-innovative to some distant downstream node. These coded packets are also created by refusing to create random linear combinations of received coded packets but also by including a combination of coded packets from some other portion of the network to deceptively cause local neighbors to believe the coded packet

is innovative. Specifically we define a global entropy attacker as a forwarder i that deviates from the protocol at time t by creating coded packets $\bar{\mathbf{c}}_i(t)$:

$$\bar{\mathbf{c}}_i(t) = \bar{\mathbf{r}}_i(t) * \mathbf{A}_i(t) + \mathbf{d}_i(t) \tag{4.4}$$

The vector $\bar{\mathbf{r}}_i(t)$ is not a random vector. The vector $\mathbf{d}_i(t)$ is not an element of the row space of $\mathbf{A}_i(t)$ but is an element of the row space of $\mathbf{A}$. That is, $\mathbf{d}_i(t)$ is a linear combination of some coded packets in the network, but it is not a linear combination of the coded packets that have been received by node i. The $\mathbf{d}_i(t)$ component of $\bar{\mathbf{c}}_i(t)$ is coded information being replayed from some other portion of the network.

4.2 Simulation Methodology

We aim to motivate the need for defenses against entropy attacks by showing the impact of entropy attacks through simulations. We conduct simulation experiments to measure the performance of a realistic system under various attack and defense scenarios.

We select MORE [9] as our wireless intra-flow network coding protocol that is based on random linear network coding. The source continuously broadcasts coded packets. A node is selected as a forwarder if it lies on a path or multiple paths from the source to destination, and these paths have sufficient link qualities. Based on global link state information, each forwarder is assigned a rebroadcast ratio which determines the number of coded packets received before creating and broadcasting a new coded packet. These ratios reflect how much each node contributes to a flow. Once the destination has received a sufficient number of coded packets for a generation, the destination sends an ACK back to the source. Upon receiving the ACK, the source starts sending a new generation.

Our experiments are conducted using the Glomosim [54] simulator. We use a raw link bandwidth of 5.5 Mbps and 802.11 [55] as the MAC layer protocol. For a realistic network topology and link qualities, we use the link quality measurements from the

Roofnet [49] network which is a 38-node 802.11b/g mesh network deployed on MIT campus.

An experiment consists of 200 simulations that each have a random flow. A random flow consists of a randomly chosen source and destination pair. In a given simulation, the source transfers data to the destination for 400 seconds. We measure performance for a simulation and display all 200 simulations as a Cumulative Distribution Function (CDF).

For performance, we measure throughput of the system. Throughput is the rate (in kbps) of data being decoded at the destination. More specifically, throughput is the total amount of data decoded at the destination (r bits) divided by the transfer time (T seconds):

$$Throughput = \frac{r}{1000 * T} \tag{4.5}$$

We select the network coding parameters to match the default settings for MORE as described in [9]. The number of rows in the matrix $\mathbf{A}$ is $n = 32$, a symbol size of 1 byte $q = 2^8$, and the size of a coded packet is 1500 bytes.

To simulate attackers, we define the specific coding behavior of attackers. Random nodes from the set of forwarders for a flow are selected as attackers, and those attackers follow the specified attack behavior.

4.3 Local Entropy Attacks

Using the methodology described in Section 4.2, we show the impact of a local entropy attack on a typical network, and then we present a defense strategy.

4.3.1 Attack Impact

The damage caused by a local entropy attack is similar to selective forwarding. A broadcast coded packet that is non-innovative to all neighboring nodes is equivalent to a node not broadcasting a coded packet at all. The local entropy attack does cause some additional damage compared to selective forwarding since bandwidth

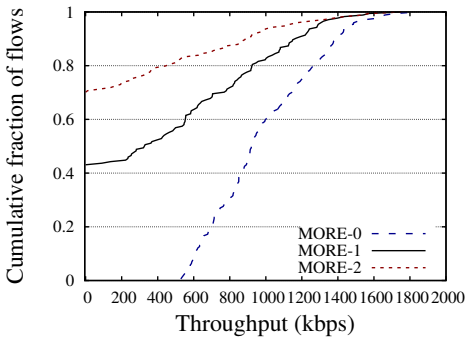

Figure 4.1.: Throughput of MORE with varying entropy attackers.

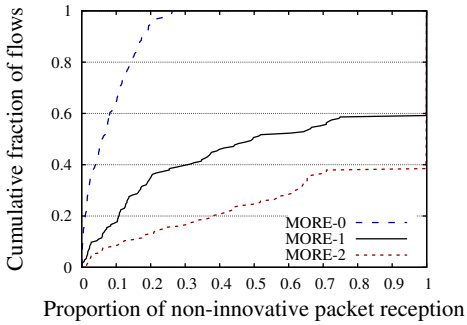

Figure 4.2.: Proportion non-innovative coded packet receptions of MORE for varying entropy attackers.

and computation of neighboring nodes is wasted to receive the non-innovative coded packets and determine that they are non-innovative.

A local entropy attack is effective because the system trusts each node to utilize the link capacities fully to deliver innovative coded packets downstream. Thus, when a local entropy attacker is located at an important position on a path or multiple paths between the source and destination, the system delivers many coded packets to the attacker under the false assumption that new innovative coded packets are delivered further downstream from this node.

To show the effectiveness of a local entropy attack, we conduct an experiment using the simulation methodology from Section 4.2 with zero, one, and two entropy attackers. These local entropy attackers choose $\bar{\mathbf{r}}_i(t) = \langle r_1, ..., r_{16}, 0, ..., 0 \rangle$, so the first 16 symbols are random while the last 16 symbols are zero. Thus, the attacker codes normally the first 16 received coded packets, but any further coded packets received are never used for coding.

Figure 4.1 shows the results of the local entropy attack. Such a simple attack results in zero throughput for 43% and 70% of flows for one and two attackers, respectively. The zero throughput flows are flows where the attackers happened to cut the topology consisting of the forwarders. Even the throughput in non-zero flows for the attacking scenarios degrade throughput significantly. Roughly 15% of the flows

in each case of attackers has non-zero throughput where the throughput is less than the lowest throughput for MORE without an attack. The non-zero flows are affected as well, as the median throughputs are 900, 400, and 0 kbps for 0, 1, and 2 attackers respectively.

4.3.2 Defense

The ideal defense against a local entropy attacker is to determine which nodes are performing the attack and remove them from the system. This is not straightforward based on local decisions since even honest nodes may unknowingly send some non-innovative coded packets. The reason is because a node knows what it has already sent, but it does not know what downstream nodes may have received from another path, and a downstream node may receive the same information along two upstream paths.

Figure 4.2 shows the proportion of received non-innovative coded packets in each flow for MORE. There is an obvious increase in non-innovative coded packet reception with attackers present, but even for the benign case some flows will contain a significant proportion of non-innovative coded packets. In these benign flows, there are multiple paths to the destination which have high packet reception probabilities, thus there is little diversity in the coded packets downstream when these paths converage which accounts for the non-innovative coded packet receptions. For 30% of benign flows 10% of received coded packets were non-innovative. This is a total for all nodes in the network, so some links in the flow potentially carry an even larger proportion of non-innovative coded packets.

Non-innovative Link Adjustment (NLA). We propose NLA as a defense against local entropy attacks. Because honest nodes also create some non-innovative packets, we do not adopt a strict node removal strategy. Instead, we punish each link proportionally with the amount of non-innovative coded packets sent on the link. The modified link qualities are obtained by multiplying the original link quality with

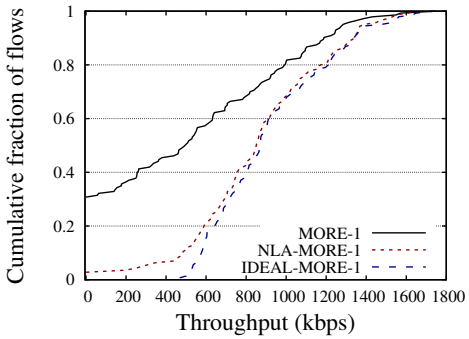

Figure 4.3.: Throughput of MORE, MORE-NLA (entropy defense), and the ideal entropy defense.

the proportion of received innovative coded packets to total coded packets on a link. A forwarder notifies the network if it notices a significant change in the modified link qualities. Thus, when a local entropy attacker sends non-innovative coded packets, the routing layer is alerted that specific links are not carrying innovative coded packets. For MORE, nodes recalculate their rebroadcast ratios based on the modified link states which will route data on paths that avoid the attacker node. Performing this securely requires mechanisms that limit the ability of an attacker to falsely accuse other nodes. We assume that such mechanisms are in place and they are out of the scope of this dissertation.

Figure 4.3 demonstrates how the local entropy attack is mitigated by NLA. As a baseline for the defense, we include an ideal defense where the entropy attacker is removed from the network. We removed flows (31 of 200) where the entropy attacker partitions the network from source to destination as there does not exist any set of forwarders that provides positive throughput. Without a defense, 30% of flows result in zero throughput, and these are cases where the entropy attacker partitions the set of forwarders chosen by the routing logic. With NLA, only 5% of flows result in zero throughput because in the majority of cases where the entropy attacker partitions the initial set of forwarders, the NLA defense severely punishes the links outgoing from the malicious node such that the routing logic chooses a new set of forwarders which can route data around the malicious node. With the exception of the lowest

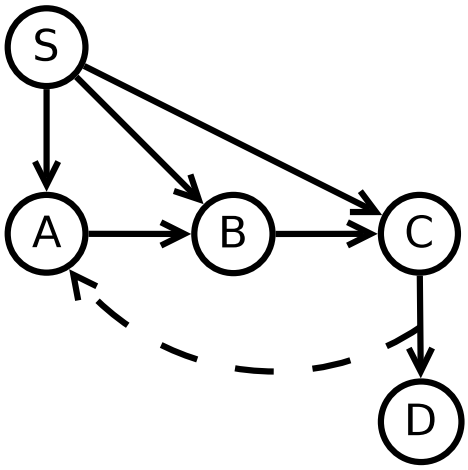

Figure 4.4.: Topology for global entropy attack scenario.

performing 20% of flows, the NLA is capable of performing within 50 kbps of the cases where the local entropy attacker is removed.

4.4 Global Entropy Attacks

In this section we first show how a global entropy attack impacts performance and argue that the NLA defense cannot mitigate such an attack. We then propose two defenses that can defend against global entropy attacks.

4.4.1 Attack Impact

We show in Figure 4.4 a specific global entropy attack example. A malicious node A is able to perform a global entropy attack that A's downstream neighbor B cannot detect. The malicious node A sets $\bar{\mathbf{r}}_A(t) = \mathbf{0}$ and $\mathbf{d}_A(t) = \mathbf{r}_A(t) * \mathbf{A}_Y(t)$ where $\mathbf{A}_Y(t)$ is a coded buffer created from packets that have been overhead from the link between C and D. With this setting, Equation 4.4 becomes:

$$\bar{\mathbf{c}}_A(t) \;=\; \bar{\mathbf{r}}_A(t) * \mathbf{A}_X(t) + \mathbf{d}_A(t) = \mathbf{r}_A(t) * \mathbf{A}_Y(t)$$

Node A has access to coded packets sent by C to D via some out-of-band channel as shown by the dashed line in Figure 4.4, which allows A to create $\mathbf{A}_Y(t)$ (this can

Table 4.1: Throughput results of various attack and defense scenarios with the topology in Figure 4.4

Defense	Throughput (kbps)
None	233
NLA	214
Ideal	345

be done as simple as overhearing, or by colluding with D). Coded packets broadcast by A are linear combinations of coded packets that are broadcast by C which include coded packets that C received directly from S. B has no knowledge of coded packets that S broadcasts, C receives, and B fails to receive. So, B will receive coded packets that are innovative to B's coding buffer from A, but these coded packets are not innovative to C's coding buffer. Thus, when C receives linearly dependent coded packets from B, C cannot be sure if B or A is the entropy attacker given only local information.

The global entropy attack focuses mainly on being stealthy, but it is still damaging and can reduce throughput significantly like the local entropy attack. We performed simulations with the topology from Figure 4.4. In this figure, S is the source, D is the destination, A is the global entropy attacker, and the solid lines indicate standard wireless links, while the dashed line indicates that node A can overhear the coded packets on the link between C and D. High link qualities are used for edges $(S, A), (A, B), (B, C), (C, D)$, and we used lower link qualities for edges (S, B) and (S, C). Thus, the network assumes many packets are routed through the path S, A, B, C, D. We measured a throughput of 769 kbps when A is honest and 233 kbps when A is a global attacker by replaying combinations of coded packets from link (C, D). Thus, node A harms system performance, and the node still sends coded packets that are innovative to local neighbors but are not innovative to neighbors further downstream.

We apply NLA to the scenario in Figure 4.4 to show how it fails to prevent the entropy attack. As summarized in Table 4.1, the throughput is 214 kbps when NLA is applied and 345 kbps when the ideal defense is applied. The ideal defense is the case where the global entropy attacker is removed from the network. NLA actually has lower throughput compared to the case of no defense which has 233 kbps because the modified link quality of (B, C) is lowered to nearly zero and thus the network only utilizes the path S, C, D while the path S, B, C, D still provides some innovative coded packets even under a global entropy attack. This topology illustrates how a global entropy can reduce throughput by roughly 30%, but in other topologies a global entropy attack has the potential to cause greater damage. We conclude that NLA cannot defend against the global entropy attack.

4.4.2 Global Entropy Defenses Overview

We present two global entropy defenses. In the Upstream Buffer Propagation (UBP) defense, nodes propagate buffer information upstream to pinpoint the origin of the global entropy attack. In the Buffer Monitoring (BM) defense, nodes monitor incoming and outgoing traffic of untrusted, neighbor nodes to immediately detect any coded packets created in a non-random fashion. To contrast these two techniques, UBP is more reactive and thus has lower overhead while BM is more proactive and requires higher overhead, placing constraints on the topology. These schemes require the following three wireless communication primitives:

- **Broadcast.** A message is broadcast once and neighboring nodes will receive the message probabilistically.
- **Reliable unicast.** A specific neighboring node is designated and the sender will repeatedly broadcast a message until the receiving node acknowledges the reception of the message. This primitive can be the standard 802.11 unicast.

- **Reliable multicast.** A set of neighboring nodes are designated and the sender will repeatedly broadcast a message until all receiving nodes acknowledge the reception of the message. This primitive does not exist naturally in the 802.11 protocol, but it can be realized efficiently by periodically broadcasting the message until an acknowledgement is received from each of the designated receivers. As this primitive is not standard, we analyze its overhead in Section 4.6.2.

Both defenses propose a mechanism for nodes to make an accurate accusation that another node is a global entropy attacker. A complete solution requires an appropriate response to such an accusation which is not straightforward since an accuser may be malicious. This is a general problem for many security protocols, and it is out of the scope of this dissertation as we can apply approaches from other work that resolve this issue. One approach from a work on a secure wireless multicast protocol [65] proposes to only remove accused nodes temporarily and limit the accusations of a node. Thus, a malicious accusation is not permanently damaging, and a malicious node cannot disrupt the network by accusing many nodes. Another approach is to use a reputation system [66, 67] to lower the reputation of nodes that have been accused or have made invalid accusations. With these reputation values, a node with low reputation can be removed from the network and its accusations can be ignored as well.

4.4.3 Upstream Buffer Propagation

The defense is initiated by the reception of non-innovative coded packets which implies that a global entropy attack is upstream. The entropy attacker may reside along any upstream path. We can determine the entropy attacker by propagating buffer information upstream until it reaches the source of the global entropy attacker. Thus, a legitimate node i, in response to receiving non-innovative coded packets, creates a packet containing its buffer information and sends this packet upstream. When this buffer information reaches the entropy attacker, the entropy attacker has

a choice to continue performing the entropy attack and be detected, or to start sending innovative coded packets. Either way, the entropy attack is mitigated.

There are two challenges to reducing the overhead of propagating buffer information upstream to make it practical. The buffer information consists of the coding headers at a node, and as a node receives more coded packets the total size of all coding headers at the node becomes large. Thus, our first challenge is to send a small constant-sized message that conveys to upstream nodes the contents of the buffer, and we do this with a special type of checksum. Our second challenge is to prevent flooding the message upstream and instead choose based on local information at each hop the upstream path that the attack is on.

Null Space Checksums

The buffer checksums utilize the *null space* of the vector space spanned by coding headers which is the space of all vectors that result in a zero when multiplied by a vector from the vector space spanned by the coding headers. We provide background on null spaces and then we explain how to use null spaces to provide a checksum for coding headers which is represented by a *coding header matrix* $\mathbf{V}_i(t)$ which is a matrix of the nonzero coding headers of $\mathbf{A}_i(t)$. We denote these checksums as *null space checksums*.

Consider a vector space A and a null space $N(A)$ of A. Given any two vectors $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{x} \in A$ and $\mathbf{y} \in N(A)$, we have:

$$\mathbf{x} * \mathbf{y}^T = 0 \tag{4.6}$$

The notation $\mathbf{w}^T$ is the transpose of the matrix or vector $\mathbf{w}$.

For our checksum, we consider A to be the row space of $\mathbf{V}_i(t)$ which has $R(\mathbf{V}_i(t))$ rows (we denote $R(\mathbf{X})$ as a function that returns the number of rows of the matrix $\mathbf{X}$). The vector of the null space is a vector $\mathbf{s}_i(t)^T$ such that:

$$\mathbf{V}_i(t) * \mathbf{s}_i(t)^T = \mathbf{0} \tag{4.7}$$

This corresponds to a linear system of $R(\mathbf{V}_i(t))$ equations and n unknowns. To find a valid $\mathbf{s}_i(t)^T$, we simply fill in $n - R(\mathbf{V}_i(t))$ symbols randomly, and we are left with $R(\mathbf{V}_i(t))$ equations and $R(\mathbf{V}_i(t))$ unknowns which is solved to fill in the remaining symbols. Thus, computing this vector is computationally inexpensive.

Given $\mathbf{s}_i(t)^T$ and the coding header matrix of another node j at a time t', $\mathbf{V}_j(t')$, we have the following probability:

$$Pr\left(\mathbf{V}_j(t') * \mathbf{s}_i(t)^T = \mathbf{0}\right) = \left(\frac{1}{q}\right)^d \tag{4.8}$$

The variable d is the number of rows of $\mathbf{V}_j(t')$ are linearly independent with all rows of $\mathbf{V}_i(t)$, and the variable q is the cardinality of the field that each symbols lies in (e.g., $q = 256$ for network coding over 1 byte symbols). Any row of $\mathbf{V}_j(t')$ that is linearly dependent with all rows of $\mathbf{V}_i(t)$ results in a zero if multiplied by the vector $\mathbf{s}_i(t)^T$. Any row of $\mathbf{V}_j(t')$ that is linearly independent with all rows of $\mathbf{V}_i(t)$ results in a random symbol when multiplied by the vector $\mathbf{s}_i(t)^T$. Thus, d symbols are random, and the probability that all d symbols are zero is $(\frac{1}{q})^d$.

Thus, a node j that is upstream from node i knows that if $\mathbf{V}_j(t') * \mathbf{s}_i(t) = \mathbf{0}$ then node j at time t' most likely has no innovative packets with respect to node i's coding buffer at time t. Also, if node j finds that $\mathbf{V}_j(t') * \mathbf{s}_i(t) \neq \mathbf{0}$ then j can definitely create coded packets that are innovative with respect to node i's coding buffer. The null space checksums are encapsulated in a Buffer Checksum Packet (BCP) which contains additional information required by the protocol. We show in Section 4.6.1 that computational overhead is low to generate and check these BCPs.

Single Path Propagation

Instead of sending the buffer information on all possible paths upstream, it only needs to be sent along one single path. The path can be determined by local decisions while ensuring with high probability that the global entropy attacker will be part of the path. A node attaches a sequence number to every coded packet it creates

and forwards, and nodes maintain some state that allows them to determine which upstream neighbor sent the last coded packet that triggered the broadcast of a new coded packet.

To determine a single upstream path for buffer propagation, each node maintains both a sequence number and a Sequence Number Table (SNT). When a node j broadcasts a coded packet, it appends its sequence number u_j to the coded packet and then increments u_j by one. Upon receiving from upstream neighbor i a coded packet that has a sequence number u_i, node j adds an entry $\langle u_i, j, u_j \rangle$ to its SNT and removes any old entries with the same u_i.

A node j receives a BCP because it had broadcast a coded packet that was globally non-innovative which triggered the propagation of this BCP at a downstream node that may be several hops downstream. Based on its SNT, node j knows the upstream neighbor i that sent the last coded packet which was used to create the globally non-innovative coded packet at node j. Thus, the attacker is either node i or some node upstream of node i which caused i to send this packet that is globally non-innovative. The BCP is forwarded upstream to node i along with the sequence number of the coded packet that i created so that, if node i is honest, it can make an accurate decision about which upstream node to use for BCP.

Protocol Description

We now describe in Algorithm 1 the UBP defense in detail. These actions are all in addition to normal network coding actions and they are triggered by timer expiration or by packet reception. We assume each node has a public/private key pair, such that any node i can sign a message with its private key K_i which is denoted by $S_{K_i}(\cdot)$ and any other node in the network can verify this signature.

The protocol is initiated when a node receives a non-innovative coded packet and starts the propagation of a BCP upstream (lines 1-3 of receiving a coded packet). Then, an entry is created for the Upstream Accusation Table (UAT), and a timer

is started for this entry (lines 4-5 of receiving a coded packet). The purpose of the UAT is to keep track of each usptream neighbor that should send an innovative coded packet since a BCP was sent to that upstream neighbor. The time that an upstream neighbor has to send an innovative coded packet is the estimate of the time taken for the BCP to propagate up to the source and then a coded packet to propagate down to j. Upon UAT expiration, the node accuses the upstream neighbor of being an entropy attacker if the upstream neighbor did not manage to send an innovative coded packet (lines 1-2 of UAT expiration).

A node that receives a BCP will first check the signature and then check whether it has innovative packets with respect to the null space checksum within the BCP (lines 1-2 of receiving a BCP). The actions taken by the node depend on whether it has innovative coded packets. If the node does have innovative coded packets with respect to the null space checksum, then the node will broadcast a coded packet and perform the appropriate updates to the SNT (lines 3-7 of receiving a BCP). Note that these same updates are applied to the SNT for every broadcast of coded packets despite it not being explicitly mentioned. In the other case, the node forwards the BCP upstream by selecting the most likely next hop that sent globally non-innovative coded packets (lines 8-14 of receiving a BCP). The forwarded BCP is modified to include the sequence number of the coded packet that is expected to have been globally non-innovative. This sequence number is known since the SNT maintains the sequence number of the coded packet received from an upstream node just before each broadcast.

The propagation of BCPs upstream continues along a path until either a malicious node refuses to keep forwarding it, a node has innovative coded packets and sends them downstream, or the BCP reaches the source. If the BCP reaches the source, then the source always has innovative coded packets and will send an innovative coded packet downstream. Thus, each node has a chance to broadcast and propagate innovative coded packets downstream before the UAT of its downstream neighbor expires. The timers for accusation should be set such that upstream nodes' timers

expire first, and only the most upstream node that makes an accusation will count. So, a malicious node will be accused if it refuses to either forward the BCP upstream or forward innovative coded packets downstream.

4.4.4 Buffer Monitoring

We now present a monitoring-based solution to defend against entropy attacks. Each forwarder is assigned one or more watchdogs. A larger number of watchdogs provides resilience to watchdog failure or misbehavior. The watchdog nodes will ensure that coded packets broadcast by a watched forwarder are random linear combinations of all received coded packets, and the coefficients of this linear combination are chosen according to a publicly known pseudo-random function. This scheme is proactive in nature and thus can immediately detect an attack. However, as any proactive scheme, it has additional overhead for each coded packet broadcast. In addition, there are some network topology constraints that might prevent some flows from having each forwarder assigned the desired number of watchdogs.

For a watchdog to determine whether a coded packet broadcast by a watched forwarder is random linear combinations of all received coded packets by that forwarder, the watchdog must know about all coded packets received by that forwarder for the generation. This poses two challenges. First, the watchdogs must have wireless links to both the watched node and every upstream neighbor of the watched node, which may prohibit BM from being applied to certain topologies. This challenge is a fundamental constraint imposed by the topology, and we analyze in Section 4.5.2 the feasibility of selecting watchdogs in a realistic wireless network. Second, once a valid set of watchdogs are chosen, we need to send minimal amount of data to the watchdog to ensure accurate detection while not hindering the opportunistic routing of the random network coding system.

Detection at a Watchdog

To determine whether a single coded packet $\mathbf{c}_i(t)$ from a node i at time t is consistent with traffic that entered node i, the watchdog must determine the coefficients $\mathbf{r}_i(t)$ used to create the coded packet from the equation:

$$\mathbf{r}_i(t) * \mathbf{A}_i(t) = \mathbf{c}_i(t) \tag{4.9}$$

This is an overconstrained system of $n + m$ equations and n unknowns. Only n equations are needed to determine the relevant elements of $\mathbf{r}_i(t)$. There are some elements of $\mathbf{r}_i(t)$ that correspond to rows of zero vectors in $\mathbf{A}_i(t)$ which cannot be determined, but these are irrelevant elements as they do not affect the coded packet being broadcast.

The impact of this result is that a watchdog only requires the coding headers of the coded packets sent and received by a watched forwarder. A coded packet with typical network coding system parameters has 32 bytes for the coding header while the entire coded packet is 1500 bytes. It is important to only reliably multicast a small portion of the total traffic since random network coding systems gain many advantages by forwarding data opportunistically instead of sending the data reliably each hop. This fundamental characteristic of random network coding systems is still retained with the additional overhead of sending a small portion of a coded packet, the coding header, with reliable multicast. However, simply determining the value $\mathbf{r}_i(t)$ does not completely defend against global entropy attacks as it is difficult to determine whether the values are chosen randomly or to cause a subtle entropy attack.

Instead of attempting to determine whether a $\mathbf{r}_i(t)$ used by a watched forwarder is truly random, we require all nodes to generate the coding coefficients based on a Pseudo-Random Function (PRF). The PRF is keyed with a key known to all nodes in the network (to guarantee the coefficients are pseudo-random, the key only needs to be picked at random and does not need to be secret). The inputs to the PRF are the node's ID along with a sequence number for the packet. The usage of a PRF makes a watchdog's job simple and deterministic to check whether a set of coefficients used

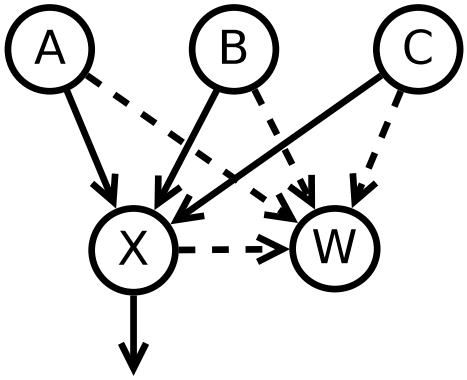

Figure 4.5.: Topology for monitoring defense scenario.

by the watched forwarder is truly random. Also, the inputs to the PRF cannot be controlled by the attacker as a sequence number increases by one with each broadcast coded packet and the node ID does not change. Due to this constraint, the global entropy attacker has no opportunity to guess inputs to the PRF that may produce coefficients that result in an entropy attack. The use of the PRF is computationally inexpensive, and the random coefficients chosen by the PRF are uniformly random which is optimal to satisfy the high decoding probabilities in random network coding systems.

Protocol Description

Figure 4.5 shows an example of a node with one watchdog and three upstream neighbors. The node X is being watched by a watchdog W. The watchdog must receive all coding headers from the upstream neighbors A, B, and C along the wireless links indicated by the dashed lines. Also, the watchdog must receive the coding headers that are broadcast by node X. With this information, along with knowledge of a global PRF used by each node, the watchdog can deterministically check whether node X is correctly creating coded packets or is performing an entropy attack.

Algorithm 4 describes the specific actions of a node j in a monitoring defense. The node j is a forwarder, a watchdog, or both. We use the notation of two sets that

exist for each node (these sets may be empty): $W(i)$ are the watchdogs of node i and $D(i)$ are the downstream neighbors of node i.

To ensure that only a small portion of each coded packet is sent reliably, the coding headers and coded data are sent separately in a Coding Header Packet (CHP) and Coded Data Packet (CDP). The CDP is broadcast unreliably (lines 1-2 of broadcasting a coded packet) and the CHP is reliably multicast to the appropriate set of nodes (lines 3-5 of broadcasting a coded packet). The appropriate set of nodes are the downstream nodes, watchdogs of the downstream nodes, and the watchdog of the broadcasting node (line 4 of broadcasting a coded packet). The watchdogs must receive the CHP to ensure that it has been formed correctly. The downstream nodes must receive the CHP to either reconstruct the coded packet if the downstream node correctly received the CDP (lines 1-2 of receiving CHP from upstream neighbor) or to notify watchdogs that they lost the CDP with a Dropped Data Packet (DDP) (lines 3-5 of receiving a CHP from upstream neighbor). Lastly, watchdogs of downstream neighbors must receive the CHP so that they have a view of the buffer information at the downstream neighbor.

When j is a watchdog and receives a CHP from a node i where $j \in W(i)$, j must create the coding header matrix $\mathbf{V}_i$ of node j (lines 1-3 of received CHP from a watched node), and then check whether the CHP is consistent with $\mathbf{V}_i$ and the random linear combination from the PRF (lines 4-5 of received CHP from a watched node). The information to perform this check is stored in the Watchdog Buffer Table (WBT) when upstream nodes send coded packets to node i (line 1 of received CHP from upstream neighbor of watched node). The WBT must be correctly modified when a node does not receive a coded packet. This is the purpose of broadcasting the DDP to watchdog nodes of j when j does not receive the corresponding CDP to a CHP. The DDP prompts the watchdogs to either remove the entry or drop a future reception of a CHP that corresponds to the dropped packet (lines 1-5 of receiving a DDP). If an attacker abuses the use of DDPs and claims to drop more coded packets than the the measured link qualities, then the watchdog can inform the routing layer

Table 4.2: Notation for BCP and BM protocols

$T^{\mathbf{c}}_{i,j}$	Avg. time of a coded packet to propagate downstream from node i to j
$T^{B}_{i,j}$	Avg. time of a BCP to propagate upstream from node i to j
T^{S}_{i}	Avg. time between coded packet sends at node i
T^{E}	Exoneration time for a hybrid of UBP
$P^{R}_{i,j}$	Attack strength from node i to j under UBP
$P^{H}_{i,j}$	Attack strength from node i to j under a hybrid of UBP with an exoneration phase
$N_{i,j}$	Average number of coded packets that can be sent by node i that are globally non-innovative to downstream node j in UBP before i must send an innovative coded packet

of the change in link qualities which will route data around the attacker much like the modified link qualities in our NLA defense.

4.5 Security Analysis

We analyze UBP in terms of *attack strength* which denotes the proportion of time which a globally non-innovative coded packet can be sent undetected. The buffer monitoring is much stronger in terms of security since an attacker is not capable of evading detection by a watchdog. However, BM cannot be applied to an arbitrary flow of a topology and has a higher network overhead. So, we analyze the proportion of flows BM can be applied to in the Roofnet topology [49].

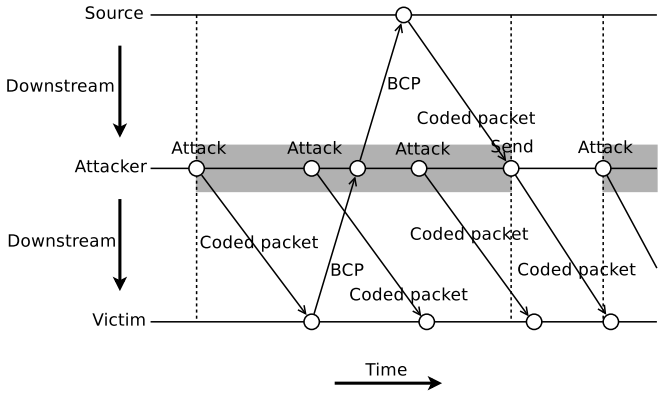

Figure 4.6.: Timeline for source, attacker, and victim for the UBP protocol.

4.5.1 Attack Strength of UBP

We aim to describe the attack strength in terms of the characteristics of the network. Specifically, attack strength represents the proportion of time that coded packets can be sent from an attacker that are globally non-innovative with respect to a victim's coding buffer and the attacker will not be detected as an entropy attacker. In the remainder of the time, the attacker cannot send a globally non-innovative coded packet without being detected. Also, since the sending times of coded packets are fixed by the protocol, the attack strength also represents the proportion of packets sent by the attacker that can be globally non-innovative while not being detected. The attack strength will depend on the network characteristics of the average time taken for both a coded packet (or combinations of the coded packet) to traverse downstream and a BCP to traverse upstream. These averages differ since coded packets are larger and sent opportunistically downstream, while BCPs are smaller and sent reliably upstream immediately at each hop. For this analysis, we use notation from Table 4.2.

Figure 4.6 shows the timeline of events that lead to points where an attacker can attack in UBP without detection. The scheme waits until an attack is detected downstream, and then it reacts by sending a BCP upstream along the path that contains the global entropy attacker. The attacker is detected by a timer expiring at the entropy attacker's immediate downstream neighbor which is the time it takes

for the BCP to reach the source from the attacker and then an innovative coded packet to traverse downstream to the attacker. During this entire time, the attacker can consecutively send globally non-innovative coded packets and send an innovative coded packet when it knows the immediate downstream neighbor's UAT is about to expire.

We first determine the number of consecutive non-innovative coded packets that can be sent by attacker node i that target victim j without detection as:

$$N_{i,j}^{R} = 1 + \frac{T_{i,j}^{\mathbf{c}} + T_{j,src}^{B} + T_{src,i}^{\mathbf{c}}}{T_{i}^{S}} \tag{4.10}$$

In addition to the one initial attack packet, there are several opportunities for the attacker to send globally non-innovative coded packets. The average number of opportunities is equal to the total time it takes before the attacker must send an innovative coded packet over the average time between coded packet sends of the attacker node.

We can then determine the attack strength from attacker node i to victim node j as:

$$P_{i,j}^{R} = \frac{N_{i,j}^{R}}{N_{i,j}^{R} + 1} \tag{4.11}$$

For each consecutive non-innovative coded packet that can be sent by an entropy attacker, the entropy attacker must send one innovative coded packet to remain undetected.

The attack strength $P_{i,j}^{R}$ is always at least 0.5 which means that at least half of the coded packets can be globally non-innovative. The attack strength increases as the values $T_{src,j}^{\mathbf{c}}$ and $T_{j,src}^{B}$ become larger compared to T_{i}^{S} which happens in larger networks and when the node i does more broadcasting. The large attack strength is due to the exoneration of the attacker given just one innovative coded packet. Thus, the single upstream path found by UBP could enter an exoneration phase for a period

of time which requires more overhead but detects a global entropy attack proactively. This results in a hybrid scheme with an attack strength of:

$$P_{i,j}^{H} = \frac{N_{i,j}}{N_{i,j} + 1 + \frac{T_E}{T_i^S}} \tag{4.12}$$

The exoneration period T_E can be varied to obtain various trade-offs between the additional overhead of the proactive detection in the exoneration period and the attack strength possible at the attacker.

An obvious way to enforce an exoneration period for a path is to assign watchdogs as in the BM scheme to these nodes. This would not impose the high overhead of BM throughout the entire network at all times as UBP can reactively determine which path an entropy attacker is taking. Alternatively, one could design a different scheme that can provide an accurate proactive defense which can be used in conjunction with UBP in this same manner.

4.5.2 Watchdog Selection Constraints of BM

The buffer monitoring defense has a much higher level of security since it can ensure an attack strength of 0. The watchdog(s) of a forwarder have complete information about the coding headers of the forwarder, and the watchdog can deterministically assess whether the forwarder created a random combination using the entire coding buffer. This scheme cannot be employed for each flow of any topology.

The constraint for using the buffer monitoring defense is:

$$\forall f \in F, \left| \left(L(f) \cap \left(\bigcap_{u \in U(f)} L(u) \right) \right) - f \right| \geq n \tag{4.13}$$

F is the set of forwarders for a flow, $L(x)$ is the set of nodes that x has a wireless link to (this includes x itself), $U(x)$ is the set of upstream neighbors of node x, and n is the minimum number of watchdogs assigned to each node. This watchdog assignment allows both the nodes in the flow and nodes outside the flow to act as

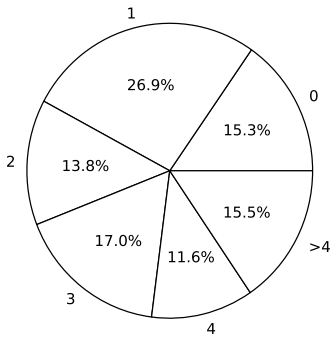

Figure 4.7.: Maximum valid assignment of n watchdogs per forwarder.

watchdogs for a forwarder. Furthermore, we allow a forwarder's upstream neighbor to act as its watchdog which will reduce the multicast overhead when the upstream neighbor must reliably multicast coding headers since the upstream neighbor does not need to spend communication overhead sending this coding header to itself.

We use the Roofnet data to represent a typical wireless network topology. There are 38 nodes in the network, so we take all $\binom{38}{2} = 1406$ possible flows. Out of these flows we discard 174 trivial flows that contain no forwarders and present results based on the remaining 1232 flows.

We present information about the maximum watchdog assignment per flow in Figure 4.7. 15.3% of flows cannot employ a buffer monitoring defense without changing the forwarder nodes that were optimally selected by the routing algorithm. These flows contain some forwarder that does not have a wireless link to any node in the topology that also has a wireless link to each upstream neighbor of the forwarder. At least one watchdog per forwarder is a minimal constraint. A network may aim to protect against an attacker that imposes false accusations. In this scenario, three watchdogs can be assigned to each forwarder to vote on detection, and only 44.1% of flows allow three watchdogs per forwarder.

4.6 Entropy Defense Overhead Analysis

We provide analysis of the overhead in terms of computation and communication of our proposed defenses attacks. The defense design targeted low overhead to ensure practical use of such defenses, and we are able to provide greater details for this lower overhead here.

4.6.1 Computation Overhead

The originator of a BCP in UBP must compute a null space checksum and a signature for the BCP. The following benchmarked time values are performed on general commodity hardware[1]. As noted earlier in Section 4.4.3, creating a null space checksum requires the solution to a system of $R(\mathbf{V}_i(t))$ equations and $R(\mathbf{V}_i(t))$ unknowns where $R(\mathbf{V}_i(t)) < n$. Solving an n by n system of equations requires roughly 0.4 ms (where $n = 32$ and a symbol is 1 byte), which is the largest system of equations that may have to be solved. A single DSA sign requires roughly 1 ms of computation. Thus, overall, the originator of a BCP requires roughly 1.4 ms of computational overhead on general commodity hardware.

Nodes receiving a BCP in UBP must verify the signatures attached to these packets. Verifying a signature requires roughly 1.1 ms of computation. The reception of a BCP message requires a check of the null space checksum that was received which is simply a matrix multiplication. The computational time of a matrix multiplication on the coding headers of a coding buffer is negligible.

4.6.2 Communication Overhead

The communication overhead in UBP are the BCPs that are sent using reliable unicast due to our strategy of finding the single upstream path that the attacker is on. This communication overhead is quite small as the BCPs are small due to our

[1] 2.4Ghz processor and OpenSSL library for DSA signature computations

use of checksums. Thus, we focus on the communication overhead of BM as it relies heavily on reliable multicasts to deliver header information reliably whenever a coded packet broadcast occurs.

Ensuring reliability on the multicast requires overhead in terms of resending the packet until each destination has received the message. Previous work exists on sending large messages with reliable multicast at the link-layer [68]. However, their key contribution is the use of forward error correction codes to break a large message into several small messages. These small messages are easier to receive since the probability of packet delivery is higher for smaller messages. Thus, the recipients just need to receive any number of small messages to reconstruct the original large message.

We reliably multicast much smaller messages that can be easily sent in one small packet (40 bytes). Thus, breaking the small message into even smaller messages will negate any performance improvements since each message has overhead of sending link-layer headers as well as physical layer overhead. Thus, we propose to send the small message multiple times until all receivers obtain the message. We analyze the number of times the message must be broadcast before each receiver obtains the message given the packet delivery probabilities on each link. We do not present analysis on the ACKs that must be sent from each receiver to the sender which would need to be sent in a way to avoid congestion.

We can analyze the number of times a message must be sent given that it is sent to N nodes over links with packet delivery probabilities of $p_1, p_2, ..., p_N$. Let X be a random variable that denotes the fewest number of times a message must be sent such that all N receivers receive the message at least once. We aim to calculate $Pr(X = k)$, so we can consider each receiver as an independent geometric random

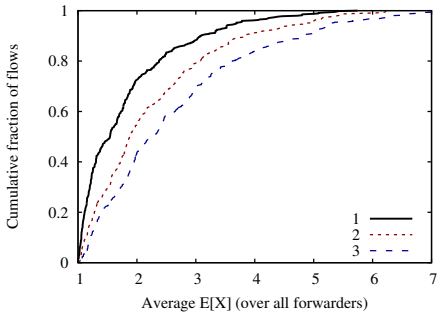

Figure 4.8.: Given flows in the Roofnet topology, we show the communication overhead of reliable multicasts in BM varying the number of watchdogs per node to 1, 2, and 3.

variable Y_i which corresponds to the link state p_i. We can express $Pr(X \leq k)$ in terms of the independent geometric random variables as follows:

$$Pr(X \leq k) = Pr\left(\bigcap_{i=1}^{N} Y_i \leq k\right) = \prod_{i=1}^{N} Pr(Y_i \leq k)$$

$$= \prod_{i=1}^{N}\left[\sum_{j=1}^{k} Pr(Y_i = j)\right] = \prod_{i=1}^{N}\left[\sum_{j=1}^{k}(1-p_i)^{j-1}p_i\right]$$

With $Pr(X \leq k)$ we can obtain $Pr(X = k)$ by $Pr(X = k) = Pr(X \leq k) - Pr(X \leq k-1)$. The function for $Pr(X = k)$ allows us to compute the expected number of broadcasts of a message such that each receiver obtains the message, $E[X]$. The average overhead for reliably multicasting M bytes of data will be $M * E[X]$.

We use a heuristic for summed link qualities to determine the best watchdog selection out of all possible watchdogs. Each forwarder has each downstream neighbor and the watchdogs of each downstream neighbor as recipients of a DHP. Given the link qualities from the topology and these sets of recipients we can apply the formula for $E[X]$ at each forwarder to obtain an average for a flow.

We use the Roofnet data to show the expected communication overhead when sending DHPs in BM with various number of watchdogs per forwarder. We consider the flows in Roofnet where an assignment of at least 3 watchdogs per nodes is possible (541 flows or 44.1% of non-trival flows). Figure 4.8 presents a CDF (Cumulative Distribution Function) for $E[X]$ of DHP reliable multicasts in BM. As expected, the

overhead increases with more watchdogs being assigned to each node due to more recipients in each wireless multicast.

4.7 Summary

We show via simulations the impact of entropy attacks on the overall routing of a wireless network coding system. We propose an effective defense against local entropy attacks and show the difficulties in defending against a global entropy attack. We propose two variations on a global entropy defense which differ in their defense capabilities and overhead. We provide analysis to quantify the defense capabilities of these global defense schemes.

Algorithm 3 Reactive upstream buffer propagation protocol for node j in addition to normal network coding actions

BCP: packet with contents $\langle$originator, null space checksum, sequence number, originator signature$\rangle$

UAT: table with entries $\langle$upstream node, originator, null space checksum$\rangle$

SNT: table with entries $\langle$local sequence number, upstream node, upstream sequence number$\rangle$

Received coded packet $\mathbf{c}$ from upstream neighbor k with sequence number u_k at time t

1: **if** $\mathbf{c}$ is non-innovative **then**
2: $BCP \leftarrow \langle j, \mathbf{s}_j(t), u_k, S_{K_j}(\mathbf{s}_j(t))\rangle$
3: reliable_unicast(k, BCP)
4: add$(\langle k, j, \mathbf{s}_j(t)\rangle, UAT)$
5: start_timer$(\langle k, j, \mathbf{s}_j(t)\rangle)$
6: **else**
7: remove$(\langle u_j, *, *\rangle, SNT)$
8: add$(\langle u_j, k, u_k\rangle, SNT)$
9: **if** $\exists\langle k', i, \mathbf{s}_i(t)\rangle \in UAT$ s.t. $k' = k$ **then**
10: **if** $\mathbf{c}$ is innovative w.r.t. $\mathbf{s}_i(t)$ **then**
11: remove$(\langle k, *, *\rangle, UAT)$

Received BCP $\langle i, \mathbf{s}_i(t'), u'_j, S_{K_i}(\mathbf{s}_i(t'))\rangle$ at time t from node l

1: **if** $S_{K_i}(\mathbf{s}_i(t'))$ is correct **then**
2: **if** $\mathbf{V}_j(t')$ has innovative coded packets w.r.t $\mathbf{s}_i(t')$ **then**
3: $c \leftarrow$ create_coded_packet()
4: broadcast$(\langle c, u_j\rangle)$
5: $\langle u_j, k, u_k\rangle \leftarrow$ get$(\langle u_j, *, *\rangle, SNT)$
6: $u_j \leftarrow u_j + 1$
7: add$(\langle u_j, k, u_k\rangle, SNT)$
8: **else**
9: $\langle u'_j, k, u_k\rangle \leftarrow$ get$(\langle u'_j, *, *\rangle, SNT)$
10: $BCP \leftarrow \langle i, \mathbf{s}_i(t'), u_k, S_{K_i}(\mathbf{s}_i(t'))\rangle$
11: reliable_unicast(k, BCP)
12: **if** k is not source **then**
13: add$(\langle k, i, \mathbf{s}_i(t)\rangle, UAT)$
14: start_timer$(\langle k, i, \mathbf{s}_i(t)\rangle)$

Expired timer $\langle k, i, \mathbf{s}_i(t)\rangle$ s.t. $\langle k, i, \mathbf{s}_i(t)\rangle \in UAT$

1: **if** no recent accusations with same originator i **then**
2: accuse(k)

Algorithm 4 Buffer monitoring protocol for node j in addition to normal network coding actions

CHP : packet with ⟨source, sequence number, coding header⟩

CDP : packet with ⟨source, sequence number, coded data⟩

DDP : packet with ⟨node dropping packet, packet source, packet sequence ⟩

WBT : table entries ⟨ watched node, source, sequence number, coding header ⟩

$W(x)$: watchdog nodes for node x

$D(x)$: downstream neighbors for node x

$PRF(x, y)$: pseudo-random function which maps $\mathbb{Z}^+ \times \mathbb{Z}^+$ to $\mathbb{F}_q^n$

Received CHP $\langle i, u_i, \mathbf{v}\rangle$ from upstream neighbor i

1: **if** Received CDP $\langle i, u_i, \mathbf{x}\rangle$ **then**
2: Reconstruct coded packet $\mathbf{c} = \langle \mathbf{v}, \mathbf{x}\rangle$ and store in buffer
3: **else**
4: $DDP \leftarrow \langle j, i, u_i\rangle$
5: reliable_multicast$(W(j), DDP)$

Broadcasting coded packet $\mathbf{c} = \langle \mathbf{v}, \mathbf{x}\rangle$ created by random vector $PRF(j, u_j)$

1: $CDP \leftarrow \langle j, u_j, \mathbf{x}\rangle$
2: broadcast(CDP)
3: $S \leftarrow W(j) \cup D(j) \cup \left(\bigcup_{i \in D(j)} W(i)\right)$
4: $CHP \leftarrow \langle j, u_j, \mathbf{v}\rangle$
5: reliable_multicast(S, CHP)

Received CHP $\langle i, u_i, \mathbf{v}\rangle$ from node i s.t. $j \in W(i)$

1: initialize_coding_header_matrix$(\mathbf{V}_i)$
2: **for all** $\langle i, *, *, v\rangle$ in WBT **do**
3: add_coding_header$(v, \mathbf{V}_i)$
4: **if** $\mathbf{V}_i * PRF(i, u_i) \neq \mathbf{v}$ **then**
5: accuse(i)

Received CHP $\langle k, u_k, \mathbf{v}\rangle$ from node k s.t. $i \in D(k)$, $j \in W(i)$

1: add$(\langle i, k, u_k, \mathbf{v}\rangle, WBT)$

Received DDP $\langle i, k, u_k\rangle$ from node i s.t. $i \in D(k)$, $i \in W(j)$

1: **if** $\langle i, k, u_k, *\rangle \in WBT$ **then**
2: remove$(\langle i, k, u_k, *\rangle, WBT)$
3: **else**
4: drop_future_receptions$(\langle k, u_k, *\rangle)$

5 RELATED WORK

The related work for Chapters 2, 3, & 4 is discussed here. Section 5.1 contains related work on diversity while the related work on network coding is grouped into Section 5.2.

5.1 Network Diversity

We present relevant related work to our network diversity problem. To our knowledge there is no other work studying the exact same problem, but there are several works considering similar problems that differ in their network and adversarial models. First, we discuss a work that uses similar diversity assignment language, but their model solves a different underlying problem. Second, we cover numerous resilient topology construction problems from graph theory and resilient key distribution problems from wireless networking. Both categories of work are similar to our work as they target resilience goals with intelligent placement, but their attack models differ vastly from ours. Finally, we mention a path diversity work that proposes methodologies to measure the diversity of an internet topology based on how many geographically diverse paths exists between endpoints on the internet.

Diversity assignment. The work most similar to ours considers diversity assignment over nodes of a distributed system [69], but the goal of that work is to prevent the spread of malware. In contrast, we assume that if a node of some variant is compromised, then all nodes of that variant are also compromised, as the attacker is not restricted to only using links within the network. To assign diversity to prevent the spread of malware, the computation problem in [69] is different from ours as they intend to minimize the number of links which contain two nodes of the same variant. Thus, their underlying optimization problem for variant assignment is a version of

the classic graph coloring algorithm. This problem is NP-Hard, so their work also explores a heuristic solution which can scale to large networks.

Fault-tolerant topology construction. Existing work has introduced the concept of the *fault-diameter* of a graph, which is a metric that bounds the diameter of a graph given that a bounded number of nodes may fail [70–73]. For a network topology, this means that if the number of failures is bounded, then the maximum number of hops between any two correct nodes will not exceed the fault-diameter. This translates to acceptable latency and overhead even in the worst case. Work in this area has considered various ways to create graphs with good fault-diameters, but these methods only consider unweighted graphs where edges are possible between any pair of nodes. In our work, we assume the topology is chosen ahead of time and fixed to ensure good link quality, and we do not need to add edges for our technique.

In wireless contexts, work has studied the allocation of energy among nodes in a wireless ad hoc network to ensure high connectivity even when some bounded number of nodes fail [74–76]. The work assumes that node positions are fixed and an amount of energy can be assigned to each node. Higher energy at a node implies a larger transmission range and more possible connections for that node. The optimization problem is to find a power assignment to nodes which minimizes the global power consumption while ensuring connectivity among correct nodes given a bounded number of nodes can fail. This optimization problem is studied in detail, providing a MIP and exploring various approximation techniques.

WSN key distribution. Wireless Sensor Networks (WSNs) consist of resource constrained devices which sense physical phenomena and deliver this information over a wireless newtork to a base station. In this context, PKI and full pair-wise key initialization are prohibitive due to the limitations of sensors. Thus, various work proposes special key distributions, where secret information is shared among more than a single pair of nodes [77–81]. This has similarities to diversity assignment as the physical capture of a single node allows an attacker to utilize the secret information on that node to attack links of other nodes which share similar secret information.

Our work does fundamentally differ as we perform diversity assignment with the complete topology information to maximize a resiliency metric where as WSN key distribution work focuses on assigning initial secret information to nodes to maximize that the potential of many links are secure. With the potential for many secure links, a random wireless topology can be created and have certain resiliency properties.

Path diversity. Other work has studied the possible geographically diverse paths of real-world topologies [82]. The assumptions of this work are that problems on today's internet are correlated geographically, so having multiple paths which contain nodes that are geographically diverse will result in higher reliability. The main contributions of this work are defining the metric of geographic diversity for a graph and analyzing this value for realistic graphs. No assignment problem exists to date in this context as diversity has been fixed by geographic location.

5.2 Byzantine Resilient Network Coding

We discuss relevant work related to ours on byzantine resilient network coding. For relevant pollution defenses we consider prior work on polluted packet detection, polluted packet correction, and polluter identification techniques. Our proposed pollution defense protocol, Split Null Keys, falls under the category of polluted packet detection. Then for relevant entropy attack and defense work we discuss two prior mentions of such an attack along with relevant wireless security work. Although entropy attacks are unique to network coding, its impact is similar to selective forwarding while leveraging techniques similar to wormhole attacks to remain stealthy.

Detecting polluted packets at intermediate nodes. Several homomorphic signature schemes proposed to provide a verification function for intermediate nodes in the network. The works in [15–19] utilize cryptographic primitives that rely on the discrete logarithm problem for security, which causes two major performance issues. A lower bound is enforced on the symbol size, and a high computational overhead is imposed by numerous modular exponentiations or elliptical curve operations. Work

has been done to address the computational overhead of cryptographic pollution defenses at intermediate nodes. Zhao et al. [83] proposes to speed up computations by utilizing graphical processing units while Gkantsidis et al. [27] proposes to probabilistically verify received blocks in peer-to-peer networks. Gennaro et al. [84] show that the scheme [15] could use smaller coefficients near the source to reduce computational and communication overhead. The coefficients become larger with each hop and eventually approach the overhead imposed by previous cryptographic schemes, so topologies with many hops will still suffer.

Boneh et al. [20] propose the first homomorphic signature scheme over binary field sizes by using lattice-based techniques. Their construction limits the number of times signatures can be combined, and the key sizes are much larger than traditional cryptographic constructions.

Agrawal et al. [21] present a homomorphic MAC that relies on pseudo-random functions to overcome performance limitations. A work [22] shows an efficient way to overcome a problem unique to homomorphic MACs known as *tag pollution*. However, the underlying scheme still does not scale with the number of attackers in the network.

Dong et al. [23] propose a protocol for wireless networks that relies on checksums being disseminated periodically throughout the network. Attackers cannot conduct a forgery attack by observing a checksum because intermediate nodes verify received packets against checksums that were created at the source at a later time than the time when packets were received by intermediate nodes. The scheme has a lower overhead than cryptographic defenses but causes coded packets to be delayed before being verified.

Kehdi et al. [24] propose an algebraic based approach that uses null space properties to defend against pollution attacks. The scheme is suitable for large peer-to-peer networks with path diversity but not applicable in wireless networks where such diversity cannot be guaranteed. Our scheme is also based on null space properties but does not rely on path diversity and has a small communication overhead per generation.

Correcting polluted packets at receivers. The work [25] encodes redundant information to reconstruct the valid coded packets at the receiver in the presence of a byzantine adversary. However, the throughput of the network is dependent on the adversary's network capacity to the receiver, so an adversary with high network capacity can potentially reduce the throughput of the network to zero. The work [26] proposes to limit nodes' network capacity by limiting the broadcasts of each node to ensure the scheme [25] retains high throughput in the presence of adversaries. Limiting broadcasts is inconsistent with practical wireless network coding systems.

Identifying polluting attackers. The work [85] uses the homomorphic MACs [21] to determine the subspaces a node has received and the subspaces a node has forwarded. This is sufficient information to determine if a node is a pollution attacker. A work based on monitoring [60] is able to detect whether a node is polluting with high probability given that nodes protect the headers of coded packets with error correcting codes and that multiple honest watchdogs exist per node in the network. The work [86] proposes a monitoring technique that requires source encoding where the amount of overhead is dependent on the channel qualities in the network. The adversary has a higher probability of being detected because it has to pollute many packets to overcome the source encoding.

Entropy attacks. Entropy attacks have been considered for network coding systems, but defenses have been proposed only to mitigate the overhead of transferring non-innovative coded packets [27, 28]. These works do not consider neither the impact on routing nor the possibility of a global entropy attack. In [27], the authors propose additional local coordination in a peer-to-peer network coding system prior to obtaining a coded packet to ensure it is innovative. The authors of [28] are concerned with the additional computation required at a node to determine whether a received coded packet is innovative or not. Their solution is to probabilistically check the linear independence of a coded packet which ensures that non-innovative coded packets are dropped immediately using minor computation.

Selective forwarding attacks. Wireless mesh network security have considered the effects of a byzantine adversary conducting a selective forwarding attack [62], where a malicious node refuses to forward some packets it receives. Such an attack can cause significant damage to network performance. Monitoring is a suitable solution to detect selective forwarding in a wireless network [64]. Nodes that neighbor an attacker can detect when the attacker has received a packet and not forwarded the packet. One of the defense schemes we propose against global entropy attacks also relies on monitoring. Unlike in defenses against selective forwarding attacks, using monitoring to defend against entropy attacks is more challenging because the attacker in an entropy attack is still forwarding packets, but the neighboring nodes need more information to determine that the attacker is coding non-innovative coded packets.

Wormhole attacks. The global entropy attack may use an out-of-band communication channel between nodes just as in a wormhole attack [87–89]. An upstream and downstream node collude by using coded packets received at the downstream node to create new coded packets at the upstream node. Existing defenses against wormhole attacks focus on individual packets which cannot be applied to network coding as packets are combined by forwarders. In [88], temporal and geographical leashes are placed on packets to ensure they are correctly forwarded through the network. Such a technique cannot be applied to network coding since packets are coded together and each forwarder creates new packets.

6 CONCLUSION AND FUTURE WORK

The attackers we consider in this dissertation have significant power as they can obtain compromising exploits for certain variants of a system. Additionally with a compromise, the attacker has protocol knowledge and the ability to craft sophisticated malicious routing actions. Today's networks can easily fail given such attackers, and these types of attackers will become more prevalent as our society continues to rely on network infrastructure for our most critical services. This dissertation provides novel techniques greatly improving the construction of resilient networks for critical services.

Our network diversity efforts have filled a vital gap in research between creating diverse systems and providing network resilience through diversity. Through our proposed techniques, networks can retain well-connected surviving portions of their network despite the use of unknown compromising exploits against the network. We were thorough in investigating the important aspects of such a problem by finding appropriate computational solutions, optimizing for various networking goals, and testing in conditions with erroneous compromise information. The network diversity techniques outlined in this dissertation can immediately be applied to today's networks to greatly improve resilience.

For networks that must provide high performance data delivery that contain a subset of malicious routers, we provide significant improvements in securing networking coding protocols. We improve the state-of-the-art in terms of practical pollution defenses and emphasize the importance of entropy attacks along with providing defense techniques. We overcome these two most crucial hurdles when aiming to leverage performance improvements of network coding in a byzantine environment. The proposed pollution defense focuses practicality to ensure we still achieve performance improvements despite the defense overhead. Our work on entropy attacks highlights

the significant impact of such an attack on a network coding system which requires some type of defense to deal with entropy attacks in a byzantine environment. We provide novel defenses analyzing their effectiveness and overhead.

Future Network Diversity Work. There are two main directions to pursue for the diversity work that fall into different disciplines of computer science. First, a series of measurement studies could be performed identifying the effectiveness of different types of vulnerabilities. Analyzing vulnerability databases could lead to some understanding of how often compromising exploits are available for different systems and whether these exploits could target multiple systems. Also, performing ethical experiments with people to attain information as to the diversity of susceptibility to coding vulnerabilities by distinct programming teams or susceptibility to social engineering attacks. Second, a theoretical investigation could be performed to better understand the hardness of the diversity assignment problems. Ideally it should be possible to provide positive and negative approximation results. There is hope for this direction as the problem has commonalities to disjoint Steiner tree packing problems.

Future Byzantine Resilient Network Coding Work. The core future work involves practical implementation, deployment, and testing under malicious settings. Such work answers many vital questions before widespread adoption of our techniques. We provide simulation and analytical evidence of low overhead which implies high network performance. However, there are commonly extraneous factors which are not easily anticipated or captured through simulation and analysis that could play an important role in performance. For a holistic test of performance under attack it is insightful to observe performance where a subset of malicious routers perform both entropy and pollution attacks.

LIST OF REFERENCES

LIST OF REFERENCES

[1] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro. OS Diversity for Intrusion Tolerance: Myth or Reality? In *Proceedings of the Conference on Dependable Systems and Networks*, pages 383–394, 2011.

[2] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser. N-Variant Systems: A Secretless Framework for Security through Diversity. In *Proceedings of the Usenix Security Symposium*, pages 105–120, 2006.

[3] I. Gashi, P. Popov, and L. Strigini. Fault Tolerance via Diversity for Off-the-Shelf Products: A Study with SQL Database Servers. *Transactions on Dependable and Secure Computing*, 4(4):280–294, 2007.

[4] Y. Deswarte, K. Kanoun, and J. Laprie. Diversity against Accidental and Deliberate Faults. In *Proceedings of Computer Security, Dependability and Assurance: From Needs to Solutions*, pages 171–171, 1998.

[5] J. Jin, T. Ho, and H. Viswanathan. Comparison of Network Coding and Non-Network Coding Schemes for Multi-Hop Wireless Networks. In *Proceedings of the Symposium on Information Theory*, pages 197–201, 2006.

[6] J. Widmer and J.-Y. Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *Proceedings of the Workshop on Delay-Tolerant Networking*, pages 284–291, 2005.

[7] D. S. Lun, M. Médard, R. Koetter, and M. Effros. Further Results on Coding for Reliable Communication over Packet Networks. *Computing Research Repository*, 1(1):3–20, 2005.

[8] Y. Wu, P. A. Chou, and S. Kung. Minimum-Energy Multicast in Mobile Ad Hoc Networks Using Network Coding. *Transactions on Communications*, 53(11):1906–1918, 2005.

[9] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *Proceedings of the Conference of the Special Interest Group on Data Communication*, 2007.

[10] X. Zhang and B. Li. Optimized Multipath Network Coding in Lossy Wireless Networks. In *Proceedings of the Conference on Distributed Computing Systems*, volume 27, pages 622–634, 2008.

[11] X. Zhang and B. Li. DICE: A Game Theoretic Framework for Wireless Multipath Network Coding. In *Proceedings of the Symposium on Mobile Ad Hoc Networking and Computing*, pages 293–302, 2008.

[12] S. Katti, D. Kabati, W. Hu, H. Rahul, and M. Medard. The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2005.

[13] J. Le, J. C. S. Lui, and D. M. Chiu. DCAR: Distributed Coding-Aware Routing in Wireless Networks. In *Proceedings of the Conference on Distributed Computing Systems*, volume 9, pages 596–608, 2008.

[14] S. Das, Y. Wu, R. Chandra, and Y. C. Hu. Context-based Routing: Technique, Applications, and Experience. In *Proceedings of the Symposium on Networked Systems Design and Implementation*, volume 8, pages 379–392, 2008.

[15] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a Linear Subspace: Signature Schemes for Network Coding. In *Proceedings of the Conference on Practice and Theory of Public-Key Cryptography*, pages 68–87, 2009.

[16] M. Krohn, M. Freedman, and D. Maziéres. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In *Proceedings of the Symposium on Security and Privacy*, pages 226–240, 2004.

[17] D. Charles, K. Jain, and K. Lauter. Signatures for Network Coding. *Proceedings of the Conference in Information Sciences and Systems*, 1(1):3–14, 2006.

[18] F. Zhao, T. Kalker, M. Médard, and K. Han. Signatures of Content Distribution with Network Coding. In *Proceedings of the Symposium on Information Theory*, pages 556–560, 2007.

[19] Q. Li, D. Chiu, and J. Lui. On the Practical and Security Issues of Batch Content Distribution Via Network Coding. In *Proceedings of the Conference on Network Protocols*, pages 158–167, 2006.

[20] D. Boneh and D. Freeman. Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In *Proceedings of the Conference on Practice and Theory of Public-Key Cryptography*, pages 1–16, 2011.

[21] S. Agrawal and D. Boneh. Homomorphic MACs: MAC-Based Integrity for Network Coding. In *Proceedings of the Conference on Applied Cryptography and Network Security*, pages 292–305, 2009.

[22] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. Shenz. Padding for Orthogonality: Efficient Subspace Authentication for Network Coding. In *Proceedings of the Conference on Computer Communications*, pages 1026–1034, 2011.

[23] J. Dong, R. Curtmola, and C. Nita-Rotaru. Practical Defenses against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks. In *Proceedings of the Workshop on Wireless security*, pages 111–122, 2009.

[24] E. Kehdi and B. Li. Null Keys: Limiting Malicious Attacks Via Null Space Properties of Network Coding. In *Proceedings of the Conference on Computer Communications*, pages 1224–1232, 2009.

[25] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard. Resilient Network Coding in the Presence of Byzantine Adversaries. In *Proceedings of the Conference on Computer Communications*, pages 616–624, 2007.

[26] D. Wang, D. Silva, and F. R. Kschischang. Constricting the Adversary: A Broadcast Transformation for Network Coding. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2007.

[27] C. Gkantsidis and P. Rodriguez Rodriguez. Cooperative Security for Network Coding File Distribution. In *Proceedings of the Conference on Computer Communications*, volume 3, page 5, 2006.

[28] Y. Jiang, Y. Fan, X. (Sherman) Shen, and C. Lin. A Self-Adaptive Probabilistic Packet Filtering Scheme against Entropy Attacks in Network Coding. *Computer Networks*, 53(18):3089–3101, 2009.

[29] A. Newell, D. Obenshain, T. Tantillo, C. Nita-Rotaru, and Y. Amir. Increasing Network Resiliency by Optimally Assigning Diverse Variants to Routing Nodes. In *Proceedings of the Conference on Dependable Systems and Networks*, pages 1–12. ©2013 IEEE. Reprinted with permission.

[30] A. Newell and C. Nita-Rotaru. Split Null Keys: A Null Space Based Defense for Pollution Attacks in Wireless Network Coding. In *Proceedings of the Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*, pages 479–487. ©2012 IEEE. Reprinted with permission.

[31] A. Newell, R. Curtmola, and C. Nita-Rotaru. Entropy Attacks and Countermeasures in Wireless Network Coding. In *Proceedings of the Conference on Security and Privacy in Wireless and Mobile Networks*, pages 185–196. DOI: 10.1145/2185448.2185473, 2012.

[32] LTN. `http://www.ltnglobal.com/`.

[33] L. Lamport. The Part-Time Parliament. *Transactions on Computer Systems*, 16(2):133–169, 1998.

[34] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the Symposium on Operating System Design and Implementation*, volume 99, pages 173–186, 1999.

[35] A. Schrijver. *Theory of Linear and Integer Programming*. 1998.

[36] High-Performance Software for Mathematical Programming and Optimization. `http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/`. Accessed: 5/31/2012.

[37] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed Integer Programming for Multi-Vehicle Path Planning. In *Proceedings of European Control Conference*, pages 2603–2608, 2001.

[38] L. Pallottino, E. Feron, and A. Bicchi. Conflict Resolution Problems for Air Traffic Management Systems Solved with Mixed Integer Programming. *Transactions on Intelligent Transportation Systems*, 3(1):3–11, 2002.

[39] G. Huang, B. Baetz, and G. Patry. Grey Integer Programming: An Application to Waste Management Planning under Uncertainty. *European Journal of Operational Research*, 83(3):594–620, 1995.

[40] Coin-OR. `http://www.coin-or.org/`.

[41] SCIP. http://scip.zib.de/.

[42] Y. Amir, B. Coan, J. Kirsch, and J. Lane. Prime: Byzantine Replication under Attack. *Dependable and Secure Computing*, 8(4):564–577, 2011.

[43] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *Proceedings of the Symposium on Networked Systems Design and Implementation*, volume 9, pages 153–168, 2009.

[44] T. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the Symposium on Theory of Computing*, pages 216–226, 1978.

[45] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung. Network Information Flow. *Transactions on Information Theory*, 46(4):1204–1216, 2000.

[46] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On Randomized Network Coding. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, volume 41, pages 11–20, 2003.

[47] J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure Network Coding for Wireless Mesh Networks: Threats, Challenges, and Directions. *Computer Communications*, 32(17):1790–1801, 2009.

[48] A. Newell, J. Dong, and C. Nita-Rotaru. On the Practicality of Cryptographic Defenses against Pollution Attacks in Wireless Network Coding. *Computing Surveys*, 45(3):39, 2013.

[49] MIT roofnet. http://pdos.csail.mit.edu/roofnet/doku.php.

[50] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding Royer. A Secure Routing Protocol for Ad Hoc Networks. pages 78–87, 2002.

[51] *Advanced Encryption Standard (AES)*. Number FIPS 197. National Institute for Standards and Technology, 2001.

[52] *The Keyed-Hash Message Authentication Code (HMAC)*. Number FIPS 198. National Institute for Standards and Technology, 2002.

[53] *Secure Hash Standard (SHA1)*. Number FIPS 180-1. National Institute for Standards and Technology, 1995.

[54] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[55] *IEEE Std 802.11, 1999 Edition*. 1999. http://standards.ieee.org/catalog/olis/lanman.html.

[56] OpenSSL. http://www.openssl.org/.

[57] J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure Network Coding for Wireless Mesh Networks: Threats, Challenges, and Directions. *Computer Communications*, 32(17):1790–1801, 2009.

[58] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen. RIPPLE Authentication for Network Coding. In *Proceedings of the Conference on Computer Communications*, pages 1–9, 2010.

[59] T. Ho, B. Leong, R. Koetter, M. Medard, M. Eros, and D. R. Karger. Byzantine Modification Detection in Multicast Networks using Randomized Network Coding. In *Proceedings of the Symposium on Information Theory*, volume 54, pages 2798–2803, 2004.

[60] M. Kim, M. Médard, and J. Barros. A Multi-hop Multi-source Algebraic Watchdog. *Computing Research Repository*, pages 1–5, 2010.

[61] M. Kim, M. Médard, a. Barros, Jo and R. Kötter. An Algebraic Watchdog for Wireless Network Coding. In *Proceedings of the Symposium on Information Theory*, pages 1159–1163, 2009.

[62] C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad Hoc Networks*, 1(2):293–315, 2003.

[63] B. Yu and B. Xiao. Detecting Selective Forwarding Attacks in Wireless Sensor Networks. In *Proceedings of the Parallel and Distributed Processing Symposium*, pages 8–15, 2006.

[64] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of the Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[65] J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure High-Throughput Multicast Routing in Wireless Mesh Networks. *Transactions on Mobile Computing*, 10(5):653–668, 2010.

[66] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation Systems. *Communications of the ACM*, 43(12):45–48, 2000.

[67] S. Buchegger and J. Le Boudec. A Robust Reputation System for Mobile Ad-Hoc Networks. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, pages 1–11, 2004.

[68] L. Rizzo and L. Vicisano. RMDP: An FEC-based Reliable Multicast Protocol for Wireless Environments. *Mobile Computing and Communications Review*, 2(2):23–31, 1998.

[69] A. O'Donnell and H. Sethu. On Achieving Software Diversity for Improved Network Security Using Distributed Coloring Algorithms. In *Proceedings of computer and communications security*, pages 121–131, 2004.

[70] M. Krishnamoorthy and B. Krishnamurthy. Fault Diameter of Interconnection Networks. *Computers & Mathematics with Applications*, 13(5):577–582, 1987.

[71] S. Latifi. On the Fault-Diameter of the Star Graph. *Information Processing Letters*, 46(3):143–150, 1993.

[72] S. Latifi. Combinatorial Analysis of the Fault-Diameter of the N-Cube. *Transactions on Computers*, 42(1):27–33, 1993.

[73] K. Day and A. Al-Ayyoub. Fault Diameter of K-ary N-Cube Networks. *Transactions on Parallel and Distributed Systems*, 8(9):903–907, 1997.

[74] M. Hajiaghayi, N. Immorlica, and V. Mirrokni. Power Optimization in Fault-Tolerant Topology Control Algorithms for Wireless Multi-Hop Networks. In *Proceedings of the Conference on Mobile Computing and Networking*, pages 300–312, 2003.

[75] X. Jia, D. Kim, S. Makki, P. Wan, and C. Yi. Power Assignment for K-Connectivity in Wireless Ad Hoc Networks. *Journal of Combinatorial Optimization*, 9(2):213–222, 2005.

[76] D. Panigrahi, P. Duttat, S. Jaiswal, K. Naidu, and R. Rastogi. Minimum Cost Topology Construction for Rural Wireless Mesh Networks. In *Proceedings of the Conference on Computer Communications*, pages 107–120, 2008.

[77] S. Çamtepe and B. Yener. Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks. *Transactions on Networking*, pages 293–308, 2007.

[78] L. Oliveira, H. Wong, M. Bern, R. Dahab, and A. Loureiro. SecLEACH-A Random Key Distribution Solution for Securing Clustered Sensor Networks. In *Network Computing and Applications*, pages 145–154, 2006.

[79] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of Security and Privacy*, pages 197–213, 2003.

[80] H. Chan and A. Perrig. PIKE: Peer Intermediaries for Key Establishment in Sensor Networks. In *Proceedings of the Conference on Computer Communications*, volume 1, pages 524–535, 2005.

[81] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili. A Pairwise Key Predistribution Scheme for Wireless Sensor Networks. *Transactions on Information and System Security*, 8(2):228–258, 2005.

[82] J. Rohrer, A. Jabbar, and J. Sterbenz. Path Diversification for Future Internet End-to-End Resilience and Survivability. *Springer Telecommunication Systems*, 56(1):49–67, 2012.

[83] K. Zhao, X. Chu, M. Wang, and Y. Jiang. Speeding Up Homomorphic Hashing using GPUs. In *Proceedings of the Conference on Communication and Computing*, pages 1–5, 2009.

[84] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure Network Coding over the Integers. In *Proceedings of the Conference on Practice and Theory of Public-Key Cryptography*, pages 142–160, 2010.

[85] A. Le and A. Markopoulou. Locating Byzantine Attackers in Intra-Session Networking Coding Using SpaceMac. In *Proceedings of the Symposium on Network Coding*, pages 1–6, 2010.

[86] G. Liang, R. Agarwal, and N. Vaidya. When Watchdog Meets Coding. In *Proceedings of the Conference on Computer Communications*, pages 1–9, 2010.

[87] W. Wang, J. Kong, B. Bhargava, and M. Gerla. Visualisation of Wormholes in Underwater Sensor Networks: A Distributed Approach. *Journal of Security and Networks*, 3(1):10–23, 2008.

[88] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In *Proceedings of the Conference on Computer Communications*, volume 3, pages 1976–1986, 2003.

[89] L. Hu and D. Evans. Using Directional Antennas to Prevent Wormhole Attacks. In *Proceedings of the Symposium on Network and Distributed Systems Security*, 2004.

VITA

VITA

Andrew Newell received his BS in computer science and mathematics at Southern Illinois University at Carbondale in 2008. He received his PhD in computer science at Purdue University in 2014. He was a member of the Dependable and Secure Distributed Systems laboratory at Purdue University. His research interests are in resilient network design, wireless networks, and network coding. He went to work at Facebook after graduation.