SECURING MULTI-CHANNEL WIRELESS NETWORKS AGAINST MALICIOUS BEHAVIOR

CHAODONG ZHENG

(B.Eng)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any other degree in any university previously.

Name: ZHENG Chaodong Date: September 11th, 2015

Summary

Wireless networks are becoming increasingly popular over the last two decades, and there is no doubt this trend will continue. The key advantage of wireless networks is that they utilize radio waves to transmit information over the air, hence allowing related devices to communicate in a cordless manner. Nevertheless, the open and shared nature of wireless networks' communication medium also makes them more vulnerable to various malicious behavior. These malicious behavior include, but are not limited to, jamming, spoofing, and *sybil attacks*.

Sybil attacks refer to the situation in which malicious users dishonestly generate large numbers of fake identities, and inject them into the network to gain unfair advantage over honest users, or to conduct other hostile activity. Compare with jamming and spoofing attacks, sybil attacks are relatively new, and are not so well-studied, especially in the environment of wireless networks. In this thesis, we focus on the topic of how to effectively thwart sybil attacks in multi-channel wireless networks, and at the same time tolerate other malicious behavior as well.

We first consider *centralized* multi-channel wireless networks, in which one central and trusted base station exists. The problem is to enforce *fairness* when multiple users are downloading data from the base station. In particular, without special care, malicious users can commence a sybil attack by simulating many fake identities, and hence obtain a large and unfair portion of the total bandwidth. To counter such behavior, we propose a protocol named SYBILCAST. SYBILCAST limits the number of fake identities, and in doing so, it ensures that each honest user gets at least a constant fraction of its fair share of the bandwidth. As a result, each honest user can complete his or her data download in asymptotically optimal time. A key aspect of this protocol is balancing the rate at which new identities are admitted and the maximum number of fake identities that can co-exist, while keeping the protocol overhead low.

We then consider a more challenging scenario: *ad hoc* multi-channel wireless networks, where *no* central base stations exist. The problem in this setting is for each user to learn the identities of the other users, even though they have no prior knowledge of the number of other users or their identities. To solve this problem, several new anti-sybil algorithms are described and analyzed. They guarantee each honest user accepts a set

of trusted and unforgeable identities that include all other honest users and a bounded number of fake identities. The proposed algorithms provide trade-offs between time complexity and sybil bounds. It is also worth noting that these algorithms solve, as subroutines, two problems of independent interest in this anonymous wireless setting: Byzantine consensus and network size estimation.

It is worth noting that all the above mentioned algorithms are randomized algorithms that can only guarantee correctness *with high probability*. Towards the end of the thesis, we study if such small chance of error is inevitable. In particular, we focus on the problem of *counting* and *node discovery*, and show that in some adversarial environments, even without sybil attacks, it is impossible to guarantee to solve these problems: chance of error may exist, or the algorithm may never terminate.

Acknowledgments

First and foremost, I must sincerely thank my advisor, Dr. Seth Gilbert. It has been a great honor and pleasure to be his Ph.D. student. Over the past five years, he has led me into the amazing world of distributed computing, wireless networking, and randomized algorithms. Frankly speaking, theoretic computer science was not my strength during my undergraduate study. Without Dr. Gilbert's guidance, knowledge, patience, and strong sense of responsibility, I will not be able to stand where I am here today. To this, I feel really grateful. Dr. Gilbert has also taught me how to be a research scientist, or more importantly, how to think critically and rigorously, with both language and action. I believe these valuable skills will be helpful for me regardless of my future career path.

I would then like to thank my fellow friends at the CIR lab and the I^3 graduate lab. Sometimes, a Ph.D. student's life can be dull and boring. It is often the encouragements and inspirations from them that help me pass these difficult times.

I would also like to thank the university and the Ministry of Education of Republic of Singapore, for providing the funding and a cozy environment for me to pursuit the degree. Singapore is a great place to live and study. With the scholarship (and the funding from my advisor in the fifth year), I was to able to focus on research activities, and enjoy a comfortable life. To this, I also feel grateful.

Finally, I devote my special thanks to my parents. They have always been there for me, making me know I always have my family to count on when times are rough.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

In recent decades, the popularity of wireless telecommunication has been ever increasing. Thanks to its cordless nature, wireless telecommunication allows devices to communicate with each other without the need for physical wires; it also enables users to stay in touch even when they are on the move.

The key technology behind the scene that makes wireless communication possible is that information can be encoded in radio waves and then transmitted. This observation reveals a key difference between wireless and wired networks: all wireless clients in a network can communicate over the *same* medium, while each pair of wired clients in a network may have to rely on a *separate* cable to communicate with each other.

Unfortunately, this *open* and *shared* nature of wireless telecommunication also makes wireless networks more vulnerable to various malicious behavior than traditional wired networks. For example, if an evil user wants to eavesdrop, intercept or disrupt the communication within a wired network, he or she usually has to obtain physical access to the network infrastructure. In the wireless setting, however, all such malicious behavior can be done tens or even hundreds of meters away from where the signal is emitted or received. Therefore, understanding the attacking strategies a malicious user may use, and the corresponding countermeasures, is a crucial step to securing wireless networks.

In this thesis, three types of malicious behavior are considered: *jamming*, *spoofing*, and *sybil attacks*; with sybil attacks being the focus. Jamming refers to the behavior of

emitting radio signals to disrupt the normal communication of other wireless devices. It is an attack on availability, and can potentially render the whole network unable to serve any request. Moreover, it is easy to implement yet hard to defend against [27]. Spoofing, on the other hand, refers to a situation where a malicious user masquerades as another, thereby gaining an illegitimate advantage. It is an attack on authenticity.

Compare to jamming and spoofing, sybil attacks [16] is a relatively new security threat. It refers to the behavior where malicious users dishonestly generate large numbers of fake identities (also called *sybil identities*) and inject them into the network to gain some benefit or to conduct some hostile activity. Researchers have shown that sybil attacks can do severe damage to various real world systems [42]. Nevertheless, how to effectively counter it, especially in the context of wireless networks, is not fully understood. In this thesis, we focus on the topic of effectively thwarting sybil attacks in multi-channel wireless networks, while at the same time tolerating jamming and spoofing as well. One point worth noting is that spoofing attacks share some similarities with sybil attacks as they both are related to masquerading identities. However, fundamental differences exist between them. In particular, spoofing attacks focus on camouflaging existing real entities, whereas sybil attacks focus on fabricating new identities that do not correspond to any existing real entities.

In the reminder of this chapter, we will first introduce several general strategies that will be used to counter the above mentioned various malicious behavior and the corresponding challenges when applying or generalizing them. We will then present the results and contributions. We conclude this chapter by introducing the organization of the reminder of the thesis.

1.2 Approach and Challenges

Although wireless networks are vulnerable to the above mentioned malicious behavior, some of its unique characteristics can be leveraged to defend against these threats.

Sybil attacks. The existence of *multi-channel* wireless networks is one such feature that can help thwart sybil attacks. Modern wireless telecommunication standards usually defines multiple communication channels. (For example, the IEEE 802.11 WiFi standard [1] defines 14 channels around 2.4GHz, and the Bluetooth standard [2] defines

79 channels.) Wireless devices that are equipped with radio transceivers can access any of these channels, with respect to the corresponding standard. However, at any given time, a radio transceiver (usually) can only operate on one channel. If an adversary controls a limited number of such radio transceivers, then we can take advantage of this limitation and effectively thwart sybil attacks. More specifically, we make two key assumptions regarding the power of the adversary: (a) each malicious device that is controlled by the adversary has a single radio transceiver that can only tune into a single channel at a time; and (b) the total number of malicious devices that is controlled by the adversary is less than the total number of available channels. The first assumption derives from the idea that the adversary has the same hardware available as the honest devices (and, notably, hardware similar to standard wireless devices today). The second assumption derives from the idea that—as we have previously mentioned—modern wireless communication standards usually specify multiple communication channels, and there is every reason to believe future standards will have even more channels available for use (see, e.g., [26, 58]). Given these two assumptions, devices can test potential identities by requiring them to participate on certain channels at certain times. Because the adversary is limited in the number of channels she can use concurrently, if these tests are constructed carefully, they can limit the number of identities the adversary can convincingly maintain.[1]

To make the idea clearer, we now give a simple example. Consider a wireless network consisting of two channels and at most one malicious device v. An honest node u can apply the following method to verify two identities v_1 and v_2: u first assigns v_1 and v_2 to go to different channels; then u asks each identity to send out some information. Finally, u randomly picks one of the two channels and listens. If the two identities are each generated by an honest node, then u can always hear something on the channel chosen by itself. Otherwise, if both v_1 and v_2 are generated by v (i.e., v_1 and v_2 are sybil identities), then v only has a 50% chance to correctly guess which channel u will listen to, and broadcast on that channel. As a result, there is a 50% chance u will be able to spot and eliminate one sybil identity. If this test is repeated for sufficiently many times, all sybil identities can be detected.

[1] Notice, if there are more malicious devices than channels, then having access to many channels does not help: the malicious users can access all the channels simultaneously and any advantage is lost (both for the purposes of avoiding disruption, and for the purpose of sybil avoidance).

The above illustrated strategy is called *radio resource testing* [39, 40, 42]. The networking community recognizes this approach as practical because most real world wireless devices have access to many more channels than they can use simultaneously, and it does not require any additional information.

Nevertheless, when generalizing the aforementioned simple strategy, many challenges arise. First and foremost, the process of testing must be implemented efficiently with limited overhead: there is little benefit to eliminating the sybil identities if a protocol for detecting these identities consumes all the bandwidth and/or time. For example, it does not suffice to test one pair of identities at a time: if there are d identities (many of which may be sybil identities), it will take $\Theta(d^2)$ time to test them all. Instead, the running time should depend on the number of real users (not the number of sybil identities). Otherwise, the adversary can potentially swamp the system with never-ending tests. In addition, the adversary may control more than one malicious device and hence can access multiple channels simultaneously. Thus, it will be insufficient to test identities on a per pair basis.

Another challenge that can arise is the continuous nature of some systems. In many scenarios—such as wireless file servers that are serving data download requests—clients can arrive continuously over time. As a result, the adversary can also continuously create new sybil identities. In these cases, it is insufficient to simply run a set of tests and then proceed with a standard protocol. In fact, for some of our proposed algorithms, a key aspect of the design is to balance the rate at which new users (and potentially new sybil identities) are admitted to the system and the rate at which sybil identities can be detected.

Jamming. The existence of multiple communication channels also helps in terms of jamming resistance. For example, if two honest devices manage to share some secrets that are unknown to the adversary with each other (or more simply, they have pre-distributed shared credentials), then these secrets can be used to generate channel hopping sequences (via, for example, a pseudorandom number generator [33]). Since the adversary does not know these channel hopping sequences, the probability that she can utilize her radio transceivers to disrupt the communication is greatly reduced.

This strategy is called *coordinated* frequency hopping, in a sense that honest devices

know a common frequency hopping pattern. In many cases, however, honest devices do not have shared secrets a prior; and may each follow a different frequency hopping pattern to mitigate adversarial jamming. This later method is usually called *uncoordinated* frequency hopping [57]. Compare with coordinated frequency hopping, the uncoordinated version does not require common knowledge in advance. Nevertheless, such schemes need to be carefully designed to ensure that honest devices can still meet sufficiently often, otherwise the participants will not be able to exchange information efficiently (which would be no better than taking no countermeasures at all).

In this thesis, we use both coordinated and uncoordinated frequency hopping to mitigate the impact of adversarial jamming.

Spoofing. In the context of network security, spoofing refers to the situation in which an entity successfully masquerades as another entity and communicates with others on this stolen identity's behalf. Spoofing is a well-known security problem and exists in many scenarios. It is an attack on authenticity, but can target confidentiality, availability, privacy, etc. In this thesis, we utilize standard cryptography tools such as cryptographically secure hash functions and asymmetric encryption to prevent spoofing attacks.

1.3 Results and Contributions

In this thesis, we develop algorithms that can effectively thwart sybil attacks (and other malicious behavior) in centralized and ad hoc settings, respectively. We have also shown that certain related problems are impossible to solve with probability one. (I.e., no algorithms can *guarantee* to solve those problems.)

Sybil resistance in centralized wireless networks. We begin by studying sybil resistance in centralized multi-channel wireless networks in which a central and trusted base station exists. We consider the problem of *fair* bandwidth allocation. More specifically, we assume multiple users are downloading data from a central base station, and these users can come and leave over time. The base station allocates equal bandwidth to all users that are currently in the system. Unfortunately, without special care, an adversary can easily circumvent this fairness mechanism and obtain an unfair share of the total bandwidth by simulating multiple users (i.e., commence a sybil attack). For example,

in a wireless network that has ten users, one malicious user can obtain 90% of total bandwidth (instead of 10%) by simulating 90 fake users.

To thwart such malicious behavior, we develop a new protocol named SYBILCAST. More specifically, the SYBILCAST protocol provides the following guarantee: in a single-hop wireless network with at most N real users, among which at most t users are dishonest, and $c > 20t$ channels, if an honest user p requests a download m of size $|m| \geq c^2 \log^3 N$, and if there are at most $\texttt{contention}(p, m)$ concurrently active real users at any point during the request, then with high probability the download will complete in $O(|m| \cdot \texttt{contention}(p, m))$ time.[2] Here, *with high probability* means with probability at least $1 - 1/N^k$, for any constant $k \geq 1$. This implies that each honest user will receive an asymptotically fair share of the total bandwidth, even when sybil attacks are present. We also note here, for smaller requests (i.e., those with size smaller than $c^2 \log^3 N$), with high probability SYBILCAST still guarantees all their data will be delivered, but the time consumption may not be asymptotically optimal.

To the best of our knowledge, SYBILCAST is the first protocol that can provide (asymptotically) fair bandwidth allocation when sybil attacks are present. It can tolerate message spoofing and wireless jamming as well. We believe that the techniques developed here may be useful in constraining sybil identities in other types of problems. In particular, the SYBILCAST approach has relatively low overhead, and hence can integrate well with other protocols.

Sybil resistance in ad hoc wireless networks. We then turn our attention to ad hoc networks in which no central and trusted base station exist. We consider the problem of neighbor discovery in *unknown* and *anonymous* wireless networks. More specifically, we assume a group of wireless devices are activated in a single-hop wireless network. Some of them are honest while others are malicious. The honest devices are provided with no advance information regarding the size of the network or the identities of the other devices. The goal is to let each honest device construct a set of trusted and unforgeable identities that includes: (a) a unique identity for each honest device, and (b) a bounded number of sybil identities.

Intuitively, this scenario seems hopeless, as honest devices have no information re-

[2]The size of m is measured in the number of data packets.

garding network size or participants, and there is no obvious way to distinguish honest devices from malicious devices. Thus, malicious devices can commence a sybil attack and create many fake identities. Nevertheless, we prove—perhaps surprisingly–that this scenario is *not* hopeless. We describe and analyze new algorithms that allow honest devices to constrain the number of sybil identities to an asymptotically optimal limit. Our results answer open questions regarding sybil resistance in ad hoc networks and provide a general foundation for establishing trusted computing in the increasingly untrustworthy world of wireless networking.

More specifically, we model a single-hop network with n devices, t of which are malicious, using c channels, where $t \leq \min\{n, c\}/\alpha$, for some sufficiently large constant α. We assume the honest devices do not know n or t in advance. We describe and analyze new anti-sybil algorithms that guarantee, with high probability, each honest device ends up accepting a set of trusted and unforgeable identities that include all other honest devices and a bounded number of sybil identities. The algorithms provide trade-offs between time complexity and sybil bounds. In more detail, our first algorithm terminates in $O(n \lg^2 n)$ time and guarantees no more than $O(t)$ sybil identities (which is asymptotically optimal) for the case where $c \geq n$. On the other hand, when $c < n$, the time bound can grow as large as $O(n^2 \lg^2 n)$ and the sybil bound as large as $O(n)$. We then describe a slower algorithm that terminates in $O(n^3 \lg^2 n)$ time, but always guarantees an asymptotically optimal bound of $O(t)$ sybil identities.

We note here, these algorithms solve, as subroutines, two problems of independent interest in this anonymous wireless setting: Monte Carlo Byzantine consensus and network size estimation.[3]

Impossibility results. As can be seen from previous discussion, all the proposed algorithms can only guarantee their properties with high probability. This implies, with certain extremely small (but non-zero) probability, these algorithms will fail to enforce their guarantees, and the adversary will succeed in conducting various malicious behaviors. This observation leads to a nature and important question: is such chance of error a result of imperfect design (and/or analysis), or simply inherently inevitable (due to the nature of the problem).

[3]The consensus protocol maintains its safety properties with high probability, but not necessarily with probability one.

To answer this question, we consider two fundamental problems that constantly arose when we were trying to design the previously discussed algorithms: the neighbor discovery problem in which each device wants to know the identities of the other devices that co-exist in the network (i.e., *who* is around); and the easier counting problem in which each device wants to know how many devices in total exist in the network (i.e., *how many* devices are around).

We find that even without sybil attacks, the malicious nodes can still render these problems unsolvable in many settings. In particular, we show that for any finite time period, no algorithm exists that can *guarantee* to solve these problems within the given time period. Moreover, we have also shown that no Las Vegas algorithms exist that can guarantee to solve these problems. (That is, no randomized algorithms can guarantee to solve them within a finite expected running time.)

1.4 Organization of the Thesis

The reminder of this thesis consists of five chapters. In Chapter 2, we will give a detailed literature review on related topics. In Chapter 3, we will discuss more details regarding sybil resistance in centralized wireless networks, and the SYBILCAST protocol. In Chapter 4, we will discuss more details regarding sybil resistance in ad hoc wireless networks, and the various distributed protocols we have proposed. In Chapter 5, we will present the impossibility results. Finally, we conclude in Chapter 6, and discuss potential future research directions.

Chapter 2

Related Work

This chapter presents a short survey which outlines the state of art defense strategies for three types of malicious behavior in the context of wireless networks: sybil attacks, jamming attacks, and spoofing attacks. In the analysis of each of these malicious behaviors, we will firstly define the concept of the attack and introduce the problems it may cause. We will then present the various countermeasures researchers have proposed. These countermeasures will be divided into different categories, each of which tries to thwart the specific attack from a different perspective. While analyzing each countermeasure, we will discuss if it can be applied to other scenarios (e.g., wired networks), or is unique to wireless networks. We will also highlight the advantages and limitations of each approach. At the end of the discussion, a short summary is provided to compare and contrast the different countermeasures. We will also point out potential future research directions whenever possible.

Notice, the focus of this thesis is thwarting sybil attacks, and we include literature review on jamming and spoofing attacks mainly for the purpose of completeness. We also note here, spoofing is a well-studied problem in the field of information security, hence the discussion on it will be short and concise, interested readers can refer to other survey papers for more details.

2.1 Basic Model

In this section, we introduce some most basic concepts and notations that will be frequently used when describing wireless networks.

In wireless networks, wireless clients use radio waves to transmit information. Different radio waves have different frequencies, and we usually call a certain frequency range as one *channel*.[1] In a wireless network, wireless clients can use one or multiple channels to transmit information. In particular, a wireless client can emit signals on a channel to *send* a message, or sense signals to try to *receive* a message. In most cases, channels are "orthogonal", and different clients sending messages on different channels will not interfere with each other. However, if multiple clients send messages on the same channel simultaneously, a *collision* may happen, i.e., different radio signals may interfere with each other. There are different proposals for modeling collisions. For example, some collision models assume all related radio signals are corrupted and hence no information can be delivered, while other collision models assume one signal "overwrites" other signals. Under different *collision detection* models, collisions may or may not be detected by the communicating parties. Collision modeling also concerns *who* can detect collisions and whether *recover* information from collisions is possible or not.

Similar to wired networks, wireless networks can be divided into *single-hop* wireless networks or *multi-hop* wireless networks. In a single-hop wireless network, all wireless clients can *directly* communicate with each other. By contrast, in a multi-hop wireless network, information may have to be *relayed* by several wireless clients before it can reach the final destination.

Synchronization is another important parameter for wireless networks. We say a wireless network is synchronized if time can be divided into continuous, equal-sized units, and each wireless client in the network has same view on the beginning and ending of each of these time units. Notice, a synchronized network does not necessarily imply the clients in the network have the same clock. I.e., in a synchronized network, clients may have different opinions on what the current time is.

2.2 Sybil Attacks

2.2.1 Overview

A *sybil attack* [16] refers to the situation in which malicious users dishonestly generate large numbers of fake identities (also called *sybil identities*, or simply *sybils*), injecting

[1] Notice, in wireless telecommunication, there also exist other definitions for "channel".

them into the system or network to gain extra benefit or to conduct some other hostile activity. This name is derived from the title of a book: "Sybil" by Schreiber [50], which describes a woman with multiple personalities.

Researchers believe sybil attacks can threat many real-world systems, which include, but are not limited to, reputation systems, peer-to-peer networks, and wireless networks. In [42], Newsome et al. discuss the various attacks the adversary can perform once she has generated enough sybil identities. They summarize the potential targets of these attacks (particularly in sensor networks) as follows: distributed storage, routing, data aggregation, voting, fair resource allocation, and misbehavior detection. For instance, in an attack on a distributed storage system, data may be stored only on the sybil identities that are generated by a single malicious node, even though the original design goal is to replicate or distribute the data across multiple different nodes. (In fact, when the term sybil attack was first coined by Douceur in [16], the problem he considered was sybil attacks' ability to violate the replication property in peer-to-peer systems.)

To better understand sybil attacks, Newsome et al. [42] divide them into different categories from three orthogonal dimensions: direct vs indirect communication, fabricated vs stolen identities, and simultaneity. In the first dimension, *direct communication* means sybil identities can "directly" exchange messages with honest nodes while *indirect communication* means all information exchanged between sybil identities and honest nodes must be "routed" by malicious nodes. In the second dimension, *fabricated identities* means the malicious nodes can directly create new sybil identities while *stolen identities* means the malicious nodes can only steal legitimate identities (e.g., by destroying or disabling the targeted honest nodes) and assign them to sybil identities.[2] In the last dimension, *simultaneous* means all the sybil identities can be present in the network simultaneously, while *non-simultaneous* means the malicious nodes can only utilize a limited portion of the sybil identities they have generated at any given time.

Researchers have also proposed many solutions to thwart sybil attacks. Most of these proposals fall into one of the following three categories: (a) *pre-distributing credentials*, (b) *social networking*, and (c) *resource testing*. Except for radio resource test-

[2]Whether identity theft should be counted as a form of sybil attack is an open question as it looks almost identical to spoofing.

ing, which belongs to the resource testing category, all other strategies can be directly applied to scenarios other than wireless networks.

We also note here, aside from the three major approaches mentioned above, there also exist several papers which takes a less common *radio fingerprinting* approach. This approach is specific to wireless networks, and cannot be applied to other scenarios. We will also briefly discuss it in this thesis for the sake of completeness.

2.2.2 Countermeasures

Pre-distributing Credentials

This approach is the simplest and most straightforward one, and people have been using such techniques (e.g., passwords, signatures/certificates) to authenticate identities in similar scenarios for a long time. For example, by using a *public key infrastructure (PKI)* [3], we can require each identity in the network to hold a digital certificate that is signed by a trusted authority. Through careful management of certificate creation, the adversary cannot fabricate sybil identities easily. Meanwhile, the built-in revocation mechanism of PKI systems also helps thwart identity theft. Unfortunately though, such PKI style solutions suffer from two potential shortcomings. Firstly, creation and revocation of certificates is a relatively slow process if the network is highly dynamic and rapidly changing. Secondly, asymmetric cryptography is traditionally believed to be "expensive" in terms of the computational resources required, hence schemes extensively using it may not be suitable for systems like sensor networks which are highly energy constrained. Nevertheless, we notice that in recent years, there are many attempts to "port" public key encryption to sensor networks in an energy-efficient manner (e.g., [35, 62]). Moreover, there also exist benchmark results which indicate public key encryption is actually not as power-consuming (e.g., [44, 61]) as previously believed.

Binding pre-distributed keys with each node's identity is another possibility. In [42], the authors propose three such schemes. The first one is called *key pool*. In this scheme, each node is given a set of keys (in the key pool) based on its identity. To validate an identity, the challenger asks the identity to demonstrate it really possesses the keys it should have owned. The key pool scheme is simple, but suffers a major drawback: it does not support pairwise authentication as it cannot guarantee each pair of identi-

ties share at least one common key. This shortcoming motivates a second scheme: the *single-space pairwise key distribution* scheme. This second scheme supports pairwise authentication, but in turn allows the adversary to create arbitrarily many sybil identities, if the adversary has compromised enough honest nodes. Towards the end of the paper, the authors combine these two schemes together and propose the *multi-space pairwise key distribution* scheme, which can provide the strongest security guarantees.

Although pre-distributing keys can effectively thwart sybil attacks, it has the same disadvantage as the PKI approach; namely, pre-distributing credentials in advance is difficult to implement in highly dynamic and rapidly changing wireless networks.

Social Networking

The emergence of social networks has inspired new techniques for detecting sybil identities. In [28], Hoffman et al. suggest *Pretty Good Privacy (PGP)* [71], a program which was originally developed for the purpose of confidential message exchange with guarantees on privacy and authenticity, is also suitable for countering sybil attacks. In contrast to PKI systems' centralized and hierarchical approach, PGP can create a decentralized and flat structure, called a "*web of trust*".[3] More specifically, PGP binds certificates (which are *not* signed by a trusted authority) with usernames and/or E-mail addresses. Users that trust each other in the real world can also choose to trust each other's certificates. In this way, trust relationships can *propagate*. For example, suppose Alice and Bob are friends and they trust each other. We further assume Bob trusts Cathy. Then, Bob can sign Cathy's certificate with his private key. Later on, when Alice sees Cathy's certificate, she may have a higher chance of trusting it since it is signed by Bob, a friend of hers. Notice, this is different from the PKI approach since Bob is *not* a universally trusted authority. For a user that does not trust Bob, Bob's signature on Cathy's certificate does not say much. These properties allow PGP to thwart sybil attacks since each sybil identity's certificate will not be signed by many, if any, real identities.

More recently, Yu et al. leverage the unique properties of social networks and propose two novel protocols named SYBILGUARD [70] and SYBILLIMIT [69]. SYBILLIMIT features many optimizations over SYBILGUARD while the insights behind them are similar. Hence, we focus our attention on SYBILLIMIT here. Imagine a distributed

[3] Current version of PGP also supports PKI style approach.

system $\mathcal{D}$ which contains a social network $\mathcal{G}$. $\mathcal{G}$ is an undirected graph in which each node represents an identity and each edge represents a human-established trust relationship. The *honest region* of $\mathcal{G}$ is the subgraph of $\mathcal{G}$ which contains all the honest nodes and the edges among them. The *malicious region* is similarly defined. An edge connecting the honest region and the malicious region is called an *attack edge*.

The SYBILLIMIT protocol works in the following way: a node u first performs $\Theta(\sqrt{m})$ independent random walks in $\mathcal{G}$ each of length $O(\log n)$. Here, n is the total number of honest nodes and m is the total number of edges in the honest region. Define the tail of a random walk to be the last step in the walk; thus u will have $\Theta(\sqrt{m})$ tails. Now, for any other identity v that has performed the same procedure, u labels v as "sybil" if and only if they have no common tails. The intuition behind this protocol is simple. In particular, the speed at which a random walk approaches the stationary distribution can be measured by a metric called the *mixing time*. Moreover, random walks that cross attack edges are likely to have large mixing time. Hence, if the length of the random walks is set to a relatively small value, then the graph generated by these random walks should mostly consist of only honest nodes.

Since Yu's approach is not complex, and can provide good guarantees on sybil bounds, there exist much follow up research: SYBILINFER [9], GATEKEEPER [59], and SUMUP [60]. For a more detailed discussion on these proposals, please refer to Yu's survey paper [68].

One drawback of the social networking approach is that it may result in privacy concern, as it requires real world social knowledge. As a result, this approach may not be practical to all systems.

Resource Testing

The basic idea behind resource testing is to demand a client to prove that it has a sufficient quantity of some resource available. If an adversary wants to simulate two identities, then she must demonstrate twice as many resources. In this way, a resource constrained adversary cannot simulate too many identities. However, as proved by Douceur in [16], such validation on identities must be carried out simultaneously, otherwise even a single adversary can still generate an arbitrarily large number of sybil identities. The resource in question can be computational power, storage, or radio transceivers.

Computational resource testing (CRT) was first introduced by Dwork et al. [18] to counter E-mail spam. Later on, it was also used as a defense strategy against denial-of-service attacks [6]. In 2005, Aspnes et al. [4] proposed to use this technique as a mechanism for thwarting sybil attacks.

In a computational resource test, a client is required to solve a moderately hard cryptographic problem in a given amount of time. Moreover, the specified problem is only solvable by brute force calculation. Since an adversary cannot have infinite computational power, the number of such cryptographic problems, and hence the number of sybil identities she can create, is limited. Some of the most frequently used cryptographic puzzles include: *hash reversal* [29], *partial hash collision* [6] and *time lock* [48]. In a hash reversal test, the identity being challenged is required to calculate the reversal, or the *pre-image*, of a hash value. Moreover, most bits of the pre-image are given to the identity with only a few left blank. The identity being challenged need to perform a brute force search until the correct answer is found. Both the generation and the validation of such puzzles are easy, hence it incurs little overhead on the challenger. In a partial collision test, the identity being challenged is given a random one-time *nonce* and is required to find a string of unknown length, such that the first b bits of the hash of the concatenation of the nonce with that string are 0. The identity being challenged needs to perform a brute force search until an answer is found. Compared with hash reversal, partial hash collision is slightly better in terms of performance since the challenger only needs to create a nonce before it can construct one such puzzle whereas in the hash reversal test an additional hash calculation is required. One potential problem of this test, however, is that the computational resource required to solve it is unpredictable since in theory it may take infinite time to find even one correct answer. In practice, however, this is not a big concern as the probability of taking a significantly longer time than the expected value will decrease rapidly toward zero. Lastly, in a time lock test, the challenger encrypts a nonce with a specially designed cryptographic scheme. The fastest decryption method is a sequential squaring process. This property forces the identity being challenged to compute in a tight loop for a predictable amount of time. Compared with the previous two methods, this approach enforces a fixed amount of work. However, it consumes more computational resources on the challenger side and hence may potentially lead to denial-of-service attacks.

Table 2.1: Comparison of different types of computational resource test.

	Hash Reversal	Partial Hash Collision	Time Lock
Variance	probabilistic bounded	probabilistic unbounded	fixed
Interactivity	interactive	both	interactive
Known Solution	yes	no	yes
Public Auditability	yes	yes	no
Public Trust	no	no	no

In [38], Mónica compares and contrasts these three types of tests by discussing the properties an ideal computational resource test should have. More specifically, Mónica enumerates the following five properties: (a) whether the *variance* on the computational resource required is low; (b) whether the challenger needs to supply parameters to the identity being challenged, i.e., whether *interactivity* is required; (c) whether there is *known solution* to the proposed test; (d) whether other nodes can *audit* the testing process; and (e) whether other nodes can *trust* the result of the test. We summarize the conclusion of the discussion in [38] in Table 2.1.

Computational resource test can effectively and reliably thwart sybil attacks, provided that the adversary indeed has a known, limited amount of computational resources. A disadvantage of this type of test, however, is that it requires detailed (i.e., fine grained) assumptions on the adversary's computational power to obtain a bound on the number of sybil identities she can create. (In fact, in many other situations or scenarios, we have made coarse grained assumptions on adversary's computational power. For example, when applying encryption algorithms, we usually assume the adversary cannot decrypt ciphertext without knowing the key.) Another potential disadvantage of computational resource testing is that it may quickly consume a lot of energy, which can make it unsuitable for environments such as sensor networks.

Radio resource testing (RRT) is another widely used resource testing technique. As previously discussed, it relies on two key assumptions: (a) each malicious device that is controlled by the adversary has a single radio transceiver that can only tune into a single channel at a time; and (b) the total number of malicious devices that is controlled by the adversary is less than the total number of available channels.

Radio resource testing was first proposed by Newsome et al. in [42] as a technique to thwart sybil attacks in multi-channel wireless sensor networks. In that paper, the authors propose the following simple scheme: assume a node u wants the verify if

there are sybil identities among its n neighbors. It can assign each of these n neighbors to a different channel and then ask them to broadcast a message at the same time. It can then choose a random channel to listen on. If the neighbor that is assigned to the chosen channel is an honest node, then u should hear a message. On the contrary, if that neighbor is a sybil identity, then with some probability it will not be able to broadcast the message and hence be spotted by u. We say "with some probability" here because the adversary does not know which channel node u will listen to, hence chances exist that the adversary will not be able to broadcast on the channel that is chosen by u (due to the limitation on the number of radio transceivers the adversary has).

The above mentioned scheme, which is quite similar to the one we introduced in Chapter 1, is just one type of radio resource test. It is called *simultaneous sender test (SST)*. In [39], Mónica et al. propose another two new types of radio resource tests, namely, *simultaneous receiver test (SRT)* and *forced collision test (FCT)*. Simultaneous receiver test operates in the following way: as in a simultaneous sender test, a node u assigns the neighbors it wants to verify each to a different channel. Then, instead of asking them to broadcast a message simultaneously, u asks them to listen on their respective channel and itself broadcasts a message on a randomly chosen channel. After that, u will ask the neighbor who should have received the message to repeat it and labels that neighbor as sybil if it fails to do so. Compared with simultaneous sender test, simultaneous receiver test significantly decreases the message complexity: at most one neighbor (that is being challenged) needs to transmit during the test.

The last type of radio resource test—i.e., forced collision test—works in the following way: suppose a node u wants to verify its neighbors. Then, for every pair of identities in these neighbors, say s_1 and s_2, u asks s_1 to transmit a message m to s_2. Moreover, if s_2 indeed receives m, then it must repeat m to u. During the transmission of m, u can choose either to broadcast noise to cause a collision or to listen on the channel and checks if s_1 and s_2 follow the protocol. Now, if s_1 and s_2 are distinct identities, then nothing abnormal happens. However, if both s_1 and s_2 are controlled by an adversary that is equipped with one single radio transceiver, then the adversary will have to guess if u caused a collision or not. Unlike the previous two types of tests, a forced collision test can work in a single-channel scenario. Nevertheless, it suffers the disadvantage of not being able to tolerate the case where the adversary owns more

than one radio transceiver. Several algorithms from Klonowski et al. have taken this approach [25, 31].

Compared with other resource testing techniques, radio resource testing does not require access to additional information or detailed assumptions on computational capabilities. Nevertheless, existing anti-sybil protocols that use radio resource tests either: (a) rely on a central trusted base station; (b) combine radio resource testing with other resource constraints and/or outside information; or (c) require time complexity that grows with the number of sybil identities (as they test each potential identity separately), allowing the malicious devices to potentially swamp the system with never-ending tests. In this thesis, by contrast, we present both centralized (i.e., central base station exists) and distributed (i.e., no central base stations) algorithms that can reduce the number of sybil identities to an asymptotically optimal limit, using only radio resource tests, and with provable time complexity guarantees that are expressed with respect to the actual number of devices.

Nowadays, radio resource testing is considered to be practical as most real world wireless devices have access to many more channels than they can use simultaneously. In the future, however, the correctness of the assumption that the number of channels the adversary can simultaneously use is less than the total number of available channels—which is fundamental to the correctness and effectiveness of radio resource testing—is not so obvious. In particular, on the one hand, new technologies (e.g., software radios, which we will discuss more later) are appearing which allows devices to scan a large chunk of the frequency spectrum at the same time. On the other hand, as we have previously mentioned, various authorities are continuously releasing new frequency bands for public use. Therefore, the future fate of radio resource testing is still unclear and remain to be seen.

Combined approaches exist as well. Computational resource testing and radio resource testing each has unique advantages and disadvantages. In particular, radio resource testing is more *reliable* and *energy efficient* in detecting sybil identities, whereas computational resource testing is more *efficient* in terms of time. This means radio resource testing has a lower false negative ratio whereas computational resource testing can identify sybil identities quicker. Having observed this fact, Mónica et al. combine

these two techniques and propose a new protocol which is able to construct sybil-free quorums in wireless ad-hoc networks [40]. A shortcoming of their work, though, is that this paper makes several limiting assumptions (e.g., knowledge of network size and sender-side collision detection).

Radio Fingerprinting

In this approach, researchers explore methods to extract the "fingerprints" of radio transceivers, and use these fingerprints to counter sybil attacks. For example, in [11], the authors use the *received signal strength indicator (RSSI)* to identify different radio transceivers and thwart sybil attacks. In [45], the authors leverage the fact that in a mobile wireless network, identities that always move together are more likely to be sybil identities. Compare with other anti-sybil strategies, this scheme incurs less time overhead. However, it can potentially report false positive results.

2.2.3 Summary and Discussion

Many security related protocols in wireless networks and distributed systems rely on the assumption that there cannot be too many faulty clients in the system. However, sybil attacks can easily invalidate this assumption and hence finding proper countermeasures for it is of great importance. Unfortunately, though, sybil attacks are not easy to defend against. One reason is because the adversary can always abandon compromised sybil identities and generate new sybil identities to try to penetrate the target system again. In fact, researchers have long realized a sad truth: sybil attacks cannot be totally prevented [16]; if there are t malicious clients, there is no way we can stop them from creating t identities. As a result, the above mentioned approaches all focus on how to *thwart* sybil attacks, rather than how to *prevent* them.

One interesting point to note is that although we divide different approaches (for thwarting sybil attacks) into three major categories, they in fact all belong to "resource testing" from a certain perspective: the pre-distributed credential approach works because the adversary has limited certificates or pre-distributed keys; the social networking approach works because the adversary has limited human-established trust relationships. This observation naturally leads to an interesting question: are there any other generic type of resources we can utilize to counter sybil attacks?

Secondly, it is also interesting to explore the impact of *software radios* on radio resource testing. Software radios, which are different from traditional radio equipment, can send and receive signals on a wide range of frequency spectrum simultaneously. Hence, such devices may break the key assumptions which make radio resource testing useful at all. However, the capability of software radios strongly depends on the processing power of the CPU embedded in the device. These devices also tend to consume more energy then commodity radio hardware. How the adversary can utilize these new devices to her advantage and what are the corresponding countermeasure we can take are both interesting and promising future research directions.

2.3 Jamming

2.3.1 Overview

Jamming refers to the behavior of emitting radio signals to disrupt the normal communications of other wireless devices. It can make a certain frequency band—narrow or wide—unable to convey any message, hence it is one type of denial-of-service attack. Jamming can be classified into two major categories, namely, unintentional jamming (i.e., disruption) and intentional jamming (i.e., jamming attacks).

Unintentional jamming mainly comes from two sources: (a) selfish devices that use the frequency band without respecting other protocols; and (b) interference from other related or unrelated devices (e.g., neighboring nodes executing the same protocol, microwaves) [12]. This type of jamming has long been observed by researchers and many widely used protocols have incorporated countermeasures. (For example, the IEEE 802.11 wireless network uses CSMA/CA to mitigate such interference [1].) Among these countermeasures, the most popular technique is *backoff*. More specifically, in a backoff protocol, each participating node will transmit its message with a probability initialized to p where $0 < p < 1$. Upon transmission failure due to interference, this probability will be decreased geometrically, i.e., $p/2$, $p/4$, $p/8$, etc. This process continues until the message is successfully delivered or the minimum allowed probability is reached. Although backoff is a simple technique, it has been shown to be quite effective in both theoretical analysis and real world experiments.

Intentional jamming, or jamming attacks, on the contrary, can cause more severe

problems to the availability of wireless networks, and it is very hard to defend against them. In a recent paper by Gummadi et al. [27], the authors find "commodity 802.11 equipment is surprisingly vulnerable to certain patterns of weak or narrow-band interference". In particular, the authors are able to "disrupt a link with an interfering signal whose power is 1000 times weaker than the victim's 802.11 signals, or to shut down a multiple AP, multiple channel managed network at a location with a single radio interferer." Furthermore, simply changing the 802.11 operational parameters such as the clear channel assessment (CCA) threshold, transmission rate or packet size cannot effectively mitigate these malicious attacks. Similar findings (and conclusions) have been observed (and drawn) by other researchers as well [7].

Both the systems and the theory community have devoted significant efforts to thwarting jamming attacks. In particular, the systems community has developed many techniques to counter jamming from different perspectives, such as *jamming detection*, *jamming evasion*, and *jamming competition*. The theory community, on the other hand, tends to focus on the potential limitations of the adversary and then propose corresponding countermeasures. In particular, they usually assume a malicious user faces one or more of the following restrictions: (a) a limited energy budget; (b) a limited jamming rate; and/or (c) a limited number of channels the adversary can jam. We will discuss each of these approaches in more detail in the following subsection.

In jamming attacks, the adversary may have different capabilities and may utilize different strategies. Xu et al. [64] summarize four major types of jammers as: *constant jammers*, *deceptive jammers*, *random jammers* and *reactive jammers*. A constant jammer continuously emits a radio signal to stop legitimate nodes from using the channel. A deceptive jammer constantly broadcasts normal packets on the channel, deceiving the legitimate nodes into the receiving state. A random jammer is similar to a constant jammer, except that it randomly alternates between the jamming state and the sleeping state. Lastly, a reactive jammer remains inactive when the channel is idle and emits noise when it senses channel activity.

Constant jammers and deceptive jammers are more energy consuming than the other two types of jammers. Reactive jamming has the highest hardware requirement for the adversary, but it is also the hardest one to detect and defend against. For a more vivid illustration of these different types of jammers, please refer to Figure 2.1. We note here

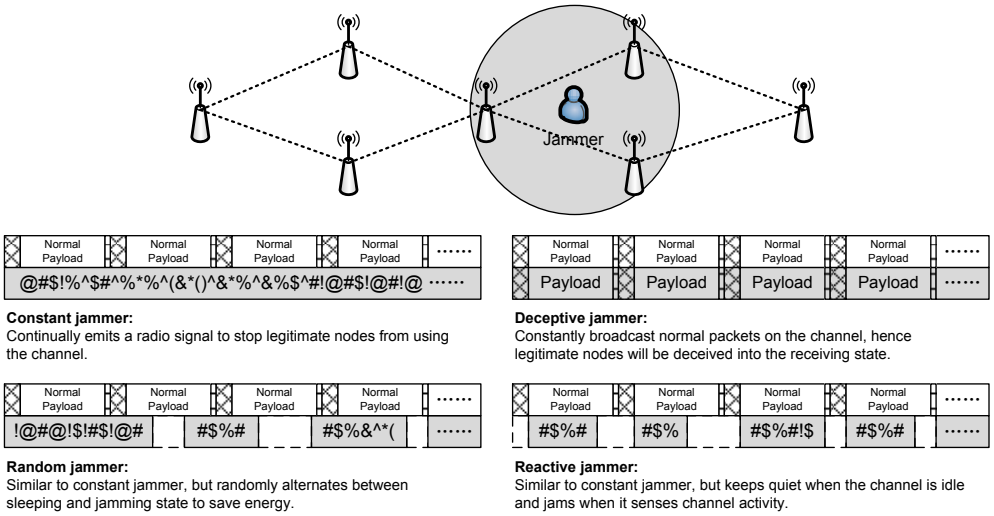

Figure 2.1: Different types of jammers [64].

that there also exist other classification mechanisms for jammers. For example, Bayraktaroglu et al. [7] propose categorizing jammers according to their capabilities of sensing and reacting to the wireless medium state (i.e., *channel-oblivious* vs *channel-aware*), and whether they maintain states that dictate their future actions (i.e., *memoryless* vs *stateful*).

2.3.2 Countermeasure

Physical Layer and MAC Layer Approaches

In the systems domain, both physical layer and medium access control (MAC) layer mechanisms have been proposed to counter jamming attacks.

At the physical layer, the two most popular techniques for evading jamming are *spread spectrum radio encoding* [52] and *channel hopping* [41,57]. In spread spectrum radio encoding, a radio signal is intentionally spread in the frequency domain, leading to a signal with wider bandwidth. This technique works because widely spread signals make the detection of the start of a packet harder, leaving the adversary insufficient time to jam it. Unfortunately though, the widely deployed 802.11 networks only use a relatively narrow spreading [1]. Frequency hopping, on the other hand, keeps wireless devices hopping among different channels according to some pre-defined sequences. As we have previously discussed, frequency hopping can either be *uncoordinated*, in which the senders and the receivers do not share common sequences; or *coordinated*, in which the hopping sequence is known to both the senders and the receivers. Coordinated fre-

quency hopping is very effective in multi-channel setting. (For example, the Bluetooth standard, which uses coordinated frequency hopping, is less susceptible to jamming than the IEEE 802.11 WiFi networks.) Nevertheless, a big challenge in applying this scheme is to establish the shared pseudorandom sequence in a reliable and efficient manner, and to keep it secret from the attackers. Uncoordinated frequency hopping, on the other hand, does not require common knowledge among the communicating parties, but needs to be carefully designed to ensure the effectiveness and efficiency.

There also exist techniques for detecting jamming attacks at the physical layer. In [66], Xu et al. survey some such possibilities, including signal strength (SS), carrier sensing time, packet delivery ratio (PDR), and multimodal approaches. For signal strength, two natural approaches are: (a) comparing average signal strength with average ambient noise level; and (b) classifying the shape or pattern of received signal samples. Unfortunately, experimental results indicate that under some scenarios, the shape of the RSSI time series for normal traffic is very similar to the one when reactive jammers are present. Hence, both approaches cannot reliably distinguish jamming from normal traffic. Carrier sensing time is another possibility, as channels might appear constantly unavailable to a device when it is jammed. The authors explore this possibility and conclude that it is only suitable when two conditions are met: (a) the adversary is neither a random jammer nor a reactive jammer; and (b) the underlying MAC protocol uses a fixed threshold to determine the channel status. Packet delivery ratio is yet another possible detection technique, as jamming can result in low PDR by disrupting transmissions. Through experiments, the authors find a threshold on PDR readings can reliably distinguish jamming from normal network congestion, regardless of the jammer's type. However, the authors also find this mechanism can lead to false positives under some network dynamics, such as device power failure or device moving out of communication range, as these dynamics will bring sudden PDR drop similar to the jamming case. Lastly, the authors combine packet deliver ratio and signal strength and propose a multimodal jamming detection technique. Experimental results show this approach is effective under various scenarios.

At the MAC layer, researchers have proposed *channel surfing* and *spatial retreat* to evade jamming attacks [34, 63, 65, 67].

Channel surfing [67] is motivated by the physical layer frequency hopping tech-

nique. However, unlike frequency hopping which operates at the physical layer and involves continuous changing of frequency, channel surfing operates at the MAC layer and only changes frequency on demand (e.g., when jamming or collision is detected). Channel surfing is relatively simple in single-hop networks, but implementing it in multi-hop networks is quite challenging.

To address this issue, researchers have proposed the *coordinated channel switching* scheme [65]. In such a scheme, a node will voluntarily move to a new channel when it is jammed. Nodes neighboring these jammed nodes, which are called boundary nodes, will detect this absence and ask the entire network to switch to the new channel. This scheme may suffer from large latencies as the entire network will have to switch channels even if only one single node is jammed. One way to alleviate this is to let the boundary nodes act as relay nodes among different channels.

Spatial retreat, on the other hand, allows jammed nodes with mobility to evacuate from the jammed region [67]. Nevertheless, simply escaping from the jammed region is not enough, as a mobile jammer can move around and cause many wireless clients to relocate. By doing so, the adversary can change the network's topology or partition the network. Therefore, spatial retreat strategies need to contain two phases to ensure robustness against mobile jammers: (a) the escape phase in which jammed nodes "retreat" from jammed regions while keeping connected to the network; and (b) the reconstruction phase in which all nodes move around to reconstruct a proper network topology. For example, in [34], the authors apply the techniques of "virtual forces" and "potential fields", which are widely used in robotic systems, to maintain the network topology in the presence of a mobile jammer.

Lastly, aside from detection and evasion, it is also possible to compete with the jammers [64]. Nodes being jammed can increase signal strength to overpower the jamming noise, but with the price of higher energy consumption and higher probability of collision (and unintentional interference). Victims of jamming can also exchange lower throughput for higher delivery ratio; e.g., by using stronger error correction code (ECC). In [43], Noubir and Lin investigate and analyze the impact of various coding schemes on the performance and robustness of wireless networks. However, many open questions still exist in this direction. For example, can we develop a protocol that can adaptively adjust the ECC rate based on the detected jamming level? This requires the underlaying

jamming detection mechanism to possess the ability of differentiating various jamming threats according to their severity.

Theoretic Analysis

The theory community has devoted much effort to countering jamming attacks as well. As stated earlier, researchers tend to focus on potential limitations the adversary may face and then use these limitations to develop corresponding countermeasures. For each such limitation, we will discuss some proposals. Notice, most works surveyed here cannot be extended to wired networks. However, exceptions do exist, and we will discuss these exceptions when relevant papers are presented.

Limitation I: limited jamming rate. In this category, the researchers usually assume the adversary can jam only a limited fraction of the time or a limited fraction of messages for every time period of some length. The major challenge is to efficiently detect and use the non-jammed periods to achieve communication.

For example, in a series of papers, Awerbuch et al. [5] and Richa et al. [46, 47] study how to thwart adaptive jammers and reactive jammers in single-channel wireless networks by limiting the jamming rate.[4] They first define the notion of $(T, 1-\varepsilon)$-*bounded adversary*: for any time window $\omega \geq T$, the adversary can jam at most $(1-\varepsilon)\omega$ time steps in that window. The goal is to develop MAC protocols that are $O(1)$-*competitive* against a jammer that is $(T, 1-\varepsilon)$-bounded for any $0 < \varepsilon < 1$. Here, a protocol is called c-competitive if it can utilize at least a c-fraction of the non-jammed time steps for communication.

The core strategy used by the authors is as follows: assume each node v sends a message in each time step with probability p_v that is initially a small constant. This implies that node v will listen in each time step with probability $1-p_v$. Now, if v indeed listens and finds the channel is idle, then it sets $p_v = (1+\delta)p_v$. Otherwise, if v successfully receives a message, then it sets $p_v = (1-\delta)p_v$. The intuition behind this strategy is that an idle state of the channel indicates more capacity is available; successful mes-

[4] Adaptive jammers and reactive jammers are very powerful jammers. They are very hard to defend against, especially in a single-channel scenario. An adaptive jammer is similar to a reactive jammer in the sense that they both are allowed to know the protocol and the protocol's past execution history. What differentiates them is that a reactive jammer can immediately jam a message when it senses channel activity while an adaptive jammer does not posses such capability (i.e., it has to decide its action at the beginning of each time step, in the synchronous setting).

sage delivery, on the other hand, indicates channel bandwidth is about to be exhausted. The elegance of this strategy lies in ignoring all the cases where interference occur, as such interference may be due to either normal collisions or adversarial jamming.

The problem then comes down to the value of δ. Ideally, the sum of p_v of all nodes should be 1 so that in each non-jammed time step one message can be successfully delivered with constant probability. This in turn leads to another question: how does each node know their current p_v is "correct" in a sense that the sum of p_v will be 1?

It turns out a simple measurement exists: for each node, if the number of idle-channel time steps observed is approximately the same with the number of successful-delivery time steps observed, then $\sum p_v \approx 1$.

Based on the above observation, the authors propose three protocols. The first two protocols [5, 46] are constant competitive against an adaptive jammer in single-hop and multi-hop wireless networks, respectively. The last protocol [47] is $e^{-\Theta(1/\varepsilon^2)}$-competitive and can tolerate a stronger reactive jammer. The authors also use simulations to test their protocols. In particular, when compared with the standard 802.11a MAC protocol, the new protocols show big performance improvement even for small ε.

Limitation II: limited channels can be jammed. Traditionally, radio transceivers can only operate on a single channel at any given time. If there are multiple channels available while the number of radio transceivers owned by the adversary is limited, then it is possible for honest nodes to choose channels that are not jammed by the adversary to exchange information. This is the key observation that motivates the limited-jammed-channels approach. The key challenge of this scheme is to design algorithms that ensure information can be exchanged efficiently.

In a paper by Meier et al. [36], the authors study how can the existence of multiple channels help resist jamming in solving the *neighbor discovery* problem. The authors consider a single-hop, synchronized wireless network that consists of c channels. In the network, there are two honest devices and one reactive jammer that can jam up to t channels. The paper analyzes the *competitiveness* of various discovery algorithms. Let REF be an optimal algorithm that has complete knowledge of t; let T^t_{REF} be the expected discovery time under REF. Further assume T^t_{ALG} is the expected discovery time for any given algorithm ALG. Then the *competitive ratio* is defined as

$\max_{0\leq t\leq c-1}\{T^t_{ALG}/T^t_{REF}\}$. A smaller competitive ratio corresponds to a better discovery algorithm (in terms of time efficiency).

The authors first study the behavior of the optimal algorithm REF. For $t = 0$, each node should choose a pre-defined channel and the expected discovery time is 2. For $1 \leq t \leq c/2$, each node should choose a channel uniformly at random from the first $2t$ channels and the expected discovery time is $8t$. Lastly, for all other cases, the optimal strategy is to choose each channel with probability $1/c$, and the expected discovery time is $2c^2/(c-t)$. The authors then propose a class of algorithms to handle the case in which t is unknown. In essence, the idea is to guess the value of t and apply the optimal algorithm REF for the guessed value. The authors prove the proposed algorithm has a competitive ratio of $O(\log^2 t)$, which is close to the optimal.

Gossiping is another classical problem in distributed computing and wireless networks. It refers to the situation where each process in the network attempts to disseminate an initial value (or "*rumor*") to all other processes. The question in concern is how long this task will take. In a series of three papers [14, 15, 20], the authors try to address the gossiping problem in jamming-prone multi-channel wireless networks.

To begin with, the authors first define the more general (ε, δ)-*gossip problem* which requires at least $(1-\delta)n$ processes to receive a common set of at least $(1-\varepsilon)n$ initial values, where n is the total number of processes.

In the first paper [14], the authors focus on achieving $(\varepsilon, t/n)$-gossip—or simply ε-gossip—in single-hop, synchronous wireless networks which contain c channels. Here, t is the maximum number of channels the adversary may jam. The authors consider *deterministic oblivious* algorithms, which fix the behavior of the processes in each round in advance. The contribution of the paper mainly consists of two parts: (a) the authors prove the minimum number of rounds required to solve the ε-gossip problem using a deterministic oblivious algorithm is $\Omega((1-\varepsilon)n/(\varepsilon c^2))$ for the case of $t = 1$; (b) the authors propose an algorithm that solves the problem and matches the lower bound.

In the second paper [20], the authors try to solve the $(t/n, t/n)$-gossip problem by using *deterministic adaptive* algorithms. The core tool they use is a combinatorial construction called a *selector* [32]. The authors propose a more general concept called a multi-selector. By using these combinatorial tools, the authors develop algorithms that can solve the problem in polynomial time when $c = \Omega(t^2)$. Nevertheless, the time

consumption grows to exponential when $c = t + 1$, and the authors prove that such high cost is inevitable in this case.

In [15], the authors close their study on gossiping by considering randomization. In particular, they develop an algorithm that can solve the $(t/n, t/n)$-gossip problem in $\Theta(n^2 t^2 \log n)$ rounds even when $c = t + 1$. This is a significant improvement over the exponential bounds in [14, 20] for the same scenario.

Limitation III: limited energy budget. Sending/receiving messages and jamming both consume energy. If the adversary has only a limited energy budget, then the number of messages or the length of time period she can jam is constrained. Approaches in this category usually assume an upper bound on the amount of energy the adversary may have and focus on how to evade the jammer or how to quickly depletes the adversary's energy. When compared with the previous two categories, this approach gives the maximum flexibility to the adversary, and may be the closest to the real world situation.

For example, in a paper by Gilbert et al. [21], the authors study the maximum possible *efficiency* of the jammers, especially when they are restricted by an energy budget. They consider a single-hop, single-channel, synchronous wireless network that has two honest players named Alice and Bob, and a jamming adversary named Collin that can jam up to β messages. The authors first show that Collin can delay the communication between Alice and Bob for at least $2\beta + (\log |M|)/2$ rounds, where M is the set of messages Alice and Bob may exchange. The authors then look into the case where Alice has a budget of $\beta + \Delta$ and tries to deliver a message to Bob, who does not broadcast. They show such protocols have a lower bound of $2\beta + \max\{\frac{2\Delta}{e}|M|^{1/(2\Delta)} - 2\Delta, \frac{\log |M|}{2}\}$ on message complexity. This bound, as put by the authors, "highlights an inherent tradeoff between Alice's message complexity and her throughput." The authors also propose a protocol which matches this bound.

Towards the end of the paper, the results are generalized to several classical n-player games via reduction to previous findings. The authors have also considered situations in which crash failures are present.

More recently, in a series of papers by King et al. [30] and Gilbert et al. [22–24], the goal is to achieve high a *resource competitive ratio* in jamming resistance. More specifically, the authors focus on *relative cost*, which is the energy consumption incurred by

Table 2.2: Comparison of systems approaches to counter jamming attacks.

Countermeasure	Type	Stack Layer
Spread Spectrum	jamming evasion	physical layer
Channel Hopping	jamming evasion	physical layer
Signal Strength (SS)	jamming detection	physical layer
Carrier Sensing Time	jamming detection	physical layer
Packet Delivery Ratio (PDR)	jamming detection	physical layer
Multimodal (SS and PDR)	jamming detection	physical layer
Channel Surfing	jamming evasion	MAC layer
Spatial Retreat	jamming evasion	MAC layer
Stronger Signal Strength	jamming competition	N/A
Error Correction Code (ECC)	jamming competition	N/A

the honest nodes divided by the energy consumption incurred by the adversary. The goal is to design protocols that can enforce a low relative cost (i.e., force the adversary to spend a lot more energy than the honest nodes) if the adversary wants to effectively block the communication.

For example, in [30], the authors look at the case where an honest sender Alice wants to send a message m to an honest receiver Bob in a single-channel, single-hop, synchronous network. Sending/receiving messages, jamming, and forging collisions each costs a certain amount of energy. The authors propose a protocol that can always transmit the message with an energy consumption of $O(T^{0.62} + 1)$, where T is the total budget the adversary has. At the end of the paper, the authors discuss how their protocol can be used to counter application level denial-of-service attacks. This is the only paper which explicitly states the relevant results can be extended to scenarios other than wireless networks.

Later in [24], the authors extend their results to the 1-to-n broadcast scenario. In their most recent paper [22], the authors complement their upper bounds with tight or nearly tight lower bounds. In particular, they prove that any 1-to-1 communication algorithm with constant probability of success has expected cost $\Omega(\sqrt{T})$. As for 1-to-n broadcast, some node has to incur cost at least $\Omega(\sqrt{T/n})$.

For more detailed discussion on resource competitive analysis, please refer to the position paper by Gilbert et al. [23].

2.3.3 Summary and Discussion

[5]In fact, in Chapter 4, we have proposed an augmentation to the SIMPLESYBILSIEVE algorithms so

Table 2.3: Comparison of theoretic works on thwarting jamming attacks.

	Jammer's Limitation	Topic	Network Model	Adversary Model	Collision Detection
Awerbuch et al. [5]	jamming rate	constant competitiveness	SH, SC	adaptive	yes
Richa et al. [46]	jamming rate	constant competitiveness	MH, SC	adaptive	yes
Richa et al. [47]	jamming rate	constant competitiveness	SH, SC	reactive	yes
Meier et al. [36]	channels to jam	node discovery	SH, MC	reactive	no
Dolev et al. [14]	channels to jam	gossiping	SH, MC	reactive	no
Gilbert et al. [20]	channels to jam	gossiping	SH, MC	reactive	no
Dolev et al. [15]	channels to jam	gossiping, authenticity	SH, MC	reactive, spoofing	no
Gilbert et al. [21]	energy budget	jammer's efficiency	SH, SC	reactive, spoofing	yes
King et al. [30]	energy budget	resource competitiveness	SH, SC	reactive	yes
Gilbert et al. [24]	energy budget	resource competitiveness	SH, SC	reactive	yes
Gilbert et al. [22]	energy budget	resource competitiveness	SH, SC	adaptive	yes
SYBILCAST	channels to jam	sybil resistance	SH, MC	adaptive	no
SIMPLESYBILSIEVE	channels to jam	sybil resistance	SH, MC	adaptive	yes[5]
SYBILSIEVEOPT	channels to jam	sybil resistance	SH, MC	adaptive	yes

Legend: SH, MH, SC, MC represents single-hop, multi-hop, single-channel, multi-channel, respectively.
Note: an adversary is a reactive jammer implies it is also an adaptive jammer.

Table 2.2 and Table 2.3 summarize some key aspects of the research we have surveyed in this subsection. Notice, for the sake of completeness, we have also included our SYBILCAST and SIMPLESYBILSIEVE/SYBILSIEVEOPT algorithms in this table. Through this examination, some interesting points which are worth discussion can be identified.

To begin with, all theoretic models assume a synchronized notion of time. Particularly, they all assume fixed-size time slots and assume all devices agree on the beginning and ending of each slot. In the real world, however, it is unclear whether this property holds. Some authors already realize this problem, but argue that in some cases the asynchronous model can be "converted" into a synchronous one with only a constant factor increase in time complexity, as shown by Roberts [49]. Some other authors, e.g., Gilbert et al. [21], explicitly analyze the impact of asynchrony in their papers. We also

that it no longer requires collision detection.

note here that more research exists on the topic of synchronization, even in the settings where adversarial behavior is present. (See, e.g., [13] by Dolev et al.)

Second, the channel model seems to have a big impact on the feasibility and efficiency of solving certain tasks. For example, the multi-channel model provides many possible approaches for thwarting jamming attacks, such as frequency hopping, selectors. In particular, frequency hopping has been shown both theoretically and experimentally to be very effective in jamming evasion. The multi-channel model also relaxes other restrictions on the adversarial model. For example, research that assumes a multi-channel model can usually tolerate reactive jammers, which have the strongest jamming capability; in single-channel scenarios, by contrast, it usually takes a lot more effort to design and analyze protocols that can provide the same (or lesser) guarantees, as has been shown in the series of papers from Awerbuch et al. [5] and Richa et al. [46, 47].

Last but not least, there exists a missing step in many of the works we have surveyed, namely, extension to multi-hop wireless networks. Wireless technology is fast moving, and soon large scale wireless networks will be deployed covering multiple hops. In [34, 67], the authors show that a multi-hop topology can bring new and challenging problems that do not exist in the single-hop scenario. The paper from Richa et al. [46] further demonstrates that even for the same protocol, the corresponding analysis in single-hop and multi-hop scenarios can be quite different.

2.4 Spoofing and Authentication

As we have previously stated, in the context of network security, *spoofing* refers to the situation in which an entity successfully masquerades as another entity and communicates with others on this stolen identity's behalf. It is an attack on *authenticity*, but can target confidentiality, availability, privacy, etc. For example, an adversary can utilize the stolen identities to access materials that she is not allowed or not supposed to view.

We (once again) emphasize that spoofing attacks share some similarities with sybil attacks as they both are related to masquerading identities. However, fundamental differences exist between them. In particular, spoofing attacks focus on camouflaging existing entities, whereas sybil attacks focus on fabricating new identities that do not correspond to any real existing entities.

To prevent spoofing, *mutual authentication* must be carried out among the communicating parties. According to RFC4949 [51], authentication is "the process of verifying a claim that a system entity or system resource has a certain attribute value." An authentication process consists of two basic steps: (a) *identification*, "in which the claimed attribute value is presented to the authentication subsystem"; and (b) *verification*, "in which the authentication information that acts as evidence to prove the binding between the attribute and that for which it is claimed is presented or generated."

Richard Smith [53] concludes five elements of an authentication system are: (a) the entity to be authenticated (e.g., a user, a web server); (b) the authenticator (e.g., distinguishing characteristic, token); (c) the proprietor (e.g., system owner, administrator); (d) the authentication mechanism (e.g., Kerberos [56], X.509 [8]); and (e) the access control mechanism (e.g., discretionary access control, mandatory access control, role-based access control [19]).

There are numerous authentication protocols that can be used to counter spoofing. However, generally speaking, only four major means exist to authenticate an entity's identity [55]: (a) something the individual *knows* (e.g., passwords, personal identification number); (b) something the individual *possesses* (e.g., smart cards, physical keys); (c) static biometrics, i.e., something the individual *is* (e.g., fingerprint, retina); and (d) dynamic biometrics, i.e., something the individual *does* (e.g., voice pattern, typing rhythm).

Finally, we note that authentication cannot be used to counter sybil attacks. In particular, authentication cannot stop an adversary from creating sybil identities that do not correspond to any real existing entities.

Chapter 3

Thwarting Sybil Attacks in Centralized Wireless Networks

3.1 Introduction

Imagine a scenario in which 100 scientists sit in an auditorium, listening to a lecture. Most have a laptop opened in front of them, and many are attempting to download data from the Internet, perhaps a video of the preceding day's lecture[1]. Luckily, the wireless base station uses a *fair* version of 802.11, ensuring that each registered user gets an equal share of the bandwidth[2]. Unfortunately, one clever attendee is upset with the slow download speed and finds a way to circumvent this fairness mechanism by simulating 200 distinct users. By doing so, she gets $2/3$ of the total available bandwidth. For the remaining users, their downloads now take three times as long as before. As we have previously discussed, such behavior is typically referred to as a sybil attack.

In this chapter, we present a new protocol, SYBILCAST, that thwarts this type of malicious behavior in multi-channel wireless networks. More specifically, our protocol ensures that each honest participant will receive at least a constant fraction of his or her fair share of the bandwidth[3]. This implies that each download of data will complete in asymptotically optimal time, even in the presence of a sybil attack. In addition, our

[1]These are very diligent scientists, watching video lectures, rather than surfing Facebook, while ignoring the current speaker.

[2]Perhaps this is a computer science conference on experimental networking, as most existing 802.11 deployments are not fair; however there is much research related to fairness.

[3]Note that in this chapter we focus on fairness in *downloading* data, assuming that data flows from the base station to the clients tend to dominate the bandwidth usage. Similar techniques would also yield protocols for uploading data.

protocol is robust to message spoofing and wireless jamming as well. We also note here that SYBILCAST is a randomized protocol, and all its guarantees hold *with high probability* (w.h.p.); i.e., with probability at least $1 - 1/N^k$ for any constant $k \geq 1$, where N is the maximum number of nodes that are executing SYBILCAST.

Approach and Challenges. The core strategy we will use to counter sybil attacks is radio resource testing. More specifically, we use a simultaneous receiver test. Nevertheless, three key challenges exist when generalizing the basic scheme (which was described in Chapter 1) to our setting: limiting the overhead, integrating newly arrived users, and coping with message spoofing from malicious users.

First, the process of testing must be implemented efficiently with limited overhead: there is little benefit to eliminating the sybil identities if the protocol for detecting these identities consumes all the available bandwidth. In general, the protocol must spend most of its time delivering data packets, not running tests. For example, it does not suffice to test one pair of identities at a time: if there are d identities (many of which may be sybil identities), it will take $\Theta(d^2)$ time to test them all. Instead, the running time should depend on the number of real users (not the number of sybil identities). In addition, the malicious users may be colluding, and may be able to monitor multiple channels simultaneously; it will be insufficient to test identities on a per pair basis and assign each of these two identities to different channels.

The second challenge is coping with the continuous nature of the system. Clients continue to arrive over time, requesting data downloads. Malicious users continue to register new sybil identities. Thus it is insufficient to simply run a set of tests and then proceed with a standard data delivery protocol. A key aspect of our design is balancing the rate at which new users (and potentially new sybil identities) are admitted to the system and the rate at which sybil identities can be detected (and eliminated). For example, by reducing the admission rate, we can achieve a tighter bound on the number of sybils; however, in doing so we risk reducing the throughput of the honest users.

A third challenge is coping with adversarial spoofing which, much like adversarial jamming, can result in a denial-of-service attack. For example, the malicious users can attempt to spoof messages from the honest clients to the base station, tricking the base station into thinking that the honest clients are dishonest. By careful use of un-

coordinated frequency hopping, along with simple techniques like hash chains, we can overcome this attack.

Results. The SYBILCAST protocol provides the following guarantees. In a system with at most N real users, among which there are at most t dishonest users, and $c > 20t$ channels: if an honest user p requests a download m of size $|m| \geq c^2 \log^3 N$, and if there are at most $\texttt{contention}(p, m)$ concurrently active real users at any point during the request, then with high probability the download will complete in $O(|m| \cdot \texttt{contention}(p, m))$ time. This implies that the honest user has received an asymptotically fair share of the bandwidth. At the same time, we guarantee that there are at most $O(\texttt{contention}(p, m))$ sybil identities, i.e., $O(1)$ sybil identities for every real identity, on average. We also note here that for smaller requests (i.e., those with size smaller than $c^2 \log^3 N$), with high probability SYBILCAST still guarantees all their data will be delivered, but the time consumption may not be asymptotically optimal.

Roadmap. In the reminder of the chapter, Section 3.2 formally models the network environment and the problem we want to solve. Section 3.3 describes the SYBILCAST protocol. Section 3.4 gives the analysis of the protocol. We conclude this chapter with summary and discussion.

3.2 Model and Problem Statement

We consider a synchronous wireless network consisting of one base station and a dynamically changing set of clients. There are at most N clients, an unknown subset of which are active at any given time. We call these clients *wireless nodes*, or just nodes when there is no confusion. Each node may either be an honest node or a malicious node. Among these N nodes, up to t are malicious.

Notice that N is an upper bound on the system size, while the actual system size is unknown in advance. In fact, the number of active nodes may change throughout an execution. We assume that a polynomial-factor estimation on N (i.e., a constant-factor approximation of $\log N$) is known.

Nodes communicate with the base station via a multi-channel radio transceiver consisting of c independent channels. We assume $t < c/20$ and $N > c$. Fundamentally,

we are assuming a reasonably large number of available channels, or a reasonably small number of colluding attackers. (Bluetooth [2], for example, specifies 79 channels, and there is every reason to believe that even more channels are technically feasible.)

Every node (including the malicious nodes) and the base station have only one radio transceiver. Time is divided into *rounds*, and each node and the base station can access only one channel in each round. The channels are collision-prone: if two or more nodes broadcast on the same channel concurrently, there is a collision and all information is lost. We do not assume collision detection. In particular, nodes and the base station can not distinguish between collision and no transmission.

A malicious node may: (a) jam channels (by broadcasting noise); (b) spoof messages (by claiming to be someone she is not); and (c) create sybil identities. Messages from the base station can be authenticated (by using signatures), but messages from wireless nodes are unauthenticated: a malicious node can claim to be any other active node or a new identity.

We assume that devices have access to standard cryptographic tools. The base station and the wireless nodes use a *pseudorandom number generator* (see, e.g., [33]) to generate sequences of "random" bits from small messages. (A pseudorandom number generator is a function which takes an input as a *key* and then generates a long sequence of pseudorandom bits. Moreover, obtaining the first i bits will not provide any information on the $(i+1)$st bit.) We also rely on hash functions, which we assume to be *pre-image resistant* (i.e., given an output of the hash function, the adversary cannot find an input which hashes to the given output). As can be seen, we assume "perfect" versions of these tools, as they are not the focus of this thesis.[4]

Problem Statement. Nodes may enter the system at any time. When an honest node enters the system, it initiates a *request*, attempting to download data from the base station. No honest node will withdraw its request halfway. An honest node (and its request) is considered to be *active* from the point at which it enters the system. A request is considered complete (and the node becomes inactive) when the node has received all the requested data.

[4]In fact, as long as the security properties of these cryptographic tools are sufficiently *hard* to break, our algorithm can still enforce the same guarantees. For example, we are only using the hash function for a polynomial (with respect to N) number of times. Therefore, as long as the hash function is pre-image resistant w.h.p., the correctness of SYBILCAST still holds.

The problem we address is developing an efficient and robust protocol for the base station to deliver requested data to active honest nodes in a fair manner. We measure the performance in terms of the time it takes for a node to complete a request. For any requested data m from a particular honest node p, we define $\texttt{contention}(p, m)$ to be the maximum number of concurrently active nodes at any point during the request. (Note that this includes both honest nodes and malicious nodes, but not sybil identities.) In a perfectly fair system, it will take at least $|m| \cdot \texttt{contention}(p, m)$ time for p to fetch m, in the worst case, if each node is given a fair share of the total bandwidth.

3.3 Protocol

We now describe the SYBILCAST protocol, which consists of three parts: (a) registration, where new nodes (and potentially sybil identities) enter the system and request a download; (b) data delivery, where the base station delivers data to the registered identities; and (c) verification, where the base station attempts to spot and eliminate sybil identities. The base station repeatedly executes these three phases, one iteration of which we call an *epoch*.

Registration. The goal of the registration phase is to deliver a unique and random binary string to each new request (and the node which sent it). We call this string a *final seed*. Each registered identity (which can represent an honest node or a sybil identity) can use its final seed as the key for a pseudorandom number generator. The output of this pseudorandom number generator is a channel hopping sequence that the registered identity follows during the ensuing data and verification phases. The length of the sequence is assumed to be sufficiently long so that for every request the corresponding node needs to register only once.

Notice, each final seed should be known only by the base station and the intended receiver. Otherwise, if it is known to the malicious nodes, then they can later leverage the knowledge of the channel hopping sequence to efficiently jam honest nodes.

Data. The data phase is used to transmit data packets and deliver authentication messages from the base station to the registered identities. In each round of a data phase, the base station randomly chooses a registered identity, and sends it a packet on the channel

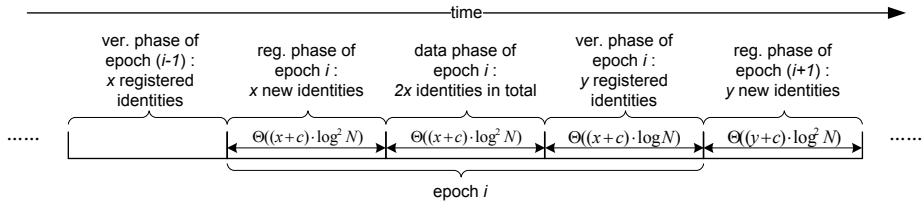

Figure 3.1: Dynamic phase length of SYBILCAST. Note, there are x registered identities when epoch i begins (i.e., x identities remain in the system after the verification phase of epoch $i-1$).

specified by its final seed. The base station also includes a one-time random string, called a *nonce*, in each packet. The base station will later use these nonces to verify whether the identities who should have been listening on that channel really were.

Verification. The verification phase allows the base station to detect and eliminate sybil identities. During the verification phase, each registered identity should transmit a verification message to the base station containing an encrypted list (using the final seed as the key) of the nonces received by the identity during the preceding data phase. (The purpose of encryption is to prevent the malicious nodes from eavesdropping and learning the nonces.) The base station will receive and verify these messages and remove identities that fail to send verification messages or provide an insufficient number of nonces[5].

The length of the three phases in an epoch is dynamically decided at the beginning of that epoch. Assume there are x registered identities at the beginning of an epoch. Then each of the three phases has length $O\left((x+c)\log^2 N\right)$. This provides sufficient time for x new identities to be registered, while preventing the number of sybil identities from growing too fast. (The phase lengths include an $(x+c)$ term—instead of just x—so that x new identities can be registered in one epoch even when $x < c$.) As we shall later see, this key design decision well balances the (new identities') admission rate and the (sybil identities') elimination rate. This schedule is illustrated in Figure 3.1.

In the remainder of this chapter, we will use $id(i)$ to denote the total number of registered identities at the beginning of epoch i, $id'(i)$ to denote the total number of registered identities at the beginning of the data phase of epoch i, and $s(i)$ to denote the total number of sybil identities at the beginning of the data phase of epoch i. We now describe each phase in more detail.

[5]We will give the precise meaning for "sufficient" and "insufficient" in Section 3.4.

3.3.1 Registration Phase

For an epoch i, the corresponding registration phase is subdivided into $\Theta((id(i) + c)\log N)$ sub-phases each of length $\Theta(\log N)$. In each sub-phase, nodes compete to send their requests to the base station. The base station will deliver a *partial seed* to one winning node. A partial seed is a unique and random binary string, and if a node has collected $\Theta(\log N)$ partial seeds within one registration phase, then its registration is complete: it generates the final seed via a bit-wise `xor` of the partial seeds. (Otherwise, if a node collects fewer than $\Theta(\log N)$ partial seeds in one registration phase, then it is not registered, and has to start over again, collecting another $\Theta(\log N)$ partial seeds during the next registration phase.) Each honest node will have at most one request and hence at most one identity at any time. However, malicious nodes may attempt to register multiple sybil identities to fulfill multiple requests simultaneously. The registration protocol is summarized in Figure 3.2. We now discuss it in more detail.

As per our previous description, the length of each registration phase is determined by the number of registered identities that remain in the system at the beginning of the epoch. Therefore, during the registration phase, there can be significant contention among the nodes that are trying to register. For example, if a large number of new nodes try to join the system during one registration phase, then each may only get a few partial seeds, with none completing the whole process. To avoid this, in the first round of each registration sub-phase in epoch i, the base station will send a *registration list* which includes requests that have already obtained at least one partial seed in the current registration phase. When the registration list is full, i.e., is of size $id(i)$, an honest node will only compete in this sub-phase if its request is already in the list.

Furthermore, to simplify the analysis, we divide each registration phase into two parts of equal length. During the first half, each unregistered honest node is eligible to compete for its first partial seed. Once it receives its first partial seed (and hence gets into the registration list), it remains silent until the halfway point. During the second half, all honest nodes that have not been added to the registration list remain silent, while all honest nodes that are already in the registration list become eligible to compete for more partial seeds (so as to finish their registration process).

The base station will do the following in each sub-phase: (a) In the first round,

Base Station Registration Phase for an Epoch:

Let ep denote epoch number.
$dict \leftarrow nil$ ▷ A dictionary holding requests that have received at least one partial seed
$list\_id \leftarrow \emptyset$ ▷ A list holding the identities of the requests that have received at least one partial seed
if $(ep = 1)$ **then** $dict\_reg \leftarrow nil$ ▷ A dictionary holding requests that have already successfully registered
$q \leftarrow \Theta((sys\_size + c) \log N)$ ▷ Number of sub-phases; sys_size is the number of registered identities
for $(sp \leftarrow 1$ to $q)$ **do**
    $sent \leftarrow false, ch \leftarrow \texttt{random}(1, c)$ ▷ $\texttt{random}(x, y)$ returns a random value between x and y
    $key \leftarrow \texttt{randStr}()$ ▷ $\texttt{randStr}()$ returns a random binary string from a sufficiently large space
    $seq \leftarrow \texttt{prng}(key)$ ▷ $\texttt{prng}(k)$ returns a pseudo random sequence with k as the key
    $\texttt{broadcast}(\langle REG\_START, seq, list\_id, ep, q, sp, sys\_size\rangle, ch)$ ▷ Broadcasts message on channel ch
    for $(r \leftarrow 1$ to $a_1 \log N)$ **do** ▷ a_1 is some sufficiently large constant
        $msg \leftarrow \texttt{listen}(seq[r])$ ▷ Listen on channel specified by the sequence
        if $(msg.type = REG\_REQ)$ **then**
            if $(sent = true)$ **then continue**
            if $(sp < q/2$ **and** $|dict| < sys\_size)$ **then**
                $req \leftarrow \texttt{genReq}(msg.id)$ ▷ $\texttt{genReq}(x)$ returns a new request structure with x as the identity
                $pseed \leftarrow \texttt{randStr}()$ ▷ $\texttt{randStr}()$ returns a random string; $pseed$ is the partial seed
                $\texttt{broadcast}(\langle REG\_RESP, pseed, msg.id, sp\rangle, seq[r])$
                $sent \leftarrow true, req.last\_hash = msg.hash$
                $dict[msg.id] = req, list\_id \leftarrow list\_id \cup \{msg.id\}$
            else if $(sp \geq q/2$ **and** $msg.id \in dict)$ **then**
                $req \leftarrow dict[msg.id]$
                if $(\texttt{hash}^{(sp-msg.lsp)}(msg.hash) = req.last\_hash)$ **then**
                    $pseed \leftarrow \texttt{randStr}()$
                    $\texttt{broadcast}(\langle REG\_RESP, pseed, msg.id, sp\rangle, seq[r])$
                    $sent \leftarrow true, req.last\_hash = msg.hash$
                    $req.pseed\_list \leftarrow req.pseed\_list \cup \{msg.last\_pseed\}$
                    if $(|req.pseed\_list| \geq a_2 \log N)$ **then** $dict\_reg[msg.id] \leftarrow req$ ▷ Request is registered

Wireless Nodes Registration Process:

$k \leftarrow 0, req \leftarrow \texttt{genReq}(id)$ ▷ k counts the number of received partial seeds; id is the identity of the node
$lep \leftarrow 0$ ▷ lep records the epoch number of the last registration phase the node has participated
$secret \leftarrow \texttt{randStr}()$
$req.lsp \leftarrow 0$ ▷ $req.lsp$ records the last sub-phase in which a partial seed is received, initially set to 0
$req.last\_pseed \leftarrow nil$ ▷ $req.lsp$ records the last partial seed the node received
while $(k < a_2 \log N + 1)$ **do** ▷ a_2 is some sufficiently large constant
    $ch \leftarrow \texttt{random}(1, c)$
    repeat
        $msg \leftarrow \texttt{listen}(ch)$
    until $msg.type = REG\_START$
    if $(lep \neq msg.ep)$ **then** ▷ Start over if cannot collect $\Theta(\log N)$ partial seeds within one epoch
        $lep \leftarrow msg.ep, k \leftarrow 0, secret \leftarrow \texttt{randStr}()$
        $req.lsp \leftarrow 0, req.last\_pseed \leftarrow nil, req.pseed\_list \leftarrow nil$
    if $(msg.sp < (msg.q)/2$ **and** $req.id \in msg.list\_id)$ **then**
        continue ▷ Try to obtain only one partial seed during the first half
    if $(msg.sp \geq (msg.q)/2$ **and** $req.id \notin msg.list\_id)$ **then**
        continue ▷ Active during the second half only if already on the registration list
    for $(r \leftarrow 1$ to $a_1 \log N)$ **do**
        if $(\texttt{random}(1, 2^{r-1}) = 1)$ **then** ▷ Broadcast with probability $1/2^{r-1}$
            $\texttt{broadcast}(\langle REG\_REQ, id, \texttt{hash}^{(msg.q-msg.sp)}(secret), req.lsp, req.last\_pseed\rangle, msg.seq[r])$
        $msg \leftarrow \texttt{listen}(msg.seq[r])$
        if $(msg.type = REG\_RESP$ **and** $msg.id = id)$ **then**
            if $(k < a_2 \log N)$ **then** ▷ Do not record the $(\Theta(\log N) + 1)$st partial seed
                $req.lsp \leftarrow msg.sp, req.last\_pseed \leftarrow msg.pseed$
                $req.pseed\_list \leftarrow req.pseed\_list \cup \{msg.pseed\}$
            $k \leftarrow k + 1$
$req.seed \leftarrow \texttt{xor}(req.pseed\_list)$ ▷ Generate the final seed by bit-wise $\texttt{xor}$ of partial seeds

Figure 3.2: Registration phase pseudocode of SYBILCAST.

the base station sends out a channel hopping sequence on a randomly chosen channel instructing each unregistered node that receives the message to use this sequence for the remainder of this sub-phase. (b) In each of the following $\Theta(\log N)$ rounds within the

current sub-phase, the base station will listen to the channel specified by the previous sequence until a request is received. If multiple nodes send out requests simultaneously, a collision occurs and all requests are lost. If the base station successfully receives a request (i.e., if only one node broadcasts and there is no jamming) from a node that is eligible to receive partial seeds, then it sends back a response containing a partial seed. It also records which partial seeds are distributed to which request.

In each sub-phase, the base station keeps silent once a partial seed is sent, which implies the base station will send out at most one partial seed in each sub-phase.

Each honest node that wants to register an identity acts as follows: (a) Initially, it listens on a randomly chosen channel until a channel hopping sequence (for one registration sub-phase) is received. (b) If it is eligible to compete for partial seeds in this sub-phase, then in each of the following $\Theta(\log N)$ rounds within current sub-phase, it broadcasts its request with probability $1/2^{(r-1)}$ on the specified channel, where r is the round number within the current sub-phase. If the node receives a response which contains a partial seed for it, it records that partial seed. (c) It repeats (a) and (b) until $\Theta(\log N)$ partial seeds are collected or this registration phase is finished.

During the registration phase, the base station needs to determine which partial seeds are assigned to which requests. To ensure authenticity, each node uses a *hash chain* to prove that all of its messages come from the same node, i.e., itself. A hash chain is a technique which works as follows: before each node enters a registration phase, it generates a $secret$ which is known only to itself. Each time a node sends out a request, it also includes $\texttt{hash}^{(q-sp)}(secret)$, where q denotes the total number of sub-phases in this registration phase, and sp denotes the current sub-phase number[6]. Every time the base station receives a request from a node, it iterates the hash function for an appropriate number of times and verifies that the (recursive) hash of the current received value matches the last received value. If the result indeed matches, then the base station can be assured that it is talking to the same node (as $secret$ is only known to the node itself and we have assumed the hash function is pre-image resistant). Note, malicious nodes can still create multiple secrets and try to register multiple identities.

Another issue is coping with partial seeds that are lost. If a node does not receive

[6] $\texttt{hash}^{(i)}(msg)$ means recursively apply $\texttt{hash}(\ldots)$ for i times. E.g., $\texttt{hash}^{(2)}(msg) = \texttt{hash}(\texttt{hash}(msg))$.

a partial seed (due to jamming), then the base station learns of this when it receives the next message from that node (which is authenticated by the hash chain). After receiving all the required partial seeds, the node participates in one additional registration sub-phase, ensuring that the base station knows it has received the last partial seed.

Since the malicious users can intercept a partial seed with at most constant probability in each registration sub-phase, by requiring $\Theta(\log N)$ partial seeds, we ensure that even if the malicious users overhear some of the partial seeds, they cannot learn the final seed of an honest node, with high probability. (If they indeed learn the final seed, they could jam all communication to this honest node during the coming data phases.) More precisely, we have:

Lemma 3.1. *For every request from an honest node that completes its registration, at least one of the partial seeds is not intercepted by any malicious user, w.h.p.*

Proof. To obtain the partial seed sent out by the base station in any registration sub-phase, the malicious users must either: (a) know the channel hopping sequence of that sub-phase; or (b) listen on the channel on which the base station broadcasts the partial seed. By assumption, the malicious users can listen to at most $c/20$ channels in any one round, hence they can only obtain the channel hopping sequence or monitor the channel on which the partial seed is broadcast each with probability at most $1/20$. Since channel hopping sequences are chosen randomly and independently for each sub-phase, and since each node has to obtain $\Theta(\log N)$ partial seeds before it can be registered, we know the probability that the malicious nodes know the final seed for a particular honest node is at most $(1/20 + 1/20)^{\Theta(\log N)}$; i.e., $O(1/N^k)$ for any constant $k \geq 1$. Take a union bound over the $O(N)$ honest nodes and we have the lemma. □

3.3.2 Data Phase

For an epoch i, there are $\Theta((id(i) + c)\log^2 N)$ rounds in the corresponding data phase.

Data phases serve two purposes. The most important one is to deliver data. In each round of a data phase, the base station will randomly choose a registered identity and send a data packet to it. Since the base station knows the channel hopping sequence (based on the final seed obtained during registration) and the malicious users do not (with high probability), we can ensure that each data packet for an honest node is deliv-

Base Station Data Phase for an Epoch:

for (every registered request req in $dict\_reg$) **do**
  $req.nonce\_list \leftarrow \emptyset$ ▷ $nonce\_list$ stores the nonces the node should have received
for ($r \leftarrow 1$ to $a_3(sys\_size + c)\log^2 N$) **do** ▷ a_3 is some sufficiently large constant
  $req \leftarrow$ `randReq()` ▷ `randReq()` returns a randomly chosen registered identity from $list\_reg$
  $seq \leftarrow$ `prng`$(req.seed)$
  $nonce \leftarrow$ `randStr()`
  `broadcast`$(\langle DATA, req.id, data, nonce\rangle, seq[r])$ ▷ $data$ is a portion of the requested data
  $req.nonce\_list = req.nonce\_list \cup \{nonce\}$
  for (each registered identity's request $req' \neq req$) **do**
    if (`prng`$(req'.seed)[r] = seq[r]$) **then**
      $req'.nonce\_list = req'.nonce\_list \cup \{nonce\}$

Wireless Nodes Data Phase for an Epoch:

$seq \leftarrow$ `prng`$(req.seed)$ ▷ req is the node's own request
$reg.nonce\_list \leftarrow \emptyset$
for ($r \leftarrow 1$ to $a_3(sys\_size + c)\log^2 N$) **do**
  $msg \leftarrow$ `listen`$(seq[r])$
  if ($msg.type = DATA$) **then**
    $req.nonce\_list = req.nonce\_list \cup \{msg.nonce\}$
    if ($msg.id = req.id$) **then**
      store $msg.data$

Figure 3.3: Data phase pseudocode of SYBILCAST.

ered with constant probability. (The only way in which a packet is lost is if the malicious users successfully include the channel in the $c/20$ channels they can jam.)

The second purpose is to transmit nonces to the registered identities. Each data packet contains a unique, randomly chosen nonce. All the identities listening on the channel specified by the sequence calculated from their final seeds—whether or not they are the intended recipient of the packet—should record the nonce. As a result, by the end of each data phase, each honest node will have many nonces which can later be used for verification.

The above description is summarized in the pseudocode in Figure 3.3.

3.3.3 Verification Phase

The verification phase is the last key component of SYBILCAST. It is summarized in the pseudocode in Figure 3.4. The purpose of the verification phase is to let the base station spot and eliminate sybil identities.

For an epoch i, the verification phase consists of $\Theta(\log N)$ sub-phases each of length $id(i) + c$. Upon entering the verification phase, each honest node that has registered an identity randomly chooses c rounds in each sub-phase, and sends out a

verification packet in each of these selected rounds on the channel specified by its final seed. The verification packet contains all the nonces the node has collected during the preceding data phase in encrypted form, using the final seed as the key.

Notice, the verification process is on a per-epoch basis. In particular, during the verification phase of epoch i, verification packets from registered identities contain only nonces collected during the data phase of epoch i. When the data requested is large, nodes may need multiple data phases and verification phases to complete their requests. (Nevertheless, nodes only need to register once for each request.)

On the other hand, the base station's work during verification is simple. It randomly chooses a channel in each round and waits for verification packets. Upon receiving one such packet, it checks if the corresponding identity has provided enough nonces to prove it is not a sybil identity. If the base station receives no packets from an identity, or if all received packets from an identity provide insufficient nonces, then the base station will assume the identity is sybil and eliminate it. The precise meaning of "sufficient" will be defined in Section 3.4.

We also note here, during the verification phases, the base station will use the nonces provided by registered identities to identify which data packets have already been successfully delivered; the base station will resend undelivered data in future data phases.

To ensure honest nodes cannot fail verification due to jamming or contention during the verification phase, SYBILCAST guarantees the following property:

Lemma 3.2. *Each registered honest node will be successfully verified by the base station w.h.p. in every epoch, as long as it has collected sufficient nonces during the preceding data phase.*

Proof. If an honest node p has enough nonces, then the only possible reasons that verification may fail are: (a) the honest node and the base station choose different channels; (b) collisions among multiple nodes sending verification messages concurrently; and/or (c) malicious jamming.

We now calculate the probability that the base station misses all verification messages from an honest node p during one verification phase. In the following proof, we call every round in which node p sends a verification message as a *slot*.

In any epoch i, during the verification phase, there are $\Theta(\log N)$ sub-phases each

Base Station Verification Phase for an Epoch:

$unver\_list \leftarrow \emptyset$ ▷ The list of unverified requests
for (every req in $dict\_reg$) **do**
    $unver\_list \leftarrow unver\_list \cup \{req.id\}$
for ($sp \leftarrow 1$ to $a_4 \log N$) **do** ▷ a_4 is some sufficiently large constant
    for ($r \leftarrow 1$ to $sys\_size + c$) **do**
        $msg \leftarrow \texttt{listen}(\texttt{random}(1, c))$
        if ($msg.type = VER\_RESP$) **then**
            $count \leftarrow 0$
            for (every $nonce$ in msg) **do**
                if ($nonce \in dict\_reg[msg.id].nonce\_list$) **then**
                    $count \leftarrow count + 1$
                    Mark the corresponding portion of the data as delivered
            if ($count$ is sufficiently large) **then**
                $unver\_list \leftarrow unver\_list \setminus \{msg.id\}$
for (every identity id in $unver\_list$ **and** every identity id that has received all the data) **do**
    $dict\_reg[id] \leftarrow nil$ ▷ Remove corresponding identity from the system

Wireless Nodes Verification Phase for an Epoch:

$seq \leftarrow \texttt{prng}(req.seed)$ ▷ req is the node's own request
$enc\_list \leftarrow \texttt{encrypt}(req.nonce\_list, req.seed)$ ▷ $\texttt{encrypt}(x, k)$ encrypts x with k as the key
for ($sp \leftarrow 1$ to $a_4 \log N$) **do**
    $rnd\_list \leftarrow \emptyset$
    while ($|rnd\_list| \leq c$) **do** ▷ Choose c random slots
        $rnd\_list \leftarrow rnd\_list \cup \{\texttt{random}(1, sys\_size + c)\}$
    for ($r \leftarrow 1$ to $sys\_size + c$) **do**
        if ($r \in rnd\_list$) **then**
            $round\_number \leftarrow sp \cdot (sys\_size + c) + r$
            $\texttt{broadcast}(\langle VER\_RESP, req.id, enc\_list\rangle, seq[round\_number])$
if (have received all data) **then exit**

Figure 3.4: Verification phase pseudocode of SYBILCAST.

containing $\Theta(c)$ slots. Hence, p has $\Theta(c \log N)$ slots in total. In each slot, the base station chooses the same channel with p with probability $1/c$. On the other hand, in each slot, another node may collide with p with probability $(c/(id(i) + c)) \cdot (1/c) = 1/(id(i) + c)$. (For another node p' to collide with p in a particular slot, p' must have chosen the same slot and same channel with p in that sub-phase.) Since there are at most $(id(i) - 1)$ other nodes, for a given slot, the probability that there is no collision is $(1 - 1/(id(i) + c))^{id(i)-1} \approx e^{-(id(i)-1)/(id(i)+c)} \geq 1/e$. Lastly, the malicious nodes may jam each slot with probability at most $(1/20) \cdot (1 - 1/N) + 1 \cdot (1/N) \leq 1/19$, as we have shown in Lemma 3.1 that w.h.p. they don't know any node's final seed.

Combining these facts, we know in each slot, the verification message from p will be missed by the base station with probability at most $1 - ((1/e) \cdot (18/19) \cdot (1/c)) = 1 - 18/19ec$. Thus, after one verification phase, the verification message from p will be missed by the base station with probability at most $(1 - 18/19ec)^{\Theta(c \log N)}$, i.e.,

$O(1/N^k)$ for any constant k. Take a union bound over the $O(N)$ honest nodes and the lemma follows. □

Finally, we note here that our verification scheme is different from traditional radio resource testing in the sense that we do not use *immediate verification* but rather *deferred verification*. More specifically, instead of asking the nodes that receive a nonce during a data round to send back an acknowledgment immediately, we *defer* this process to the verification phase. This design may allow sybil identities to remain in the system longer, but it brings a key benefit: immediate verification suffers from message spoofing and wireless jamming. Malicious nodes can spoof verification packets or jam the channels and hence force the base station to incorrectly remove honest nodes. This is not possible in SYBILCAST.

3.4 Analysis

In this section, we will analyze the total time consumption for an honest node to complete a download request. We will also show that SYBILCAST can constrain the maximum number of sybil identities and provide asymptotically optimal transmission speed.

We assume an honest node p is requesting data m. We use $\texttt{contention}(p, m)$, or n^* for short, to denote the maximum number of *active nodes* during the entire request. An active node may be in one of two states: *joining* the network (i.e., registering), or *downloading* data. We use d^* to denote the maximum number of *registered identities* during the entire request. A registered identity can be an honest node or a sybil identity.

Before presenting the detailed analysis, we will first introduce the following two well-known Chernoff Bounds [37]. They will be constantly used in the proofs.

Theorem 3.3 (Chernoff Bounds [37])**.** *Let $X_1, X_2, \cdots, X_n$ be independent Poisson trials such that $\mathbb{P}(X_i) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}(X)$. Then the following Chernoff Bounds hold:*

$$\textit{for } 0 < \delta \leq 1, \mathbb{P}(X \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/3}; \tag{3.1}$$

$$\textit{for } 0 < \delta < 1, \mathbb{P}(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}. \tag{3.2}$$

3.4.1 Total Time Complexity

We first analyze how many rounds it will take for p to register with the base station.

Lemma 3.4. *An honest node p completes its registration process with the base station in $O((n^* + c)c\log^3 N + (d^* + c)\log^2 N)$ rounds, w.h.p.*

Proof. We begin by proving two claims:

Claim 3.4.1. *In a registration sub-phase, assume there are exactly w honest nodes competing for the partial seed. Then, if $w < c$, each of these w nodes gets the partial seed with probability $\Omega(1/c)$. Otherwise, if $w \geq c$, each of these w nodes gets the partial seed with probability $\Omega(1/w)$.*

Proof. We first consider the $w < c$ case. In this case, with at least constant probability, the w honest nodes will spread on at least $w/3$ channels. To see this, assume W is the event that these nodes spread on at most $w/3$ channels. We have:

$$\begin{aligned}\mathbb{P}(W) &= \binom{c}{w/3}\left(\frac{w/3}{c}\right)^w \leq \left(\frac{ec}{w/3}\right)^{w/3}\left(\frac{w/3}{c}\right)^w = e^{w/3}\left(\frac{w/3}{c}\right)^{2w/3} \\ &< e^{w/3}\left(\frac{1}{3}\right)^{2w/3} = \left(\frac{e}{9}\right)^{w/3} \leq \left(\frac{e}{9}\right)^{1/3}\end{aligned}$$

Hence, with probability $\Omega(w/c)$, the base station will broadcast the channel hopping sequence on a channel where there is at least one honest node. Meanwhile, since the malicious nodes can listen to or jam at most $1/20$ fraction of all channels at any time, we know with probability $19/20 \cdot \Omega(w/c) = \Omega(w/c)$, the channel hopping sequence will be successfully received by some honest nodes and will be unknown to the malicious nodes. Conditioned on these two events, assume $w' \leq w$ honest nodes receive the channel hopping sequence; then in one of the following $\Theta(\log N)$ rounds, there must exist a round in which each of these w' nodes sends out its request with probability p_s, where $1/2w' \leq p_s \leq 2/w'$. In that particular round, with probability at least $19/20$ there will be no malicious interference, hence each of these w' nodes can successfully obtain the partial seed with probability $19/20 \cdot p_s \cdot (1-p_s)^{w'-1} \approx \Theta(1/w') = \Omega(1/w)$.

Combining these results with the previous condition which happens with probability $\Omega(w/c)$ will immediately lead to the first part of our claim.

We then consider the $w \geq c$ case. In this case, with at least constant probability, these w honest nodes will spread on at least $c/3$ channels. To see this, assume W' is the event that these nodes spread on at most $c/3$ channels, and thus we have:

$$\begin{aligned}\mathbb{P}(W') &= \binom{c}{c/3}\left(\frac{c/3}{c}\right)^w \leq \left(\frac{ec}{c/3}\right)^{c/3}\left(\frac{1}{3}\right)^w = (3e)^{c/3}\left(\frac{1}{3}\right)^w \\ &\leq (3e)^{c/3}\left(\frac{1}{3}\right)^c = \left(\frac{e}{9}\right)^{c/3} \leq \left(\frac{e}{9}\right)^{1/3}\end{aligned}$$

Hence, with at least some constant probability, the base station will broadcast the channel hopping sequence on a channel where there is at least one honest node. Meanwhile, since the malicious nodes can listen to or jam at most $1/20$ fraction of all channels at any time, we know with probability $19/20 \cdot \Omega(1) = \Omega(1)$, the channel hopping sequence will be successfully received by some honest nodes and will be unknown to the malicious nodes. Conditioned on these two events, assume $w' \leq w$ honest nodes receive the channel hopping sequence; then in one of the following $\Theta(\log N)$ rounds, there must exist a round in which each of these w' nodes sends out its request with probability p_s, where $1/2w' \leq p_s \leq 2/w'$. In that particular round, with probability at least $19/20$ there will be no malicious interference, hence each of these w' nodes can successfully obtain the partial seed with probability $19/20 \cdot p_s \cdot (1-p_s)^{w'-1} \approx \Theta(1/w') = \Omega(1/w)$. Combining these results with the previous condition which happens with probability $\Omega(1)$ will immediately lead to the second part of our claim. □

Claim 3.4.2. *In any epoch i, if by the halfway point of the registration phase there are $x \leq id(i)$ requests from the honest nodes in the registration list, then w.h.p. these x requests will each obtain $\Theta(\log N)$ partial seeds by the end of that registration phase, and the honest nodes that sent them will be able to join the network.*

Proof. Let $\mathcal{Y} = \{y_1, y_2, \cdots, y_x\}$ denote the set of honest nodes that sent these x requests, and let Y_j be a random variable which denotes the number of partial seeds the jth honest node in $\mathcal{Y}$ received during the second half of the registration phase.

According to Claim 3.4.1, during the second half of the registration phase, in each sub-phase, each of these x honest nodes gets a partial seed with probability at least $\min\{\Omega(1/c), \Omega(1/x)\} = \Omega(1/(x+c))$. Since there are $\Theta((id(i)+c)\log N)$ independent sub-phases during the second half of the registration phase, we conclude that $\mathbb{E}(Y_j) = \Theta((id(i)+c)\log N) \cdot \Omega(1/(x+c)) = \Omega(\log N)$. Applying a Chernoff Bound where $\mathbb{P}(Y_j \leq (1-\delta)\mathbb{E}(Y_j)) \leq e^{-\mathbb{E}(Y_j)\delta^2/2}$ with $\delta = 1/2$, we conclude $\mathbb{P}(Y_j \leq \Omega(\log N)) \leq O(1/N^k)$ for any constant k. Hence, after the second half of the registration phase, each of these x requests will have $\Theta(\log N)$ partial seeds w.h.p. $\square$

We now prove the lemma. Claim 3.4.2 implies that in any epoch, if a request from an honest node is in the registration list by the halfway point of the corresponding registration phase, then the honest node which sent the request will be registered, w.h.p., in that epoch (i.e., within $\Theta((id(i)+c)\log^2 N)$ rounds). Recall that d^* is the maximum number of registered identities while p is requesting m, which implies $id(i) \leq d^*$. We can thus conclude that once p's request is in the registration list by the halfway point of a registration phase, it will be registered after $O((d^*+c)\log^2 N)$ rounds w.h.p. In fact, the total number of rounds consumed since the beginning of this last epoch is $O((d^*+c)\log^2 N)$ as well.

We now calculate the number of rounds before p is added to the registration list during the first half of some registration phase.

Assume that after some l epochs denoted as i_1 to i_l, at last in epoch i_{l+1}, node p successfully obtains its first partial seed during the first half of the registration phase. We also define the following: in the first half of any registration phase, we call a sub-phase *open* if the registration list is not full at the beginning of that sub-phase.

By Claim 3.4.1, in an open sub-phase, node p has a probability of $\Omega(1/c)$ to be added to the registration list if there are fewer than c honest nodes competing. If there are at least c honest nodes competing, then p has a probability of $\Omega(1/n^*)$ to be added to the registration list. Therefore, in expectation, $O(n^*+c)$ open sub-phases will elapse before p can successfully obtain its first partial seed. In fact, w.h.p., node p gets its first partial seed within $O((n^*+c)\log N)$ open sub-phases, as $(1-\Omega(1/(n^*+c)))^{O((n^*+c)\log N)} \approx O(1/N^k)$ for any constant k.

Since each epoch i_j, where $1 \leq j \leq l$, contains at least $id(i_j)$ open sub-phases,

we have $\sum_{j=1}^{l} id(i_j) = O((n^* + c)\log N)$. As a result, we can conclude that after $\sum_{j=1}^{l} O((id(i_j) + c)\log^2 N) = \log^2 N \cdot \sum_{j=1}^{l} O(id(i_j) + c) = \log^2 N \cdot O((n^* + c)c\log N) = O((n^*+c)c\log^3 N)$ rounds, w.h.p. p's request can get into the registration list during the first half of a registration phase. (Notice, we have ignored the data rounds and the verification rounds in these l epochs, as for every epoch, the length of the data phase and the verification phase is asymptotically equal to or less than the length of the corresponding registration phase.)

To sum up, w.h.p. the total number of rounds consumed before p is registered is: $O((n^* + c)c\log^3 N) + O((d^* + c)\log^2 N) = O((n^* + c)c\log^3 N + (d^* + c)\log^2 N)$. This proves the lemma. □

We now analyze the number of rounds consumed while p is downloading data.

Lemma 3.5. *After registration, if $|m| = \Omega(\log N)$, p can finish downloading m in $O(d^*|m|)$ rounds, w.h.p.*

Proof. After p registers, it has to participate in another series of epochs before it can complete the reception of message m. During these epochs, there are at most d^* identities. Moreover, w.h.p. the malicious nodes may jam each data round with probability at most $1/20$. Hence, in each data round, node p receives a data packet with probability at least $(1/d^*) \cdot (1 - (1/20) \cdot (1 - 1/N) - 1 \cdot (1/N)) = \Omega(1/d^*)$. Since p's data is of size $|m|$, we expect $O(d^*|m|)$ data rounds will elapse before p's request can be completed. With the assumption $|m| = \Omega(\log N)$ and a standard Chernoff Bound (see Equation 3.1), we know p will get all requested data within $O(d^*|m|)$ data rounds w.h.p.

On the other hand, since in any epoch the data phase is (asymptotically) of the same length as the registration phase and is (asymptotically) longer than the verification phase, and since the last epoch for downloading m is at most twice as long as the next-to-last epoch, we can conclude that w.h.p. the total number of rounds consumed while m is being delivered to p is $O(d^*|m|)$. □

3.4.2 Constraining Sybil Identities

Lemma 3.4 and Lemma 3.5 do not guarantee each honest node a fair share of the bandwidth, as d^* includes both real and sybil identities. In this subsection, we show that

SYBILCAST guarantees an upper bound on the number of sybil identities and hence can provide asymptotically optimal transmission speed for each honest node.

We first introduce the following definition, which describes the event where a sybil identity does not get a nonce that it should later report to the base station.

Definition 3.6. *For any given sybil identity q, we say a data round is a* losing round *for q if in this round: (a) q is assigned to listen on channel b according to its final seed; (b) the base station broadcasts a data packet (with a nonce in it) on channel b; and (c) none of the t malicious users listen on channel b.*

We now argue that if there are more than $12t$ registered sybil identities, then all but $12t$ of them suffer a large number of losing rounds, with high probability. This lemma implies a lower bound on the length of the data phase (i.e., it has to be long enough to ensure that the sybil identities suffer enough losses), and it shows the threshold on the number of losses that the base station will tolerate before indicating a sybil identity.

Lemma 3.7. *In any epoch i, if $s(i) \geq 12t$ and $id(i) \leq N + 12t$, then after k data rounds where $k \geq 225 \cdot (id'(i) + c)/t \cdot \ln(1/\varepsilon)$, for sufficiently small $\varepsilon < 1/(2N^{13t})$, with probability at least $1 - 2N^{13t}\varepsilon$, at least $s(i) - 12t$ sybil identities will each suffer at least $\frac{id'(i)+2c}{id'(i)\cdot c} \cdot \frac{k}{17}$ losing rounds*[7].

Proof. We first show that in any data round, if there are $12t \leq s(i) \leq c$ sybil identities, then with probability at least 0.99, at least $s(i)/3$ channels will each be assigned at least one sybil identity by the final seeds.

To see this. Suppose W is the event that at most $s(i)/3$ channels are each assigned with at least one sybil identity. With the assumption $c \geq 20t$, we know:

$$\begin{aligned}
\mathbb{P}(W) &\leq \binom{c}{s(i)/3}\left(\frac{s(i)/3}{c}\right)^{s(i)} \leq \left(\frac{ec}{s(i)/3}\right)^{s(i)/3}\left(\frac{s(i)/3}{c}\right)^{s(i)} \\
&= e^{s(i)/3}\left(\frac{s(i)/3}{c}\right)^{2s(i)/3} < e^{s(i)/3}\left(\frac{1}{3}\right)^{2s(i)/3} \\
&= \left(\frac{e}{9}\right)^{s(i)/3} \leq \left(\frac{e}{9}\right)^{4t} \\
&\leq \left(\frac{e}{9}\right)^{4} < \frac{1}{100}
\end{aligned}$$

[7] We have made no effort to optimize the constants, focusing on ease of explanation.

We now prove the lemma. Fix a set $\mathcal{S}$ containing exactly $(12t+1)$ sybil identities that have finished registration either in the current registration phase, or in some prior epoch. We will show, with probability at least $1-2\varepsilon$, at least one identity in $\mathcal{S}$ suffers $\frac{id'(i)+2c}{id'(i)\cdot c}\cdot\frac{k}{17}$ losing rounds. We then take a union bound over the $O(N^{13t})$ possible sets $\mathcal{S}$ of size $12t+1$. From this we conclude, with probability at least $1-2N^{13t}\varepsilon$, the largest set of sybil identities that none of which suffer $\frac{id'(i)+2c}{id'(i)\cdot c}\cdot\frac{k}{17}$ losing rounds is of size at most $12t$.

As already shown, during the k data rounds, in expectation, sybil identities in $\mathcal{S}$ will be assigned to at least $|\mathcal{S}|/3 \geq 4t$ channels in $0.99k$ rounds. Applying a standard Chernoff Bound (see Equation 3.2), we know that when $k \geq \frac{2}{0.99\delta^2}\ln(1/\varepsilon)$, with probability at least $1-\varepsilon$, sybil identities in $\mathcal{S}$ will be assigned to at least $|\mathcal{S}|/3 \geq 4t$ channels in at least $(1-\delta)0.99k$ rounds. Here, $0<\delta<1$. We call these $(1-\delta)0.99k$ rounds as $\mathcal{R}$.

Recall $id'(i)$ denotes the number of registered identities at the beginning of the data phase of epoch i. For each round in $\mathcal{R}$, at least one sybil identity in $\mathcal{S}$ will receive a nonce from the base station with probability at least $|\mathcal{S}|/id'(i) + (1-|\mathcal{S}|/id'(i))\cdot(4t/c) \geq 12t/id'(i) + (1-12t/id'(i))\cdot(4t/c) = 4t\cdot(3/id'(i)+(1-12t/id'(i))/c) = 4t\cdot(id'(i)+3c-12t)/(id'(i)\cdot c) > 4t\cdot(id'(i)+2c)/(id'(i)\cdot c)$.

Moreover, for each round in $\mathcal{R}$, if some sybil identities in $\mathcal{S}$ should receive a nonce from the base station, the nonce will be missed by the corresponding sybil identities with probability at least $1/4$. To see this, consider an arbitrary round in $\mathcal{R}$. Assume in that round, the sybil identities in $\mathcal{S}$ are assigned to $x \geq |\mathcal{S}|/3 \geq 4t$ channels. On the one hand, since w.h.p. the malicious nodes do not know honest nodes' final seeds, hence if there exist many honest nodes on these x channels, then the base station will likely to choose each channel equally likely, resulting the malicious users to lose the nonce with probability (approximately) at least $1-t/x \geq 1-t/4t = 3/4$. On the other hand, if there are no honest nodes on these x channels, then any $x-t$ of these channels will contain at least $x-t$ sybil identities, resulting the malicious users to lose the nonce with probability at least $(x-t)/|\mathcal{S}| \geq (4t-t)/|\mathcal{S}| = 3t/(12t+1) \approx 1/4$.

Therefore, during data phase, in expectation, sybil identities in $\mathcal{S}$ lose at least $(1-\delta)0.99k\cdot 4t\cdot(id'(i)+2c)/(id'(i)\cdot c)\cdot(1/4) = (1-\delta)0.99kt\cdot(id'(i)+2c)/(id'(i)\cdot c)$ nonces. Applying a Chernoff Bound (see Equation 3.2), we know that, with probability at least $(1-\varepsilon)^2$, sybil identities in $\mathcal{S}$ will in total lose at least $(1-\delta)^2\cdot 0.99kt\cdot$

$(id'(i)+2c)/(id'(i)\cdot c)$ nonces when $k \geq \frac{2}{0.99t(1-\delta)\delta^2} \cdot \frac{id'(i)\cdot c}{id'(i)+2c} \cdot \ln(1/\varepsilon)$. Since $id'(i)+c > (id'(i)\cdot c)/(id'(i)+2c)$, we know the above claim holds true for $k \geq \frac{2}{0.99t(1-\delta)\delta^2} \cdot (id'(i)+c) \cdot \ln(1/\varepsilon)$ as well.

Now, since each lost nonce affects at least one sybil identity in $\mathcal{S}$, we know that with probability at least $(1-\varepsilon)^2$, at least one sybil identity in $\mathcal{S}$ will lose at least $(1-\delta)^2 \cdot 0.99kt \cdot (id'(i)+2c)/(id'(i)\cdot c) \cdot (1/(12t+1)) \approx \frac{0.99(1-\delta)^2}{13} \cdot \frac{id'(i)+2c}{id'(i)\cdot c} \cdot k$ nonces when $k \geq \frac{2}{0.99t(1-\delta)\delta^2} \cdot (id'(i)+c) \cdot \ln(1/\varepsilon)$. With assignment $\delta = 1/10$, we can conclude when $k \geq 225 \cdot (id'(i)+c)/t \cdot \ln(1/\varepsilon)$, with probability at least $(1-\varepsilon)^2 \geq (1-2\varepsilon)$, at least one sybil identity in $\mathcal{S}$ will suffer at least $\frac{id'(i)+2c}{id'(i)\cdot c} \cdot \frac{k}{17}$ losing rounds.

Now, consider all possible sets $\mathcal{S}$ of size $12t+1$. We have assumed that there are at most $N+12t$ registered identities at the beginning of epoch i. Hence at the end of the registration phase for epoch i, there are at most $2(N+12t)$ identities, and the number of possible sets $\mathcal{S}$ is bounded by $\binom{2N+24t}{12t+1} \leq (e(2N+24t)/(12t+1))^{12t+1} \leq N^{13t}$. Thus, by a union bound, the probability that every set $\mathcal{S}$ of size $12t+1$ has at least one identity suffer at least $\frac{id'(i)+2c}{id'(i)\cdot c} \cdot \frac{k}{17}$ losing rounds is at least $1-2N^{13t}\varepsilon$. □

Lemma 3.7 implies that if the malicious users register too many sybil identities, then over some period of time, most of these sybil identities will lose a significant fraction of the nonces they should receive. The base station can leverage this fact to eliminate most of the sybil identities during verification phases. We can also here identify precisely the threshold for rejecting an identity during the verification phase: *if the data phase in epoch i is of length $len(i)$, then reject an identity as a sybil if it loses at least $\frac{id'(i)+2c}{id'(i)\cdot c} \cdot \frac{len(i)}{17}$ nonces that were sent to it.* This results in the following lemma:

Lemma 3.8. *In any epoch, when the verification phase completes, there are at most $12t$ remaining sybil identities, w.h.p.*

Proof. We proceed by induction over epochs. The base case, epoch one, follows immediately from the fact that no more than $12t$ identities are admitted in epoch one. Consider epoch i, and assume that there are at most $12t$ sybil identities remaining at the end of epoch $i-1$. Since there are at most N honest identities, this implies that $id(i) \leq N+12t$.

Now, from Lemma 3.7 and our protocol description, we can show that the length

of the data phase of epoch i, which is denoted as $len(i)$, is sufficiently long so that w.h.p. all but at most $12t$ sybil nodes are always rejected during the verification phase. Notably, for $2N^{13t}\varepsilon$ to be polynomially small in N, the number of data rounds we need is $\Omega((id'(i)+c)/t \cdot \log(2N^{13t})) = \Omega((id'(i)+c) \cdot \log N)$. This always holds true as $id'(i) \leq 2 \cdot id(i)$, and the length of the data phase is $\Theta((id(i)+c)\log^2 N)$. □

With Lemma 3.9, we can now bound the number of sybil identities:

Lemma 3.9. *For any honest node p, in every round during its request, the number of sybil identities in the network is at most $O(n^*)$, w.h.p.*

Proof. Lemma 3.8 states that at the end of any verification phase, w.h.p. at most $12t$ sybil identities remain in the system. Hence, among the $id(i)$ identities at the beginning of any epoch i, at most $12t$ are sybil identities created by the t malicious nodes; and the remaining $id(i) - 12t$ identities are all honest identities each created by an honest node, i.e., $id(i) - 12t \leq n^*$. This further implies $id(i) \leq O(n^*)$ since $n^* \geq t$. On the other hand, SYBILCAST allows at most $id(i)$ new identities to register in epoch i. These facts immediately yields the lemma. □

We now turn our attention to the honest nodes. In particular, we want to show that no honest nodes will be incorrectly removed. To prove this, we first show—even subject to lost messages due to jamming—an honest identity will not miss too many nonces.

Lemma 3.10. *In any epoch i, after k data rounds where $k \geq 4860 \cdot (id'(i)+c) \cdot \ln(1/\varepsilon')$, with probability at least $1-\varepsilon'$, an honest node p will lose at most $\frac{id'(i)+c}{id'(i)\cdot c} \cdot \frac{k}{18}$ nonces.*

Proof. According to our protocol, in a data round, for an honest node p, it will receive a nonce with probability at most $1/id'(i) + (1 - 1/id'(i)) \cdot (1/c) = (id'(i) + c - 1)/(id'(i) \cdot c) \leq (id'(i)+c)/(id'(i) \cdot c)$. Moreover, when p indeed should receive a nonce, it will lose that nonce with probability at most $1/20$ due to adversarial jamming.

Therefore, in expectation, after k data rounds, p will lose at most $(id'(i)+c)/(id'(i) \cdot c) \cdot (k/20)$ nonces. Since each round is independent, apply a Chernoff Bound (see Equation 3.1), we know when $k \geq \frac{60}{\delta^2} \cdot \frac{id'(i)\cdot c}{id'(i)+c} \cdot \ln(1/\varepsilon')$, with probability at least $1-\varepsilon'$, p will lose at most $(1+\delta) \cdot (id'(i)+c)/(id'(i) \cdot c) \cdot (k/20)$ nonces. Since $id'(i) + c > (id'(i) \cdot c)/(id'(i)+c)$, we know the above claim holds true when $k \geq \frac{60}{\delta^2} \cdot (id'(i)+c) \cdot \ln(1/\varepsilon')$ as well. With assignment $\delta = 1/9$, the lemma follows. □

Much as in the proof of Lemma 3.8, one can observe and show that the length of the data phase for any epoch i is sufficiently long so that w.h.p. honest node p, by Lemma 3.10, loses at most $\frac{id'(i)+c}{id'(i)\cdot c} \cdot \frac{len(i)}{18}$ nonces, which is smaller than the threshold for removing an identity. Taking a union bound over all the $O(N)$ honest nodes, and combining with Lemma 3.2, the following lemma immediately follows:

Lemma 3.11. *No honest nodes will be (incorrectly) removed before its request is completed, w.h.p.*

Lastly, we state our main claim:

Theorem 3.12. *If an honest node p requests data m where $|m| \geq c^2 \log^3 N$, and if there are at most $\texttt{contention}(p, m)$ concurrently active nodes during the request, then p can get the data in $O(|m| \cdot \texttt{contention}(p, m))$ time, w.h.p.*[8]

Proof. First, notice Lemma 3.11 guarantees honest nodes will not be incorrectly removed before their requests are completed. On the other hand, Lemma 3.9 shows that during the request of m, the number of sybil identities is bounded by $O(n^*)$. As a result, d^*, the maximum number of registered identities during the request of m, is also bounded by $O(n^*)$, or $O(\texttt{contention}(p, m))$. Hence, Lemma 3.4 and Lemma 3.5 immediately imply the theorem. □

Theorem 3.12 implies SYBILCAST is asymptotically optimal in transmission speed: even in an environment where there are no malicious nodes, any fair-share protocol will in the worst case take $\Theta(|m| \cdot \texttt{contention}(p, m))$ rounds to deliver $|m|$ packets to an honest node if there are $\texttt{contention}(p, m)$ active nodes in total.

As a note, we believe the constraint on the message size is not far from optimal. For example, it seems inevitable that for the base station and the client to exchange even one message with constant probability requires $\Omega(c)$ time; to achieve this single message exchange w.h.p. requires $\Omega(c \log N)$ time. (See, e.g., [13,36], for a discussion of lower bounds on simple communication in a jamming-prone wireless network.)

[8] In fact, one can also easily show the following guarantee which is somewhat stronger: in any round where the base station is delivering packets, if there are at most $\texttt{contention}_r(p, m)$ active real users in round r, then a given honest user p receives a data packet with probability $\Theta(1/\texttt{contention}_r(p, m))$; a constant fraction of the rounds are used to deliver data packets to honest users.

3.5 Summary and Discussion

In this chapter, we propose a new protocol called SYBILCAST that counters sybil attacks in multi-channel wireless networks. It limits the maximum number of sybil identities and provides asymptotically optimal transmission speed for honest nodes. It can also tolerate message spoofing and wireless jamming.

One natural question is whether we can achieve better performance if we limit malicious users' capabilities. For example, what if the malicious users cannot spoof messages? Unfortunately, this constraint does not help as much as one might expect, as the most time consuming part is registration, which cannot be reduced (in either time or channels) if we want to prevent the malicious users from eavesdropping on the final seed. Nevertheless, this constraint does eliminate the need for a hash chain. If the adversary cannot jam communication *or* spoof messages, then it may be feasible to reduce the length of the registration phase and/or the number of channels needed. This change might lead to more sybil identities, but the faster registration process may make it cost-effective.

Another question concerns whether knowing an upper bound on the maximum system size, i.e., N, is necessary. Here, we assume that a polynomial estimate of N, i.e., a constant-factor estimate of $\log N$, is known (even though the actual number of active nodes is unknown). Knowledge of $\log N$ serves two purposes in SYBILCAST: (a) keeping the nodes and the base station tightly synchronized during an epoch (as it is used to define the length of the epochs, phases, etc); and (b) ensuring that necessary events happen with high probability. It seems likely that SYBILCAST can be modified to work without knowing N, as the base station is always aware of the *current* system size. In this case, probabilistic guarantees hold with high probability with respect to the *current* system size, not N.

Lastly, can we leverage the real motivations of malicious users to simplify the protocol? Currently, we assume malicious users can be both *selfish* (e.g., hog bandwidth) and *adversarial* (e.g., disrupt communication). What if they are only selfish and conduct attacks only if such behavior can bring them more bandwidth? Is this an easier attack to prevent? To some extent, in the context of SYBILCAST, the answer seems to be negative. Both jamming and spoofing can be used for selfish purposes: mali-

cious users can jam honest users in data phases to stop them from obtaining sufficient nonces, and they can also spoof exit messages from honest users[9]. Both behaviors will result in honest users getting disconnected, and hence more bandwidth is allocated to the malicious users.

[9]Our current design assumes no honest nodes will withdraw requests halfway, but one can imagine in a more realistic implementation, nodes may be allowed to voluntary quit before their requests are completed. In that case, more bandwidth will be allocated to other remaining requests.

Chapter 4

Thwarting Sybil Attacks in Ad Hoc Wireless Networks

4.1 Introduction

Imagine the following scenario: a group of wireless devices are activated in a single hop wireless network. Some of them are honest while others are malicious. The honest devices are provided with no information regarding the size of the network or the identities of the other devices. The goal is to solve some standard distributed computing problem; e.g., the honest devices may need to reach agreement, vote on a proposal, simulate a shared object, or establish a fair schedule to share a limited resource.

Intuitively, this *unknown and anonymous wireless network* scenario seems hopeless. Honest devices are provided with no advance information on network size or participants, and there is no obvious way to distinguish them from malicious devices. Thus, malicious devices can commence a sybil attack in which they create many sybil identities (also called *sybils*) to bias a given distributed algorithm. For example, when running a consensus algorithm that depends on a majority vote, the malicious devices could create enough sybil identities to ensure they control a majority of voters.

In this chapter, we prove—perhaps surprisingly—that this scenario is not hopeless. We describe and analyze fully distributed algorithms that leverage radio resource testing strategies, so that honest devices can constrain the number of sybil identities to an asymptotically optimal limit. Our results answer open questions regarding sybil resistance in ad hoc networks and provide a general foundation for establishing trusted

computing in the increasingly untrustworthy world of wireless networking.

Results. We consider a single hop wireless network consisting of $c > 1$ channels. We divide time into synchronous slots and assume that the n devices (or *nodes*) are activated simultaneously. At most $t \leq \min\{n, c\}/\alpha$ of these devices are malicious, where $\alpha \geq 1$ is a sufficiently large constant. Honest devices do not know n or t in advance. We assume standard collision rules: if multiple devices send on the same channel during the same slot, the messages are lost due to a collision, which can be detected at the receivers' end. Collisions further complicate our task as: (a) malicious devices can intentionally jam (to cause collision); and (b) honest devices must deal with contention resolution even without knowledge of the network size. We assume the availability of standard asymmetric cryptographic primitives. This allows devices to generate unique public/private key pairs and use their public keys as identities. Because devices can sign their messages with their private keys, these identities are unforgeable. The challenge in our setting, therefore, is to prevent the malicious devices from creating *too many* sybil identities. The algorithms described below are randomized and their guarantees hold with high probability (i.e., with probability at least $1 - 1/n^{\beta}$ for any constant $\beta \geq 1$).

We begin by describing and analyzing a basic algorithm: SIMPLESYBILSIEVE. This algorithm terminates in $O(n \lg^2 n \cdot \max\{1, n/c\})$ time. It provides each honest node with a set of unforgeable identities that includes all other honest nodes and at most $O(t \cdot \max\{1, n/c\})$ sybil identities. Notice, when c is larger than n, the time complexity is $O(n \lg^2 n)$ and the number of sybil identities is bounded at an (asymptotically) optimal $O(t)$.[1] On the other extreme, when c is small compared to n, the time complexity grows to $O(n^2 \lg^2 n)$ and the sybil bound grows to $O(n)$. In settings where c is large or $O(n)$ sybils is tolerable, this basic algorithm is sufficient. On the other hand, in many settings, such as large networks, it might be preferable to obtain an optimal $O(t)$ bound on the number of sybils, even when $c < n$.

Motivated by this goal, we next present an augmented algorithm: SYBILSIEVE. This algorithm offers the same bound on the number of sybil identities as the basic algorithm, but is a factor of n slower. In exchange for this extra time, however, the algorithm provides two new strong guarantees: (a) all honest nodes terminate simulta-

[1] $O(t)$ sybil identities is clearly asymptotically optimal as the malicious nodes could behave honestly.

neously (hence providing "barrier synchronization"); and (b) all honest nodes agree on the same estimate of n that is within a constant factor of the actual count.

With SYBILSIEVE in hand, we introduce our strongest algorithm: SYBILSIEVEOPT. This algorithm builds on the useful guarantees of SYBILSIEVE to reduce the number of sybil identities to an optimal $O(t)$, even for $c < n$. Similar to SYBILSIEVE, its running time is also bounded by $O(n^2 \lg^2 n \cdot \max\{1, n/c\})$. As can be seen, these three algorithms provide users with trade-offs between time complexity and sybil resistance.

It is also worth noting that SYBILSIEVEOPT contains a subcomponent that can solve Monte Carlo Byzantine consensus in sybil-prone environments.[2] Both this consensus subroutine, and the synchronous termination and network size estimation implemented by SYBILSIEVE, may be of independent interest, as they simplify the bootstrapping of more advanced protocols in this anonymous wireless setting.

Towards the end of this chapter, we extend SIMPLESYBILSIEVE to the scenario in which collision detection is not available. The resulting protocol, which is named SIMPLESYBILSIEVEWCD (i.e., "SIMPLESYBILSIEVE without collection detection"), can provide same asymptotic bound in terms of running time and maximum number of sybil identities that may be accepted by honest nodes. Nevertheless, the constants behind the asymptotic notations are worse. As a result, we believe trade-offs exist between these two protocols as well.

4.2 Model and Problem Statement

We assume a single hop wireless network (i.e., a complete network) consisting of n devices (called *nodes* in the following) and $c > 1$ communication channels. To simplify calculations, we assume c is a power of 2. (If not, round down.)

We divide time into synchronous *slots* and assume all nodes start in the same slot. In each slot, each node can participate on one of the c available channels. It can then decide to send or receive. We assume a sending node cannot receive, and vice versa (i.e., the channel is half-duplex).

When a single node u sends on a channel k during a slot r, all nodes receiving on channel k during slot r receive u's message. If multiple nodes send on channel k during

[2]The consensus protocol maintains its safety properties with high probability, but not with probability one.

slot r, then all nodes receiving on channel k detect a collision.

We assume nodes have access to standard asymmetric cryptography primitives; i.e., they can generate public/private key pairs and use them to encrypt/decrypt, and sign/verify messages. We assume nodes cannot break the cryptographic system.

Adversary Model. We allow some nodes to suffer Byzantine failure. In particular, a faulty node can: (a) try to create sybil identities; (b) cause collisions (i.e., jam) on a channel by broadcasting noise; and/or (c) try to spoof messages (i.e., pretend to be some other *real* honest nodes). We assume a bound t on the maximum number of faulty nodes. More specifically, the algorithms we consider require that $t \leq \min\{n, c\}/\alpha$, for a constant $\alpha \geq 1$. We assume that honest nodes do *not* know n or t, but they know c and α. The faulty nodes, on the other hand, are aware of all parameters. We also allow the faulty nodes to collude, and adapt to past execution history. Therefore, we sometimes refer to them collectively as a single adaptive malicious entity named *Eve* that can use up to t channels per time slot.

Problem Statement. The goal of our sybil-thwarting algorithm is for each honest node to construct a set of trusted and unforgeable identities that includes: (a) a unique identity for each honest node, and (b) a bounded number of sybil identities. We assume that each honest node generates a unique public/private key pair at the beginning of an execution and uses its public key as its identity. Each node can subsequently sign its messages with its private key, confirming its identity (so that malicious nodes cannot spoof messages on behalf of an honest node). The challenge, therefore, is to minimize the number of sybil identities accepted by honest nodes, while simultaneously trying to minimize the time complexity. In this chapter, we study randomized algorithms and require that their guarantees hold with high probability (w.h.p.).

4.3 The SIMPLESYBILSIEVE Algorithm

We first present the SIMPLESYBILSIEVE algorithm which introduces our core strategies for defeating sybil attacks in this unknown and anonymous setting. Notice, throughout this section, we assume $\alpha \geq 6$; and throughout this chapter, we use $0 < \delta < 1$ to denote a small constant. (For the ease of presentation, sometimes we will assume $\delta = 1/100$.)

One point worth clarifying is, as we shall later see, tweaking δ between two small constants (that are smaller than one) only puts a constant factor change on the running time of the proposed algorithms. (Therefore, assuming $\delta = 1/100$ is only for ease of presentation. The number $1/100$ is not a special or magic number.)

On the other hand, changing α to a smaller value (which implies the algorithm can tolerate more malicious nodes) can sometimes be critical and render the protocol unable to achieve its guarantees. As a result, determining the minimum feasible value of α is an interesting open question that worth exploring in the future.

4.3.1 Protocol Description

The first issue that needs to be resolved when designing the algorithm is the unknown number of nodes, and we estimate it in the usual manner: the protocol proceeds in *epochs*, where in epoch i, we assume there are $n_i = 2^i$ nodes in total. In addition, in epoch i, we use $n_i/2$ channels; if $c < n_i/2$, then we simulate the $n_i/2$ needed channels using $n_i/(2c)$ slots for each "round" of the protocol.

At a high level, nodes have two tasks during each epoch: (a) check whether the current estimation of n is accurate; and (b) if the estimation is (roughly) correct, then use the results of uncoordinated radio resource tests to spot honest identities and eliminate sybil identities. To achieve these goals, in each round, each honest node will broadcast or listen—each with probability $1/2$—on a random channel that is chosen uniformly from $[1, n_i/2]$.

To accomplish the first task, honest nodes count the number of silent rounds in each epoch to determine the accuracy of the current estimation. If the estimate is too small, there is nothing Eve can do to make it looks correct, as the honest nodes alone generate enough contention on each channel to prevent too many silent rounds. Once the estimate is correct, by contrast, Eve cannot make a good estimate looks bad, because she can only jam a limited number of channels simultaneously.

Once an honest node believes the current estimate is correct, it will re-examine the messages it has received during the current epoch to determine which identities to accept. More specifically, an honest node accepts an identity if it has heard that identity sufficiently many times in the current epoch. Since the number of radios Eve has is limited, the number of sybils she can successfully create in one epoch is limited as well.

Pseudocode of SIMPLESYBILSIEVE executed at node u:

Generate asymmetric key pair. Let k_p^u be u's public key, and k_o^u be u's private key.
$ids \leftarrow \emptyset$. ▷ Set containing identities that u accepts.
for (every epoch $i \geq 1$) **do**
  $n_i \leftarrow 2^i$, $count_{listen} \leftarrow 0$, $count_{silent} \leftarrow 0$.
  $ids_{count} \leftarrow \emptyset$. ▷ ids_{count} is a dictionary structure with value being an integer.
  for (every round $1 \leq j \leq an_i \lg n_i$) **do**
    $ch \leftarrow \texttt{random}(n_i/2)$. ▷ $\texttt{random}(x)$ returns a random value from $[1, x]$.
    for (every slot $1 \leq k \leq \max\{1, n_i/2c\}$) **do** ▷ Simulate one round with multiple slots.
      if ($\lfloor (ch-1)/c \rfloor = k-1$) **then**
        if ($\texttt{random}(2) = 1$) **then** ▷ Broadcast with probability $1/2$.
          $\texttt{broadcast}(\langle k_p^u \rangle, ((ch-1) \textbf{ mod } c) + 1)$. ▷ $\texttt{broadcast}(m, h)$ broadcasts m on h.
        else ▷ Listen with probability $1/2$.
          $count_{listen} \leftarrow count_{listen} + 1$.
          $msg \leftarrow \texttt{listen}(((ch-1) \textbf{ mod } c) + 1)$. ▷ $\texttt{listen}(h)$ listens on channel h.
          if ($msg \neq nil$ **and** $msg \neq noise$) **then** ▷ Node has heard a message.
            Let k_p^v be the public key inside msg.
            if ($k_p^v \notin ids_{count}$) **then**
              $ids_{count}[k_p^v] \leftarrow 1$.
            else
              $ids_{count}[k_p^v] \leftarrow ids_{count}[k_p^v] + 1$.
          else if ($msg = nil$) **then** ▷ Node has heard a silent round.
            $count_{silent} \leftarrow count_{silent} + 1$.
  if ($count_{silent}/count_{listen} \geq (1-\delta)(1-1/\alpha)e^{-1/2}$) **then**
    return ids.
  else if ($count_{silent}/count_{listen} \geq (1-\delta)(1-4/\alpha)e^{-2}$) **then**
    for (every identity k_p^v in ids_{count}) **do**
      if ($ids_{count}[k_p^v] \geq a(1-\delta)(1-4/\alpha)\lg n_i/(2e^2)$) **then**
        $ids \leftarrow ids \bigcup \{k_p^v\}$.

Figure 4.1: Pseudocode of SIMPLESYBILSIEVE.

We now describe each honest node's behavior in more detail. In epoch i, there are $an_i \lg n_i$ rounds, where $a > 0$ is some sufficiently large constant. In each round, every node will go to a random channel that is chosen uniformly from $[1, n_i/2]$. Then, each node will broadcast or listen, each with probability $1/2$. If a node chooses to broadcast, it will broadcast its identity (i.e., its public key). If a node chooses to listen, it will record whether it has heard silence (i.e., nothing), noise, or a packet from another identity.

After each epoch, for every node: if at least $\gamma_1 = (1-\delta)(1-1/\alpha)e^{-1/2} = \Theta(1)$ fraction of rounds were silent, among the rounds that it chose to listen, the node will terminate (without accepting any new identities). Otherwise, if at least $\gamma_2 = (1-\delta)(1-4/\alpha)e^{-2} = \Theta(1)$ fraction of slots were silent, where $\gamma_2 < \gamma_1$, then the node accepts every identity from which it has received at least $a\gamma_2(\lg n_i)/2$ messages. This procedure is summarized in Figure 4.1.

4.3.2 Analysis

In this subsection, we prove that every honest node will terminate by epoch $\lfloor \lg n \rfloor + O(1)$ and accept all other honest nodes. We will also show that there are at most $O(t \cdot \max\{1, n/c\})$ sybils that are accepted by honest nodes.

Before presenting the detailed analysis, we will first introduce the following two well-known Chernoff Bounds [37]. They will be constantly used in the proofs.

Theorem 4.1 (Chernoff Bounds [37])**.** *Let $X_1, X_2, \cdots, X_n$ be independent Poisson trials such that $\mathbb{P}(X_i) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}(X)$. Then the following Chernoff Bounds hold:*

$$\textit{for } 0 < \delta \leq 1, \mathbb{P}(X \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/3}; \tag{4.1}$$

$$\textit{for } 0 < \delta < 1, \mathbb{P}(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}. \tag{4.2}$$

We will now present the detailed analysis.

Termination and Correctness. To begin with, we argue that no honest nodes will accept any identities or terminate before (or during) epoch $\lg(n/(g \lg n))$, where g is a positive constant. The claim follows via standard coupon collector analysis as, in these epochs, the number of broadcasting honest nodes exceeds the number of used channels, creating contention and preventing silent rounds.

Lemma 4.2. *For any epoch i where $1 \leq i \leq \lg(n/(g \lg n))$, for some constant $g > 1$: w.h.p. honest nodes will hear only noisy rounds, and hence will not accept any identities during epoch i or terminate at the end of epoch i.*

Proof. Fix an epoch $1 \leq i \leq \lg(n/(g \lg n))$. In expectation, $(n-t)/2 = \Theta(n)$ honest nodes will choose to broadcast in every round in epoch i. Since each honest node decides whether to broadcast or listen independently, apply a Chernoff bound (in particular, Equation 4.2) and we know, in a fixed round in epoch i, w.h.p. at least $(1-\delta)(n-t)/2$ honest nodes will choose to broadcast. Take a union bound over all rounds in epoch i, and another union bound over the $\lg(n/(g \lg n))$ epochs and we know, for every round before (and include) epoch $\lg(n/(g \lg n))$, w.h.p. at least $(1-\delta)(n-t)/2 = \Theta(n)$ honest nodes will choose to broadcast.

On the other hand, according to the protocol, for every round in every epoch before (and include) $\lg(n/(g\lg n))$, at most $n/2g\lg n$ channels will be used or simulated. Consider a fixed round of a fixed epoch i where $1 \leq i \leq \lg(n/(g\lg n))$. Conditioned on the event that at least $(1-\delta)(n-t)/2 = \Theta(n)$ honest nodes choose to broadcast in this round, we know in expectation, each used channel will have at least $g(1-\delta)(n-t)\lg n/n = \Theta(\lg n)$ broadcasting honest nodes. Since honest nodes choose which channel to go to randomly, by a Chernoff bound (in particular, Equation 4.2) and a union bound, we know for sufficiently large g, w.h.p. at least two honest nodes will choose to broadcast on every channel. Take another union bound over all rounds before (and include) epoch $\lg(n/(g\lg n))$, we know w.h.p. at least two honest nodes will choose to broadcast on every channel in all rounds before (and include) epoch $\lg(n/(g\lg n))$. This means w.h.p. all channels that will be used have no silent rounds before (and include) epoch $\lg(n/(g\lg n))$, and hence we have proved the lemma. □

We continue to consider epochs from $\lg(n/(g\lg n))$ to epoch $\lfloor\lg n\rfloor$. For these epochs, we claim that all honest nodes will still not terminate, since they will not hear enough silent rounds.

Lemma 4.3. *For any epoch i where $\lg(n/(g\lg n)) \leq i \leq \lfloor\lg n\rfloor$, w.h.p. no honest nodes will terminate after epoch i.*

Proof. We do an induction on epochs. According to Lemma 4.2, we know w.h.p. no honest nodes will terminate before (and include) epoch $\lg(n/(g\lg n))$. Assume for every epoch $\lg(n/(g\lg n)) \leq i < \lfloor\lg n\rfloor$, w.h.p. all honest nodes decide to continue after epoch i. We now consider epoch $i+1$.

In epoch $i+1$, there are at least $an_{i+1}\lg n_{i+1} = \Omega(n)$ rounds. Since each honest node chooses to listen with probability $1/2$ in every round, we know in expectation, in epoch $i+1$, an honest node will choose to listen in $\Omega(n)$ rounds. Since each round is independent, by a Chernoff bound (in particular, Equation 4.2), we know in epoch $i+1$, an honest node will choose to listen in at least $\Omega(n)$ rounds, w.h.p. Take a union bound over the $O(n)$ honest nodes, we know this claim holds true for them as well. Now, fix an honest node u, we calculate the maximum probability that it hears a silent round in a round that it decides to listen. For this to happen, first, the other $n-t-1$ honest nodes must not broadcast on the channel chosen by u, which happens with probability at most

$(1-(1/2)(2/n_{i+1}))^{n-t-1} \approx e^{-(n-t)/n_{i+1}} \leq e^{-(n-t)/n} \leq e^{-(1-1/\alpha)}$. Secondly, u must not choose a channel that is disrupted by Eve, which happens with probability at most 1. Therefore, in expectation, u will hear at most $e^{-(1-1/\alpha)}$ fraction of silent rounds among the rounds it chose to listen. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.1) and we know, w.h.p. u will hear at most $(1+\delta)e^{-(1-1/\alpha)}$ fraction of silent rounds among the rounds it chose to listen. On the other hand, u will choose to terminate after epoch $i+1$ if it has heard at least $(1-\delta)(1-1/\alpha)e^{-1/2}$ fraction of silent rounds among the rounds it chose to listen. Notice, when $\alpha \geq 5$, for sufficiently small δ (e.g., $1/100$), one can easily verify $(1+\delta)e^{-(1-1/\alpha)} < (1-\delta)(1-1/\alpha)e^{-1/2}$. Hence, w.h.p. u will not terminate after epoch $i+1$. Take a union bound over the $O(n)$ honest nodes and we know w.h.p. no honest nodes will terminate after epoch $i+1$. $\square$

We then show that every honest node will accept all other honest nodes during epoch $\lfloor \lg n \rfloor$, i.e., each honest node will hear from all the other honest nodes sufficiently often in that epoch.

Lemma 4.4. *Every honest node will accept all honest nodes in epoch $\lfloor \lg n \rfloor$, w.h.p.*

Proof. Fix two honest nodes u and v, we calculate the minimum probability that u hears v's identity in a round. For such an event to happen: (a) u must choose to listen, which happen with probability $1/2$; (b) v must choose to broadcast, which happens with probability $1/2$; (c) u and v must choose the same channel, which happens with probability $2/2^{\lfloor \lg n \rfloor}$; (d) the other $n-t-2$ honest nodes do not broadcast on the channel chosen by u, which happens with probability $(1-(1/2)(2/2^{\lfloor \lg n \rfloor}))^{n-t-2} \approx e^{-(n-t)/2^{\lfloor \lg n \rfloor}} > e^{-2(n-t)/n} \geq e^{-2}$; and (e) u does not choose a channel that is disrupted by Eve, which happens with probability at least $1-t/\min\{2^{\lfloor \lg n \rfloor - 1}, c\} > 1-4/\alpha$. Hence, in a round, u will hear v's identity with probability at least $(1/2)\cdot(1/2)\cdot(2/2^{\lfloor \lg n \rfloor})\cdot(1/e^2)\cdot(1-4/\alpha) = (1-4/\alpha)/2e^2 2^{\lfloor \lg n \rfloor}$. Thus, during epoch $\lfloor \lg n \rfloor$, in expectation, u will hear v's identity at least $a(1-4/\alpha)\lfloor \lg n \rfloor / 2e^2$ times. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.2) and we know, w.h.p. u will hear v's identity at least $a(1-\delta)(1-4/\alpha)\lfloor \lg n \rfloor / 2e^2$ times. Take two union bounds (each over $O(n)$ honest nodes) and we know, for any two honest nodes u and v, u will hear v's identity at least $a(1-\delta)(1-4/\alpha)\lfloor \lg n \rfloor / 2e^2$ times, w.h.p.

We then show, in epoch $\lfloor \lg n \rfloor$, every honest node will hear at least $(1-\delta)(1-$

$4/\alpha)e^{-2}$ fraction of silent rounds among the rounds it chose to listen, w.h.p. Fix an honest node u. Since each node chooses to listen with probability $1/2$ in each round, and since each round is independent, by using a Chernoff bound (in particular, Equation 4.1 and 4.2), we know u will choose to listen in $\Theta(n \lg n)$ rounds in epoch $\lfloor \lg n \rfloor$, w.h.p. In each round u chose to listen, it will hear a silent round if: (a) the other $n - t - 1$ honest nodes do not broadcast on the channel chosen by u, which happens with probability $(1 - (1/2)(2/2^{\lfloor \lg n \rfloor}))^{n-t-1} \approx e^{-(n-t)/2^{\lfloor \lg n \rfloor}} > e^{-2(n-t)/n} \geq e^{-2}$; and (b) u does not choose a channel disrupted by Eve, which happens with probability $1 - t/\min\{2^{\lfloor \lg n \rfloor - 1}, c\} > 1 - 4/\alpha$. Hence, in each round u chose to listen, it will hear a silent round with probability at least $(1 - 4/\alpha)/e^2$. Therefore, in expectation, u will hear at least $(1 - 4/\alpha)/e^2$ fraction of silent rounds among the rounds it chose to listen. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.2) and we know, u will hear at least $(1 - \delta)(1 - 4/\alpha)/e^2$ fraction of silent rounds among the rounds it chose to listen, w.h.p. Take a union bound over all honest nodes and we know, this claim holds true for them as well. By now, we have proved the lemma. □

Lastly, we show all honest nodes will terminate at most two epochs later.

Lemma 4.5. *All honest nodes will finish executing* SIMPLESYBILSIEVE *no later than the end of epoch* $\lfloor \lg n \rfloor + 2$*, w.h.p.*

Proof. Assume u is an honest node that is still active during epoch $\lfloor \lg n \rfloor + 2$. Since each node chooses to listen with probability $1/2$ in each round, and since each round is independent, apply a Chernoff bound (in particular, Equation 4.1 and 4.2), we know u will choose to listen in $\Theta(n \lg n)$ rounds in epoch $\lfloor \lg n \rfloor + 2$, w.h.p. Now, in each round u chose to listen, it will hear a silent round if: (a) the other at most $n - t - 1$ honest nodes do not broadcast on the channel chosen by u, which happens with probability $(1 - (1/2)(2/2^{\lfloor \lg n \rfloor + 2}))^{n-t-1} \approx e^{-(n-t)/2^{\lfloor \lg n \rfloor + 2}} > e^{-(n-t)/2n} \geq e^{-1/2}$; and (b) u does not choose a channel that is disrupted by Eve, which happens with probability at least $1 - t/\min\{2^{\lfloor \lg n \rfloor + 1}, c\} \geq 1 - 1/\alpha$. Hence, in each round u chose to listen, it will hear a silent round with probability at least $(1 - 1/\alpha)e^{-1/2}$. Therefore, in expectation, u will hear at least $(1 - 1/\alpha)e^{-1/2}$ fraction of silent rounds among the rounds it chose to listen. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.2) and we know, u will hear at least $(1 - \delta)(1 - 1/\alpha)e^{-1/2}$ fraction of silent rounds

among the rounds it chose to listen, w.h.p. Take a union bound over all the $O(n)$ honest nodes and we know, the above claim holds true for every honest node that is active during epoch $\lfloor \lg n \rfloor + 2$. As a result, according to the protocol, all these honest nodes will terminate after epoch $\lfloor \lg n \rfloor + 2$, w.h.p. □

Constraining Sybil Identities. In this part, we show SIMPLESYBILSIEVE can ensure the total number of sybil identities that are accepted by any honest nodes is at most $O(t \cdot \max\{1, n/c\})$. Firstly, we identify the epochs in which honest nodes can potentially accept sybil identities.

Lemma 4.6. *Honest nodes will accept sybil identities only between epoch* $\lfloor \lg n \rfloor - 1$ *and epoch* $\lfloor \lg n \rfloor + 1$*, w.h.p.*

Proof. According to Lemma 4.2, we known w.h.p. no honest nodes will hear silent rounds during epochs before (and include) $\lg(n/(g \lg n))$, hence they will not accept any sybil identities in these epochs. On the other hand, according to Lemma 4.5, we know w.h.p. all honest will terminate after epoch $\lfloor \lg n \rfloor + 2$, hence they will not accept any sybil identities in epoch $\lfloor \lg n \rfloor + 2$.

For epochs $\lg(n/(g \lg n)) + 1$ to $\lfloor \lg n \rfloor - 2$, according to Lemma 4.3, we know all honest nodes will be active during these epochs, w.h.p. Fix an honest node u, fix epoch i where $\lg(n/(g \lg n)) + 1 \leq i \leq \lfloor \lg n \rfloor - 2$, according to our protocol, in expectation, it will choose to listen in $an_i \lg n_i/2$ rounds. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.2), we know u will choose to listen in at least $a(1-\delta)n_i \lg n_i/2 = \Omega(n)$ rounds in epoch i, w.h.p. Take two union bounds (one over the $O(n)$ honest nodes, and another over the $O(\lg \lg n)$ epochs), we know every honest node will choose to listen for at least $\Omega(n)$ rounds in each of these epochs, w.h.p.

We now consider the maximum probability that an honest node u hears a silent round in a round that it chooses to listen, in an epoch i where $\lg(n/(g \lg n)) + 1 \leq i \leq \lfloor \lg n \rfloor - 2$. For this to happen: (a) the other $n - t - 1$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1 - (1/2)(2/n_i))^{n-t-1} \approx e^{-(n-t)/n_i}$; and (b) u does not choose a channel that is disrupted by Eve, which happens with probability at most 1. Hence, in each round (that u chose to listen) in epoch i, u will hear a silent round with probability at most $e^{-(n-t)/n_i}$. Therefore, in expectation, u will hear at most $e^{-(n-t)/n_i}$ fraction of silent rounds among the rounds it chose to listen.

Since each round is independent, apply a Chernoff bound (in particular, Equation 4.1) and we know, in epoch i, w.h.p. u will hear at most $(1+\delta)e^{-(n-t)/n_i}$ fraction of silent rounds among the rounds it chose to listen. Take a union bound over the $O(n)$ honest nodes and another union bound over the $O(\lg\lg n)$ epochs, we know for any epoch $\lg(n/(g\lg n))+1 \le i \le \lfloor\lg n\rfloor - 2$, honest nodes will hear at most $(1+\delta)e^{-(n-t)/n_i}$ fraction of silent rounds among the rounds they chose to listen, w.h.p.

Notice, for any epoch i where $\lg(n/(g\lg n))+1 \le i \le \lfloor\lg n\rfloor - 2$, we have $(1+\delta)e^{-(n-t)/n_i} \le (1+\delta)e^{-4(n-t)/n}$. Moreover, when $\alpha \ge 6$, for sufficiently small δ (e.g., $1/100$), one can easily verify $(1+\delta)e^{-4(n-t)/n} < (1-\delta)(1-4/\alpha)e^{-2}$. Hence, we know no honest nodes will accept any identities between epoch $\lg(n/(g\lg n))+1$ to $\lfloor\lg n\rfloor - 2$, w.h.p. By now, we have proved the lemma. □

At this point, we can summarize honest nodes' behavior as follows: (a) for every epoch before and including $\lfloor\lg n\rfloor$, no honest nodes will terminate; (b) during epoch $\lfloor\lg n\rfloor - 1$ to epoch $\lfloor\lg n\rfloor + 1$, honest nodes may accept identities, which include other honest nodes and sybil identities; (c) by the end of epoch $\lfloor\lg n\rfloor$, every honest node must have accepted all other honest nodes; and (d) some (or all) honest nodes may terminate after epoch $\lfloor\lg n\rfloor + 1$, and all remaining will terminate after epoch $\lfloor\lg n\rfloor + 2$.

In the following key technical lemma, we show for every epoch i where $\lfloor\lg n\rfloor - 1 \le i \le \lfloor\lg n\rfloor + 1$, at most $O(t\cdot\max\{1, n/c\})$ sybil identities will be accepted by honest nodes. The intuition is: in each round in these epochs, each honest node will, in expectation, broadcast its identity $\Theta(1)$ times. On the other hand, Eve can broadcast an identity at most $O(t\cdot\max\{1,n/c\})$ times in every round, each of which we call a broadcast round-channel combination. This implies Eve can advocate an identity at most $O(t\cdot\max\{1,n/c\})$ times faster than honest nodes. Since honest nodes' broadcast rate allows each of them to be accepted $O(1)$ times (by other honest nodes) in each of these three epochs, we know Eve can successfully create at most $O(t\cdot\max\{1,n/c\})$ sybil identities in each of these three epochs. Notice, one tricky point within the proof is to carefully analyze the (in)dependence relationship between multiple broadcast round-channel combinations, and then apply Chernoff bounds accordingly.

Lemma 4.7. *For epoch i where $\lfloor\lg n\rfloor - 1 \le i \le \lfloor\lg n\rfloor + 1$, at most $\frac{2(1+\delta)e^2\cdot e^{-(n-t)/n_i}}{(1-\delta)(1-4/\alpha)}\cdot t\cdot\max\{1,n_i/2c\} = O(t\cdot\max\{1,n/c\})$ sybil identities will be accepted by honest nodes*

in that epoch, w.h.p.

Proof. To begin with, we introduce the notion of *broadcast round-channel combination*. Such a combination refers to Eve broadcasting on behalf of a sybil identity v in a round on a channel.

We then calculate, for epoch i where $\lfloor \lg n \rfloor - 1 \leq i \leq \lfloor \lg n \rfloor + 1$, the maximum probability that an honest node u receives v's identity in a broadcast round-channel combination, given that Eve is advocating v in that broadcast round-channel combination. For this to happen: (a) u must choose to listen in that round, which happens with probability $1/2$; (b) u and Eve must choose the same channel, which happens with probability $2/n_i$; and (c) the other (at most) $n - t - 1$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1-(1/2)(2/n_i))^{n-t-1} \approx e^{-(n-t)/n_i}$. Hence, in each broadcast round-channel combination that Eve advocates v, u will receive v's identity with probability $(1/2)(2/n_i)e^{-(n-t)/n_i} = e^{-(n-t)/n_i}/n_i$. Therefore, according to our protocol, in epoch i, in expectation, Eve must advocate v for $\frac{a(1-\delta)(1-4/\alpha)\lg n_i/2e^2}{e^{-(n-t)/n_i}/n_i} = \frac{(1-\delta)(1-4/\alpha)}{2e^2 \cdot e^{-(n-t)/n_i}} \cdot a n_i \lg n_i$ broadcast round-channel combinations before u accepts v.

Now consider all the broadcast round-channel combinations in which Eve advocates v in an epoch. Let $X_{j,k}$ be an indicator random variable which denotes whether u successfully receives v's identity when Eve broadcasts on channel k in round j.

For any set $\{X_{j_1,k_1}, X_{j_2,k_2}, \cdots, X_{j_y,k_y}\}$, we claim if $\{j_1, j_2, \cdots, j_y\}$ are different numbers, then it must be the case that $\{X_{j_1,k_1}, X_{j_2,k_2}, \cdots, X_{j_y,k_y}\}$ is a set of independent random variables. (I.e., if Eve advocates v in different rounds each on one channel, then whether u will hear v in a round is independent of whether u will hear v in other rounds.) To see this, consider the three conditions (mentioned in previous paragraph) that are required for Eve to successfully let u hear v in a broadcast round-channel combination. For each of these conditions, between broadcast round-channel combinations in different rounds, whether it will be satisfied are independent. As a result, we can conclude $\{X_{j_1,k_1}, X_{j_2,k_2}, \cdots, X_{j_y,k_y}\}$ is a set of independent random variables.

Meanwhile, for any set $\{X_{j,k_1}, X_{j,k_2}, \cdots, X_{j,k_y}\}$, we claim if $\{k_1, k_2, \cdots, k_y\}$ are different numbers, then $\{X_{j,k_1}, X_{j,k_2}, \cdots, X_{j,k_y}\}$ must be a set of *negatively-dependent* random variables. (I.e., if Eve advocates v on multiple channels in one round, then whether u will hear v on one channel negatively depends on whether u has heard v on

other channels.) According to the definition in [17], we say a set of random variables X_i (where $i \in \{1, 2, \cdots, N\}$) are *negatively-dependent* if for all disjoint subsets $\mathcal{I}, \mathcal{J} \subseteq \{1, 2, \cdots, N\}$ and all non-decreasing functions f and g, the following inequality holds: $\mathbb{E}(f(X_i, i \in \mathcal{I})g(X_j, j \in \mathcal{J})) \leq \mathbb{E}(f(X_i, i \in \mathcal{I}))\mathbb{E}(g(X_j, j \in \mathcal{J}))$. To prove $\{X_{j,k_1}, X_{j,k_2}, \cdots, X_{j,k_y}\}$ is a set of negatively-dependent random variables, assume $f(0, \cdots, 0) = h_1$ and $g(0, \cdots, 0) = h_2$. Fix some disjoint set $\mathcal{I}, \mathcal{J} \subseteq \{1, \cdots, y\}$. Notice, if u has heard v in a broadcast round-channel combination in a round, then u will never hear v again in that round as each honest node can listen on at most one channel in each round. Therefore, we know $\mathbb{E}((f(X_{j,k_l}, l \in \mathcal{I}) - h_1)(g(X_{j,k_m}, m \in \mathcal{J}) - h_2)) = 0 \leq \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I}) - h_1)\mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J}) - h_2)$. According to linearity of expectation, we know $\mathbb{E}((f(X_{j,k_l}, l \in \mathcal{I}) - h_1)(g(X_{j,k_m}, m \in \mathcal{J}) - h_2)) = \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I})g(X_{j,k_m}, m \in \mathcal{J})) - h_1 \cdot \mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J})) - h_2 \cdot \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I})) + h_1 h_2$, and $\mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I}) - h_1)\mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J}) - h_2) = \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I}))\mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J})) - h_1 \cdot \mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J})) - h_2 \cdot \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I})) + h_1 h_2$. Hence, we know $\mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I})g(X_{j,k_m}, m \in \mathcal{J})) \leq \mathbb{E}(f(X_{j,k_l}, l \in \mathcal{I}))\mathbb{E}(g(X_{j,k_m}, m \in \mathcal{J}))$. As a result, we can conclude $\{X_{j,k_1}, X_{j,k_2}, \cdots, X_{j,k_y}\}$ is a set of negatively-dependent random variables.

Since independence is just a special case of negative dependence (according to the definition in [17]), we know any set $\{X_{j_1,k_1}, X_{j_2,k_2}, \cdots, X_{j_y,k_y}\}$ is a set of negatively-dependent random variables. Now, recall we have previously shown that in epoch i, in expectation, Eve must advocate v for at least $\frac{(1-\delta)(1-4/\alpha)}{2e^2 \cdot e^{-(n-t)/n_i}} \cdot an_i \lg n_i$ broadcast round-channel combinations before u accepts v. Apply a Chernoff bound under negative dependence (in particular, Equation 4.1) and we know, in epoch i, w.h.p. Eve must advocate v for at least $\frac{(1-\delta)(1-4/\alpha)}{2(1+\delta)e^2 \cdot e^{-(n-t)/n_i}} \cdot an_i \lg n_i$ broadcast round-channel combinations before u will accept v. Take a union bound over the $O(n)$ honest nodes and we know, w.h.p. Eve must advocate v for at least $\frac{(1-\delta)(1-4/\alpha)}{2(1+\delta)e^2 \cdot e^{-(n-t)/n_i}} \cdot an_i \lg n_i$ broadcast round-channel combinations before any honest nodes will accept v.

Since Eve has at most $t \cdot \max\{1, n_i/2c\} \cdot an_i \lg n_i$ broadcast round-channel combinations in epoch i (recall Eve can broadcast on at most t different channels in each time slot; each round contains $\max\{1, n_i/2c\}$ time slots; and epoch i contains $an_i \lg n_i$ rounds), we know w.h.p. the number of sybil identities that are accepted by honest nodes in epoch i is at most:

$$= \frac{\frac{t \cdot \max\{1, n_i/2c\} \cdot an_i \lg n_i}{\frac{(1-\delta)(1-4/\alpha)}{2(1+\delta)e^2 \cdot e^{-(n-t)/n_i}} \cdot an_i \lg n_i}}{} = \frac{2(1+\delta)e^2 \cdot e^{-(n-t)/n_i}}{(1-\delta)(1-4/\alpha)} \cdot t \cdot \max\{1, n_i/2c\}$$

By now, we have proved the lemma. □

Based on the above six lemmas, we can obtain the following theorem which states the key guarantees provided by SIMPLESYBILSIEVE.

Theorem 4.8. *For any $t \leq \min\{n, c\}/\alpha$ where $\alpha \geq 6$,* SIMPLESYBILSIEVE *finishes within $O(n \lg^2 n \cdot \max\{1, n/c\})$ slots, and guarantees the following, w.h.p.: (a) there are at most $O(t \cdot \max\{1, n/c\})$ sybil identities accepted by the honest nodes, collectively; and (b) every honest node accepts all other honest nodes.*

4.4 The SYBILSIEVE Algorithm

Our ultimate goal is to build on the foundation provided by SIMPLESYBILSIEVE to get an optimal $O(t)$ bound on the number of sybil identities for all values of c (not just when $c > n$). An obstacle, however, is that nodes executing SIMPLESYBILSIEVE do not necessarily terminate at the same time. This makes it difficult to use SIMPLESYBILSIEVE in a more involved anti-sybil strategy. To address this issue, we present an improved version of SIMPLESYBILSIEVE that we call SYBILSIEVE. This new algorithm offers the same sybil bounds as SIMPLESYBILSIEVE, but is slower. In exchange for the extra time, it guarantees all nodes terminate at the same time and share a common constant-factor estimate of n (both of which will prove to be useful for our final algorithm that is going to be presented in the next section).

At the core of SYBILSIEVE is a consensus primitive that is called SYBILSENSUS. Nodes execute SYBILSENSUS at the end of each epoch to decide whether or not to terminate. This ensures nodes stop simultaneously. Nodes then use the epoch number (when they stop) to derive their common estimate of the network size.

For ease of presentation, we define the following notations: (a) let H denote the set of honest nodes; (b) for any honest node u, let ids_u (or just ids when there is no

ambiguity) denote the set of identities it has already accepted; and (c) for any protocol, if all honest nodes start executing it simultaneously, then we say we have a *synchronized execution* of it. We also note here, in the reminder of this chapter, we assume $\alpha \geq 256$.

4.4.1 SYBILSENSUS: A Consensus Building Block

SYBILSENSUS is a wireless variant of the Byzantine consensus algorithm described in [54], with sybil attacks taken into consideration. Before presenting it, we first briefly discuss a broadcast primitive named CONSISTBCST, which is an implementation of the authenticated broadcast (i.e., *Echo Broadcast*) algorithm from [54]. CONSISTBCST is used in SYBILSENSUS.

CONSISTBCST. Consistent broadcast ensures consistency among the messages that are accepted by honest nodes. In particular, it enforces: (a) if an honest node broadcasts a message, then all honest nodes will eventually *accept* that message; (b) if Eve sends a message on behalf of an honest node (i.e., Eve is spoofing), then no honest nodes will accept that message; and (c) if an honest node accepts a message, then all honest nodes will eventually accept that message. When sybil attacks are present, in our context, consistent broadcast should also guarantee: (d) if Eve sends a message on behalf of a sybil identity that is not accepted by any honest nodes, then no honest nodes will accept that message.

CONSISTBCST is an implementation of the Echo Broadcast algorithm described in [54]: initially, the sender broadcasts the message to everyone; everyone who receives the message directly from u, or who receives enough "echo" messages, sends an "echo" message repeating the initial message; finally, everyone who receives enough total copies of the message accepts it. In our context, we make the following small changes to the original protocol: (a) a node only processes messages that are properly signed from other identities that it has already accepted (in an earlier SYBILSIEVE epoch); (b) the protocol is parametrized to operate based on an estimated network size, and the guarantees hold only if this estimate is within a constant factor of being correct; and (c) the guarantees hold only if the number of sybil identities that are accepted by the honest nodes is not too large.

More specifically, each execution of CONSISTBCST contains multiple phases. For

an honest node u, the execution of each phase requires four input parameters: (a) the phase number; (b) estimation of n, denoted as $\hat{n}$; (c) set of identities that are currently accepted by u, denoted as ids_u; and (d) messages to broadcast (if any). The output of CONSISTBCST for an honest node u is a set of messages that are accepted by u.

Each phase of CONSISTBCST consists of two steps, and each step contains $\Theta(\hat{n} \lg \hat{n})$ rounds. (As we shall later see, the existence of multiple rounds is to ensure that messages broadcast by honest nodes can be heard by all other honest nodes.) In the first step, if u has any messages to send, it will broadcast an init tuple for each message. u will also listen and record init tuples that come from other identities in ids_u. In the second step, for every message that is in an init tuple u has heard during the preceding step, and for every message that has appeared in at least $\hat{f} + 1$ different init or echo tuples ("different" meaning each comes from a different identity in ids_u) since the first phase, u will broadcast an echo tuple for that message. Here, $\hat{f} = \lfloor \hat{n}/3 \rfloor - 1$. During the second step, u will also listen and record echo tuples that come from other identities in ids_u. Lastly, after these two steps, for every message for which u has received at least $\hat{n} - \hat{f}$ different init or echo tuples since the first phase, u will accept that message. These descriptions are summarized in Figure 4.2.

In the following lemma, we state the guarantees CONSISTBCST can provide.

Lemma 4.9. *For a synchronized execution of* CONSISTBCST *which contains $l = O(n)$ phases, if all honest nodes start execution with the same $\hat{n}$, then:*

- *If (a) $n/16 \leq \hat{n} \leq n$; and (b) $\forall u \in H$ we have $H \subseteq ids_u$. Then: if an honest node sends a message m at the beginning of phase i, then m will be accepted by all honest nodes after phase i, w.h.p.;*
- *If $|(\bigcup_{u \in H} ids_u) \setminus H| \leq \hat{f}$. Then: if Eve sends a message m on behalf of an honest node, then m will not be accepted by any honest nodes, w.h.p.;*
- *If (a) $n/16 \leq \hat{n} \leq n$; (b) $\forall u \in H$ we have $H \subseteq ids_u$; and (c) $|(\bigcup_{u \in H} ids_u) \setminus H| \leq \hat{f}$. Then: if an honest node accepts a message m by the end of phase $i < l$, then m will be accepted by all honest nodes after phase $i + 1$, w.h.p.;*
- *If $|(\bigcup_{u \in H} ids_u) \setminus H| \leq \hat{f}$, Then: if Eve sends a message m on behalf of a sybil identity that is not accepted by any honest nodes, then m will not be accepted by any honest nodes, w.h.p.*

Pseudocode of phase i of CONSISTBCST executed at node u:

Input: let i be the phase number.
Input: let $\hat{n}$ be the estimation of network size, and $\hat{f} \leftarrow \lfloor \hat{n}/3 \rfloor - 1$.
Input: let ids be the identities that u currently accepts.
Input: let $msgs$ be the messages u wants to send in this phase.

Let k_p^u be u's public key, and k_o^u be u's private key.
if (i=1) **then**
    $msgs_{accpted} \leftarrow \emptyset$. ▷ Set containing messages that u accepts.
    $echo_{count} \leftarrow \emptyset$. ▷ $echo_{count}$ is a dictionary structure with value being a set.
    $echo_{sent} \leftarrow \emptyset$.
$pkt \leftarrow nil$, $echo_{list} \leftarrow \emptyset$. ▷ STEP I.
for (every $msg \in msgs$) **do**
    $tuple \leftarrow \mathsf{init}||msg||i$. ▷ "||" denotes concatenation operation.
    $pkt \leftarrow pkt||\langle k_p^u, \mathtt{sign}(k_o^u, tuple)\rangle$. ▷ $\mathtt{sign}(k, m)$ signs m with key k.
    $echo_{count}[msg] \leftarrow echo_{count}[msg] \bigcup \{k_p^u\}$.
for (every round $1 \leq j \leq a\hat{n} \lg \hat{n}$) **do**
    $ch \leftarrow \mathtt{random}(\hat{n}/2)$.
    for (every slot $1 \leq k \leq \max\{1, \hat{n}/2c\}$) **do**
        if ($\lfloor (ch-1)/c \rfloor = k-1$) **then**
            if ($\mathtt{random}(2) = 1$) **then**
                $\mathtt{broadcast}(pkt, ((ch-1) \bmod c) + 1)$.
            else
                $pkt' \leftarrow \mathtt{listen}(((ch-1) \bmod c) + 1)$.
                if (pkt' comes from an identity v in ids) **then**
                    for (every tuple $\langle \mathsf{init}||msg||i\rangle$ in pkt' such that $msg \notin echo_{sent}$) **do**
                        $echo_{list} \leftarrow echo_{list} \bigcup \{msg\}$.
                        $echo_{count}[msg] \leftarrow echo_{count}[msg] \bigcup \{k_p^v\}$.
$pkt \leftarrow nil$. ▷ STEP II.
for (every entry msg in $echo_{list}$) **do**
    $echo_{sent} \leftarrow echo_{sent} \bigcup \{msg\}$, $pkt \leftarrow pkt||\langle k_p^u, \mathtt{sign}(k_o^u, \mathsf{echo}||msg||i)\rangle$.
for (every entry msg in $echo_{count}$) **do**
    if ($msg \notin echo_{sent}$ **and** $|echo_{count}[msg]| \geq \hat{f} + 1$) **then**
        $echo_{sent} \leftarrow echo_{sent} \bigcup \{msg\}$, $pkt \leftarrow pkt||\langle k_p^u, \mathtt{sign}(k_o^u, \mathsf{echo}||msg||(i-1))\rangle$.
for (every round $1 \leq j \leq a\hat{n} \lg \hat{n}$) **do**
    $ch \leftarrow \mathtt{random}(\hat{n}/2)$.
    for (every slot $1 \leq k \leq \max\{1, \hat{n}/2c\}$) **do**
        if ($\lfloor (ch-1)/c \rfloor = k-1$) **then**
            if ($\mathtt{random}(2) = 1$) **then**
                $\mathtt{broadcast}(pkt, ((ch-1) \bmod c) + 1)$.
            else
                $pkt' \leftarrow \mathtt{listen}(((ch-1) \bmod c) + 1)$.
                if (pkt' comes from an identity v in ids) **then**
                    for (every tuple $\langle \mathsf{echo}||msg||i'\rangle$ in pkt' such that $i' \leq i$) **do**
                        $echo_{count}[msg] \leftarrow echo_{count}[msg] \bigcup \{k_p^v\}$.
for (every $msg \in echo_{count}$) **do** ▷ POST-PROCESSING.
    if ($|echo_{count}[msg]| \geq \hat{n} - \hat{f}$) **then**
        $msgs_{accpted} \leftarrow msgs_{accpted} \bigcup \{msg\}$.
return $msgs_{accpted}$.

Figure 4.2: Pseudocode of one phase of CONSISTBCST.

Proof. We first prove the following claim:

Claim 4.9.1. *In a synchronized execution of* CONSISTBCST*, if honest nodes have the*

same $\hat{n}$ such that $n/16 \leq \hat{n} \leq n$, then during the execution, for every init *or* echo *tuple that comes from an honest node, that tuple will be heard by all honest nodes by the end of the corresponding step, w.h.p.*

Proof. Fix two honest nodes u and v, assume u wants to broadcast a tuple. We calculate the minimum probability that v hears u's tuple in a round. To let this event happen: (a) u must choose to broadcast, which happens with probability $1/2$; (b) v must choose to listen, which happens with probability $1/2$; (c) u and v must choose the same channel, which happens with probability $2/\hat{n} \geq 2/n$; (d) the other at most $n - t - 2$ honest nodes must not broadcast on the channel chosen by u, which happens with probability at least $(1 - (1/2)(2/\hat{n}))^{n-t-2} \approx e^{-(n-t)/\hat{n}} \geq e^{-16(n-t)/n} \geq e^{-16}$; and (e) u does not choose a channel that is disrupted by Eve, which happens with probability $1 - t/\min\{\hat{n}/2, c\} \geq 1 - 32/\alpha$. Hence, in a round, v will hear u's tuple with probability at least $(1/2) \cdot (1/2) \cdot (2/n) \cdot e^{-16} \cdot (1 - 32/\alpha) = (1 - 32/\alpha)/2e^{16}n = \Theta(1/n)$. Since each round is independent, after one step which consists of $\Theta(n \lg n)$ rounds, v will hear u's tuple w.h.p. Take two union bounds each over $O(n)$ honest nodes, and another union bound over the $O(n)$ phases, we can have the claim. □

We now prove the lemma.

We begin with the first property. If an honest node u sends out an init tuple that contains message m at the beginning of phase i, then due to Claim 4.9.1, by the end of step one of phase i, every other honest node must have heard that tuple, w.h.p. Since every honest node accepts u as an honest identity, we know every honest node will send an echo tuple for m during step two of phase i. Again, every honest node's echo tuple that contains m will be heard by all other honest nodes, w.h.p. As a result, by the end of phase i, every honest node must have received at least $n - t \geq \hat{n} - \hat{f}$ different init or echo tuples for m, and hence accept m, w.h.p. (Recall we assume $\hat{n} \leq n$.)

We then consider the second property. On one hand, notice w.h.p. Eve cannot spoof honest nodes' tuples since they are authenticated by private keys. (Even though Eve cannot break the asymmetric cryptography system we used, she can always blind guess honest nodes' private keys and has a small chance of being correct.) On the other hand, notice we assume there are at most $\hat{f}$ sybil identities that are accepted by any honest nodes. Hence, if Eve spoofs m, then w.h.p. there will be at most $\hat{f}$ different init or echo

tuples for m that are accepted by any honest nodes, each from a sybil identity. This is not enough to trigger honest nodes to broadcast echo tuples for m (as that needs $\hat{f}+1$ different init or echo tuples). Moreover, $\hat{f} \leq \hat{n} - \hat{f}$. Hence, w.h.p. m will not be accepted by any honest nodes.

We continue with the third property. Assume message m from some identity v is accepted by honest node u by the end of phase $i < l$, then it must be the case that u have received at least $\hat{n} - \hat{f} = \hat{n} - \lfloor \hat{n}/3 \rfloor + 1 \geq 2\hat{f} + 1$ different init or echo tuples for m by the end of phase i. Among these tuples, w.h.p. at least $\hat{f}+1$ must each come from an honest node. Hence, by the end of phase i, every honest node must have heard at least $\hat{f}+1$ different init or echo tuples for m, each from an honest node, w.h.p. As a result, by the end of phase $i+1$, every honest node must have broadcast an echo tuple for m, w.h.p. Therefore, by the end of phase $i+1$, every honest node must have heard at least $n - t \geq \hat{n} - \hat{f}$ different init or echo tuples for m and hence accepts it, w.h.p.

Lastly, we consider the fourth property. On one hand, notice w.h.p. Eve cannot spoof honest nodes' tuples since they are authenticated by private keys. On the other hand, notice we assume there are at most $\hat{f}$ sybil identities that are accepted by any honest nodes. Hence, if Eve broadcasts a message m on behalf of a sybil identity that is not accepted by any honest nodes, w.h.p. there will be at most $\hat{f}$ different init or echo tuples for m that are accepted by honest nodes, each from a sybil identity. This is not enough to trigger honest nodes to broadcast echo tuples for m (as that needs $\hat{f}+1$ different init or echo tuples). Moreover, $\hat{f} \leq \hat{n} - \hat{f}$. Hence, w.h.p. m will not be accepted by any honest nodes. □

SYBILSENSUS. We now describe the SYBILSENSUS protocol, which solves a variant of consensus in which nodes agree on a set of items (instead of on a single value). Notice, initially, in SYBILSIEVE, we only use SYBILSENSUS to solve traditional (binary) consensus, in which nodes agree on whether or not to terminate; later, in SYBILSIEVEOPT, we will use the more general version of it to let honest nodes agree on a set of identities. It operates under the assumption that we have a reasonable network size estimation, and that there are only a limited number of sybil identities present.

Each execution of SYBILSENSUS at an honest node u requires three parameters: (a) a set of items, denoted as I_u; (b) an estimation of n, denoted as $\hat{n}$; and (c) a set of

identities that is currently accepted by u, denoted as ids_u.

When the protocol terminates, each node outputs a new set S_u. These sets should agree: every node should output the same set. The validity condition has two parts. First, if an item x is in the input set I_u for every honest node u, then x is in the output set. Second, if an item x is *not* in the input set I_u for any honest node u, then x is *not* in the output set.

An execution of SYBILSENSUS consists of $\hat{f}+1$ phases, where $\hat{f} = \lfloor \hat{n}/3 \rfloor - 1$. During the execution of SYBILSENSUS, nodes will use CONSISTBCST to broadcast messages. In phase i, an honest node u will decide whether to broadcast messages, and the contents of the messages, based on the following rules: (a) in phase one, for every item x in I_u, u will broadcast a message that is a concatenation of its identity and x; and (b) in any phase $i \geq 2$, for every item x that has appeared in at least $\hat{f}+i-1$ different accepted messages before the start of phase i, u will broadcast a message which is a concatenation of its identity and x. Meanwhile, during each phase, honest nodes will also record items that have appeared in accepted messages. By the end of these $\hat{f}+1$ phases, for every item x that has appeared in at least $2\hat{f}+1$ different accepted messages, u will add it to S_u. Finally, SYBILSENSUS returns S_u as the return value. The pseudocode of SYBILSENSUS is summarized in Figure 4.3.

In the following lemma, we show SYBILSENSUS leads to consensus.

Lemma 4.10. *If honest nodes perform a synchronized execution of* SYBILSENSUS *with same $\hat{n}$, then* SYBILSENSUS *guarantees: all honest nodes finish executing* SYBILSENSUS *simultaneously, and each outputs a set S containing zero or more items. Moreover, if: (a) $n/16 \leq \hat{n} \leq n$; (b) $|(\bigcup_{u \in H} ids_u) \setminus H| \leq \hat{f} = \lfloor \hat{n}/3 \rfloor - 1$; (c) $\forall u \in H$ we have $H \subseteq ids_u$; and (d) there are at most $O(n)$ differently named items that are initially in some honest nodes' sets I. Then,* SYBILSENSUS *further guarantees:*

- *(Agreement) Every honest node will output the same set S, w.h.p.*
- *(Validity) For any item x that is initially in every honest node's set I, after the execution of* SYBILSENSUS*, $\forall u \in H$ we have $x \in S_u$, w.h.p. Similarly, for any item x that is initially not in any honest nodes' I, after the execution of* SYBILSENSUS*, $\forall u \in H$ we have $x \notin S_u$, w.h.p.*

Proof. If all honest nodes start simultaneously with same $\hat{n}$, then the first sentence of

Pseudocode of SYBILSENSUS executed at node u:

Input: let $\hat{n}$ be the estimation of network size, and $\hat{f} \leftarrow \lfloor \hat{n}/3 \rfloor - 1$.
Input: let ids be the identities that u currently accepts.
Input: let I_u be a set of items.
Let k_p^u be u's public key, and k_o^u be u's private key.
$I_{msgs} \leftarrow \emptyset$. ▷ I_{msgs} is a dictionary structure with value being a set.
for (every phase $1 \leq i \leq \hat{f} + 1$) **do**
    $msgs \leftarrow \emptyset$.
    if ($i = 1$) **then**
        for (every item x in I_u) **do** $msgs \leftarrow msgs \bigcup \{k_p^u || x\}$.
    else
        for (every entry x in I_{msgs} such that $|I_{msgs}[x]| \geq \hat{f} + i - 1$) **do**
            $msgs \leftarrow msgs \bigcup \{k_p^u || x\}$.
    execute CONSISTBCST with parameter i, $\hat{n}$, ids, and $msgs$.
    Let $msgs_{accepted}$ be the return value of CONSISTBCST.
    for (every $k_p^v || x \in msgs_{accepted}$) **do**
        if ($I_{msgs}[x] = nil$) **then**
            $I_{msgs}[x] \leftarrow \{k_p^v\}$.
        else
            $I_{msgs}[x] \leftarrow I_{msgs}[x] \bigcup \{k_p^v\}$.
$S_u \leftarrow \emptyset$.
for (every entry x in I_{msgs}) **do**
    if ($|I_{msgs}[x]| \geq 2\hat{f} + 1$) **then**
        $S_u \leftarrow S_u \bigcup \{x\}$.
return S_u.

Figure 4.3: Pseudocode of SYBILSENSUS.

the lemma obviously follows from protocol description.

We then prove the validity property. Consider an item x. If every honest node's initial I contains x, then according to the protocol and the first property of Lemma 4.9, by the end of phase one, every honest node will accept all $n - t \geq 2\hat{f} + 1$ honest nodes' messages that contain x, w.h.p. Thus, when honest nodes finish execution, w.h.p. they will all include x in S. Take a union bound over all the $O(n)$ items that are initially in every honest node's I, and we have the first part of the validity requirement.

On the other hand, if every honest node's initial I does not contain x, then none of them will send out messages that contain x, w.h.p. This is because, due to the second and fourth property of Lemma 4.9, we know by the end of any phase $i \geq 1$, w.h.p. each honest node will accept at most $\hat{f}$ messages that contain x, each from a sybil identity. This is not enough to trigger honest nodes to broadcast messages that contain x during phase $i+1$, as that requires an honest node to have accepted at least $\hat{f}+(i+1)-1 \geq \hat{f}+1$ different messages that contain x. As a result, we know w.h.p. every honest node will accept less than $2\hat{f}+1$ messages that contain x during the execution. Thus, when honest

nodes finish execution, w.h.p. they will not include x in S. (Notice, we do not need to take a union bound over different x that are created by Eve and are initially not in any honest node's I. In particular, no matter how many of these x Eve has created, as long as she does not know any honest node's private key, which happens w.h.p., these x will not be included in any honest node's output set S.)

We continue to consider the agreement property. Fix an item x that is initially in some honest nodes' I, we show if an honest node u includes x in S_u, then all other honest nodes will include x in their S as well, w.h.p.

Since u includes x in S_u, we know it must have accepted at least $2\hat{f}+1$ messages that contain x by the end of phase $\hat{f}+1$. Let U denote the set of honest nodes that have sent messages that contain x. Due to the second and fourth property of Lemma 4.9, we know $|U| \geq \hat{f}+1$, w.h.p.

If all honest nodes in U initially include x in their I, then according to the first property of Lemma 4.9, by the end of phase one, all honest nodes will accept these $|U|$ honest nodes' messages that contain x, w.h.p. Then, during step one of phase two, according to the protocol, every honest node that does not initially include x in their I will broadcast message that contain x. As a result, we know all honest nodes will accept at least $n-t \geq 2\hat{f}+1$ messages that contain x by the end of phase two, w.h.p. Hence, when honest nodes finish execution, they will all include x in their S, w.h.p.

If some honest nodes in U initially do not contain x in I, then without lose of generality, assume u' is one such node. In such case, due to the second property of Lemma 4.9, w.h.p. u' must have broadcast a message that contains x during step one of some phase j, where $2 \leq j \leq \hat{f}+1$. Moreover, this implies it must have accepted at least $\hat{f}+j-1$ messages that contain x before phase j. Due to property three of Lemma 4.9, we know w.h.p. all honest nodes must have accepted these $\hat{f}+j-1$ messages by the end of phase j as well. Furthermore, since u' broadcasts a message that contains x during step one of phase j, due to property one of Lemma 4.9, we know all honest nodes will accept this message from u' by the end of phase j, w.h.p. As a result, by the end of phase j, each honest node must have accepted at least $\hat{f}+j$ messages that contain x, w.h.p.

Now, if $j=\hat{f}+1$ (i.e., phase j is the last phase), then according to the protocol, all honest nodes will include x in S since they each has accepted at least $2\hat{f}+1$ messages

that contain x. Otherwise, if $1 \leq j \leq \hat{f}$, then in step one of phase $j+1$, all honest nodes that have not broadcast a message that contain x yet will do so. Due to property one of Lemma 4.9, we know each honest node will accept at least $n - t \geq 2\hat{f} + 1$ messages that contain x by the end of phase $j + 1$, and hence include x in its S when it finishes execution, w.h.p.

Take a union bound over all $O(n)$ items, and we have the agreement property. □

In the special cases where network size estimation is not accurate and honest nodes have not accepted each other yet, SYBILSENSUS can still provide "partial" validity for binary consensus. This property is stated in the lemma below and will later be proved to be useful for SYBILSIEVE.

Lemma 4.11. *If: (a) honest nodes perform a synchronized execution of* SYBILSENSUS *with the same $\hat{n}$; (b) $|(\bigcup_{u \in H} ids_u) \setminus H| \leq \hat{f} = \lfloor \hat{n}/3 \rfloor - 1$; and (c) each honest node's initial I is an empty set. Then, all honest nodes will finish executing* SYBILSENSUS *simultaneously, and each outputs an empty set, w.h.p.*

Proof. Since each honest node's initial I is an empty set, we know any item that can potentially be added to S must come from the sybil identities that are created by Eve. Assume x is one such item. We claim no honest nodes will send out messages that contain x, w.h.p. This is because, due to the second and fourth property of Lemma 4.9, we know by the end of any phase $i \geq 1$, w.h.p. each honest node will accept at most $\hat{f}$ messages that contain x, each from a sybil identity. This is not enough to trigger honest nodes to broadcast messages that contain x during phase $i+1$, as that requires an honest node to have accepted at least $\hat{f} + (i + 1) - 1 \geq \hat{f} + 1$ different messages. As a result, we know w.h.p. every honest node will accept less than $2\hat{f} + 1$ messages that contain x during the execution. Thus, when honest nodes finish execution, w.h.p. they will not include x in S. □

We note here, if it is known that there are only a limited number of accepted sybil identities, then SYBILSENSUS can directly be used as a consensus protocol. Otherwise, in an unknown and anonymous environment, nodes need to run a sybil-resistant node discovery protocol (such as SYBILSIEVE) as a preliminary step (in order to limit the number of accepted sybil identities).

Pseudocode of SYBILSIEVE executed at node u:

```
1: n̂ ← −1.                                   ▷ Node's estimation on network size.
2: ids ← ∅.                                  ▷ Set containing identities that u accepts.
3: for (every epoch i ≥ 1) do
4:     n̂ ← 2^i, I_u ← ∅.
5:     execute epoch i of the original SIMPLESYBILSIEVE protocol.
6:     if (SIMPLESYBILSIEVE decides node should terminate after this epoch) then
7:         I_u ← {term}.
8:     execute SYBILSENSUS with parameter I_u, n̂/4, and ids.
9:     if (term ∉ S_u) then
10:        i ← i + 1.
11:    else
12:        return n̂/2 and ids.
```

Figure 4.4: Pseudocode of SYBILSIEVE.

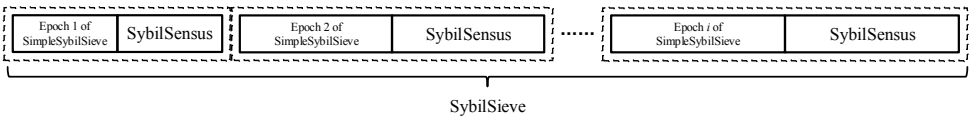

Figure 4.5: High-level structure of SYBILSIEVE.

4.4.2 Maintaining Synchrony

With SYBILSENSUS in hand, we can now proceed to present the SYBILSIEVE protocol. SYBILSIEVE is similar to SIMPLESYBILSIEVE, with the key difference being termination: at the end of each epoch, nodes run SYBILSENSUS to decide whether to terminate or not; the input to SYBILSENSUS depends on whether the termination condition under the SIMPLESYBILSIEVE protocol is met. In particular, an input set contains a singleton item $term$ if and only if the termination condition is met under the SIMPLESYBILSIEVE protocol. A node terminates after the current epoch only if all nodes have agreed to terminate. The pseudocode of SYBILSIEVE is shown in Figure 4.4, and the high-level structure of SYBILSIEVE is illustrated in Figure 4.5.

In the following lemma, we show SYBILSIEVE guarantees all honest nodes will terminate after the same epoch (and hence obtain same estimate on network size). The key intuition behind it is: (a) for every epoch before $\lfloor \lg n \rfloor$, although trusted identities are not fully established, Lemma 4.11 ensures honest nodes will not incorrectly terminate; and (b) starting from epoch $\lfloor \lg n \rfloor$, the conditions which allow SYBILSENSUS to act like a "full" consensus protocol are met, hence all honest nodes will not terminate after epoch $\lfloor \lg n \rfloor$ due to the validity property, and will all terminate simultaneously either after epoch $\lfloor \lg n \rfloor + 1$ or $\lfloor \lg n \rfloor + 2$ due to the agreement (or validity) property.

Lemma 4.12. *All honest nodes will finish executing* SYBILSIEVE *simultaneously, either after epoch* $\lfloor \lg n \rfloor + 1$ *or after epoch* $\lfloor \lg n \rfloor + 2$*, w.h.p.*

Proof. We first consider epochs $i < \lfloor \lg n \rfloor - 1$. Due to Lemma 4.6, we know no honest nodes will accept any identities during these epochs, w.h.p. Moreover, due to Lemma 4.2 and 4.3, we know all honest nodes' initial I will not contain $term$ during these epochs, w.h.p. As a result, we can apply Lemma 4.11 and conclude no honest nodes will decide to terminate in these epochs, w.h.p.

We then consider epoch $i = \lfloor \lg n \rfloor - 1$. Due to Lemma 4.2 and 4.3, we know all honest nodes' initial I will not contain $term$ in this epoch, w.h.p. On the other hand, due to Lemma 4.7, when $\alpha \geq 256$, for sufficiently small δ (e.g., $1/100$), we can conclude that there are less than $\hat{f}_i = \lfloor 2^i/12 \rfloor - 1$ sybil identities that are accepted by any honest nodes, w.h.p. Hence, due to Lemma 4.11, we know no honest nodes will terminate after this epoch, w.h.p.

We continue with epochs $\lfloor \lg n \rfloor$ to $\lfloor \lg n \rfloor + 2$. We first show the requirements to apply Lemma 4.10 are satisfied in these epochs, w.h.p. More specifically, according to protocol description, the synchronized execution and identical estimation requirements are met, so is the requirement that $n/16 \leq \hat{n} \leq n$. Meanwhile, according to Lemma 4.4, requirement (c) of Lemma 4.10 follows, w.h.p. Lastly, according to Lemma 4.7, when $\alpha \geq 256$, for sufficiently small δ (e.g., $1/100$), we can easily verify: (1) by the end of epoch $\lfloor \lg n \rfloor$, there are less than $\hat{f}_{\lfloor \lg n \rfloor}$ sybil identities accepted by honest nodes, w.h.p.; (2) by the end of epoch $\lfloor \lg n \rfloor + 1$, there are less than $\hat{f}_{\lfloor \lg n+1 \rfloor}$ sybil identities accepted by honest nodes, w.h.p.; and (3) by the end of epoch $\lfloor \lg n \rfloor + 2$, there are less than $\hat{f}_{\lfloor \lg n+2 \rfloor}$ sybil identities accepted by honest nodes, w.h.p. Hence, requirement (b) of Lemma 4.10 follows as well, w.h.p.

We now consider honest nodes' behavior during epoch $\lfloor \lg n \rfloor$ to epoch $\lfloor \lg n \rfloor + 2$. In epoch $\lfloor \lg n \rfloor$, due to Lemma 4.3, we know all honest nodes will execute SYBILSENSUS with empty I, w.h.p. Due to the validity property, all honest nodes will continue into epoch $\lfloor \lg n \rfloor + 1$, w.h.p. In epoch $\lfloor \lg n \rfloor + 1$, some (or all) honest nodes may execute SYBILSENSUS with initial I that contains $term$. Due to the agreement property, w.h.p. either all honest nodes decide to terminate, or all honest nodes decide to continue. If all honest nodes decide to continue, then due to Lemma 4.5, in epoch $\lfloor \lg n \rfloor + 2$, all honest nodes will execute SYBILSENSUS with initial I that contains $term$, w.h.p. Due to the

validity property, in such case, all honest nodes will decide to terminate after epoch $\lfloor \lg n \rfloor + 2$, w.h.p. By now, we have proved the lemma. □

Combining Lemma 4.12 and our analysis in Subsection 4.3.2, we immediately have the following theorem which states the key guarantees SYBILSIEVE can provide.

Theorem 4.13. *For any $t \leq \min\{n, c\}/\alpha$ where $\alpha \geq 256$,* SYBILSIEVE *terminates in $O(n^2 \lg^2 n \cdot \max\{1, n/c\})$ slots, and guarantees the following, w.h.p.: (a) there are at most $O(t \cdot \max\{1, n/c\})$ sybil identities accepted by the honest nodes, collectively; (b) every honest node accepts all other honest nodes; and (c) all honest nodes terminate simultaneously and have the same estimate of network size which is either $2^{\lfloor \lg n \rfloor}$ or $2^{\lfloor \lg n \rfloor + 1}$.*

4.5 The SYBILSIEVEOPT Algorithm

In this section we describe SYBILSIEVEOPT, which provides the strongest bounds on the number of sybil identities. This algorithm first runs SYBILSIEVE. It takes advantage of the synchronous termination and network size estimation of SYBILSIEVE to then run a sybil-reduction phase that guarantees no more than $O(t)$ sybil identities. The final time complexity is $O(n^2 \lg^2 n \cdot \max\{1, n/c\})$, which is n times slower than SIMPLESYBILSIEVE (but asymptotically identical to SYBILSIEVE).

In more detail, SYBILSIEVEOPT contains three parts. The first part, as mentioned, executes SYBILSIEVE. The second part executes our consensus algorithm, SYBILSENSUS, so that honest nodes agree on a common set of accepted identities. The third part uses repeated instances of a variant of our previously developed centralized anti-sybil algorithm (i.e., a variant of SYBILCAST), rotating who plays the role of base station, and reduces the number of sybil identities to the (asymptotically) optimal $O(t)$. This structure is illustrate in Figure 4.6. We now describe the second and third part of SYBILSIEVEOPT in more detail, and then conclude with our final theorem statement.

4.5.1 Part Two of SYBILSIEVEOPT: Agree on Set of Accepted Identities

After executing SYBILSIEVE, honest nodes may have accepted different sets of sybil identities. This creates difficulties for the third part of SYBILSIEVEOPT. Hence, honest nodes first run SYBILSENSUS to reach agreement on which identities to accept.

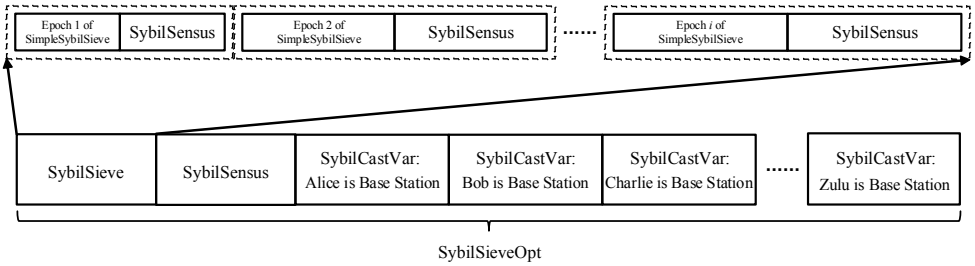

Figure 4.6: High-level structure of SYBILSIEVEOPT.

For the reminder of this section, we will refer to the list of identities that an honest node accepts after the execution of SYBILSIEVE as a *pre-list*; and we refer to the list of identities that an honest node accepts after the execution of the second part of SYBILSIEVEOPT as a *common-list*.

For an honest node u, the second part of SYBILSIEVEOPT is an execution of the SYBILSENSUS algorithm with an input set containing $\Theta(n)$ items: each denotes one identity that is in u's pre-list. (In particular, each item's name is the corresponding identity's public key.)

SYBILSENSUS's input for estimation on network size is $\hat{n}_u/2$, where $\hat{n}_u$—which is either $2^{\lfloor \lg n \rfloor}$ or $2^{\lfloor \lg n \rfloor + 1}$ due to Theorem 4.13—is the estimation on network size obtained by u during the first part of SYBILSIEVEOPT.

The return value of SYBILSENSUS is u's common-list.

The following lemma shows SYBILSENSUS ensures all honest nodes will obtain an identical common-list, which contains all honest nodes, and $O(n)$ sybil identities.

Lemma 4.14. *For the* SYBILSIEVEOPT *algorithm, after the execution of its second part, w.h.p. all honest nodes will obtain an identical common-list which contains all honest nodes and at most* $0.0667n = O(n)$ *sybil identities.*

Proof. To begin with, we show during part two of SYBILSIEVEOPT, the requirements for applying Lemma 4.10 are satisfied. Due to Theorem 4.13 and the protocol description, the synchronized execution and identical estimation requirements are met, as are requirement (c) and the requirement that $n/16 \leq \hat{n} \leq n$. Moreover, due to Lemma 4.7, when $\alpha \geq 256$, for sufficiently small δ (e.g., $1/100$), one can easily verify that by then end of epoch $\lfloor \lg n \rfloor + 2$, there are less than $\hat{f}$ sybil identities that are accepted by any honest nodes, w.h.p. Thus, requirement (b) of Lemma 4.10 follows as well, w.h.p.

Therefore, due to the agreement property of SYBILSENSUS, we know all honest nodes will obtain identical common-list, w.h.p.

We then consider why the common-list contains all honest nodes. Due to Theorem 4.13, we know each honest node's pre-list will contain all honest nodes, w.h.p. Hence, according to the validity property of SYBILSENSUS, we know all honest nodes will appear in the common-list, w.h.p.

Lastly, we consider the maximum number of sybil identities in the common-list. According to Lemma 4.7, when $\alpha \geq 256$ and $\delta = 1/100$, after part one of SYBILSIEVEOPT, there are at most $\max\{2.09t, 0.0021n\}+\max\{5.66t, 0.0111n\}+\max\{9.31t, 0.0364n\} \leq 0.0667n$ sybil identities that are accepted by any honest nodes, w.h.p. Hence, due to the validity property of SYBILSENSUS, we know at most $0.0667n$ sybil identities will appear in the common-list, w.h.p. By now, we have proved the lemma. □

We note here, Lemma 4.14 also yields another important corollary: *by appending* SYBILSENSUS *after* SYBILSIEVE*, honest nodes can reach consensus anonymously with no prior knowledge of other participants, w.h.p.* This effectively solves the Byzantine consensus problem in unknown and anonymous wireless networks, with sybil attacks and other malicious behavior present.

4.5.2 Part Three of SYBILSIEVEOPT: Reduce Number of Sybil Identities

In the last part of the SYBILSIEVEOPT protocol, honest nodes will execute many repetitions of a variant of the SYBILCAST protocol which we refer to as SYBILCASTVAR, and reduce the total number of sybil identities to $O(t)$.

Due to Lemma 4.14, after executing the second part of SYBILSIEVEOPT, all honest nodes will agree on an identical common-list, which includes all honest nodes and $O(n)$ sybil identities. At the beginning of the third part of SYBILSIEVEOPT, honest nodes will sort this common-list according to some pre-defined order (e.g., dictionary order).

After sorting, honest nodes will repeatedly execute SYBILCASTVAR, with identities in the common-list taking turns (in order) playing the role of "base station." (Notice, this is why the honest nodes must accept a common set of identities.) In each repetition, the "base station" verifies the identities and broadcasts a list which contains the identities it believes are honest. When all repetitions are done, honest nodes accept identities that

Pseudocode of the third part of SYBILSIEVEOPT executed at node u:

Let $\hat{n}$ be the estimation on network size. ▷ $\hat{n} \in \{2^{\lfloor \lg n \rfloor}, 2^{\lfloor \lg n \rfloor + 1}\} \subseteq (n/2, 2n]$.
Let ids be the set of identities u accepts after the second part of SYBILSIEVEOPT.
$ids_{sorted} \leftarrow \texttt{sort}(ids)$. ▷ Sort the identities according to some pre-defined rule.
$count \leftarrow \emptyset$, $ids_{final} \leftarrow \emptyset$. ▷ $count$ is a dictionary structure with value being a set.
for (every $k_p^v \in ids$) **do**
    $count[k_p^v] \leftarrow \emptyset$.
for (every repitition $1 \leq i \leq |ids_{sorted}|$) **do**
    $id_{rep} \leftarrow ids_{sorted}[i]$.
    execute SYBILCASTVAR with id_{rep} being the base station.
    Let ids_{rep} be the set of identities that id_{rep} believes to be honest. ▷ ids_{rep} is the rep-list.
    for (every $k_p^v \in ids_{rep}$) **do**
        $count[k_p^v] \leftarrow count[k_p^v] \bigcup \{id_{rep}\}$.
for (every $k_p^v \in ids$) **do**
    if ($|count[k_p^v]| \geq |ids| - \hat{n}/6$) **then**
        $ids_{final} \leftarrow ids_{final} \bigcup \{k_p^v\}$.
return ids_{final}.

Figure 4.7: High-level structure of the third part of SYBILSIEVEOPT.

are considered to be honest in at least a majority of repetitions. This descriptions of the general structure of part three of SYBILSIEVEOPT is summarized in Figure 4.7.

SYBILCASTVAR. We will now introduce the SYBILCASTVAR algorithm. SYBILCASTVAR is a simplified version of the SYBILCAST algorithm we introduced in last chapter. In particular, SYBILCASTVAR does not take data dissemination into consideration. Moreover, it also does not need to address the issue that nodes may dynamically join and leave. Nevertheless, the core ideas for SYBILCAST and SYBILCASTVAR to identify sybil identities are similar: nodes hop among channels according to seeds which are generated by the base station, and collect nonces disseminated by the base station. If there are too many sybil identities in the network, then most of them will lose many nonces and hence will be identified.

More specifically, each execution of SYBILCASTVAR consists of three phases: the dissemination phase, the collection phase, and the verification phase. In the dissemination phase, the "base station" will disseminate different random binary strings, called *seeds*, to every other identity in the common-list, which instructs them as to which channels to listen on. In the collection phase, the base station will broadcast many one-time random binary strings, called *nonces*; other nodes, on the other hand, will hop among channels according to the sequence defined by its seed, and collect nonces. Finally, in the verification phase, nodes will send the nonces they have collected during the pre-

ceding collection phase back to the base station. The base station will add identities that can provide sufficient nonces to its *rep-list*, and then broadcast the rep-list.

We now describe each phase of SYBILCASTVAR in more detail.

Dissemination Phase. As mentioned above, the goal of the dissemination phase is to allow the identity that acts as the base station deliver a unique and random binary string to every other identity in the common-list. We call this string as a seed. Each identity can use its seed as the key for a pseudorandom number generator (PRNG) [33]. The output of this pseudorandom number generator is a channel hopping sequence that the identity should follow during the coming collection phase.

To achieve this goal, the dissemination phase contains multiple time slots. In each time slot, the base station will broadcast the seeds on a randomly selected channel, while each node will listen on a randomly selected channel and try to receive its seed.

There are two points worth noting. Firstly, to ensure each seed is only known to the base station and the intended receiver, it is encrypted by the corresponding receiver's public key. (Otherwise, Eve can eavesdrop and later jam honest nodes.) Secondly, every seed is also authenticated by the base station's private key. (Otherwise, Eve can spoof seeds on behalf of the base station.)

The pseudocode for the dissemination phase is summarized in Figure 4.8.

Collection Phase. The collection phase is used to deliver authentication messages from the base station to other identities. In each slot, the base station randomly chooses a channel and sends a packet on it. Each packet contains a one-time random string, called a nonce. All identities that should listen on that channel (according to their seeds) need to record that nonce. The base station will later use these nonces to verify whether the identities who should have been listening on that channel really were.

The pseudocode for the collection phase is summarized in Figure 4.9.

Verification Phase. The verification phase serves two purposes. Firstly, it allows the base station to detect sybil identities. During the verification phase, each identity will transmit verification messages back to the base station, containing all the nonces it has received during the preceding collection phase. The base station will receive and verify these messages and add identities that can provide sufficient nonces to its rep-

Base station u's pseudocode for the dissemination phase:

Let k_p^u be u's public key, k_o^u be u's private key, and $\hat{n}$ be the estimation on network size.
Let ids be the common-list.
$msg \leftarrow nil$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$.
$seeds \leftarrow \emptyset$. ▷ $seeds$ is a dictionary structure with value being a seed.
for (every $k_p^v \in ids$) **do**
    Generate seed $seed$.
    $seeds[k_p^v] \leftarrow seed$.
    $msg \leftarrow msg || \langle k_p^v, \texttt{encrypt}(k_p^v, seed) \rangle$. ▷ $\texttt{encrypt}(k, m)$ encrypts m with key k.
for (every slot $1 \leq i \leq a\hat{c} \lg \hat{n}$) **do**
    $\texttt{broadcast}(k_p^u || \texttt{sign}(k_o^u, msg), \texttt{random}(\hat{c}))$.

Client node u's pseudocode for the dissemination phase:

Let v denote the identity that should be the base station in this repetition.
Let $\hat{n}$ be the estimation on network size.
$seed \leftarrow nil$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$.
for (every slot $1 \leq i \leq a\hat{c} \lg \hat{n}$) **do**
    $pkt \leftarrow \texttt{listen}(\texttt{random}(\hat{c}))$.
    if (pkt comes from v) **then**
        Extract seed for u as $seed$ from pkt.

Figure 4.8: Dissemination phase of SYBILCASTVAR.

Base station u's pseudocode for the collection phase:

Let k_p^u be u's public key, k_o^u be u's private key, and $\hat{n}$ be the estimation on network size.
Let ids be the common-list.
$nonces \leftarrow \emptyset$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$. ▷ $nonces$ is a dictionary structure with value being a set.
for (every slot $1 \leq i \leq a\hat{c} \lg \hat{n}$) **do**
    $ch \leftarrow \texttt{random}(\hat{c})$.
    Generate nonce as $nonce$.
    for (every $k_p^v \in ids$) **do**
        if ($\texttt{prng}(seeds[k_p^v], i) = ch$) **then** ▷ $\texttt{prng}(k, i)$ gives the i^{th} output of PRNG under key k.
            $nonces[k_p^v] \leftarrow nonces[k_p^v] \bigcup \{nonce\}$.
    $\texttt{broadcast}(k_p^u || \texttt{sign}(k_o^u, nonce), ch)$.

Client node u's pseudocode for the collection phase:

Let v denote the identity that should be the base station in this repetition.
Let $\hat{n}$ be the estimation on network size.
$nonce\_list \leftarrow \emptyset$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$.
for (every slot $1 \leq i \leq a\hat{c} \lg \hat{n}$) **do**
    $ch \leftarrow \texttt{prng}(seed, i)$.
    $pkt \leftarrow \texttt{listen}(ch)$.
    if (pkt comes from v) **then**
        Extract nonce as $nonce$ from pkt.
        $nonce\_list \leftarrow nonce\_list \bigcup \{nonce\}$.

Figure 4.9: Collection phase of SYBILCASTVAR.

list. (The precise meaning for "sufficient" is provided in the pseudocode and the later analysis.) The base station will then broadcast its rep-list to all identities, which is the second purpose of the verification phase.

Base station u's pseudocode for the verification phase:

Let k_p^u be u's public key, k_o^u be u's private key, and $\hat{n}$ be the estimation on network size.
Let ids be the common-list.
$ids_{rep} \leftarrow \{k_p^u\}$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$.
for (every slot $1 \leq i \leq a\hat{n}\lg\hat{n}$) **do**
    $msg \leftarrow \texttt{listen}(\texttt{random}(\hat{c}))$.
    if (msg comes from $k_p^v \in ids$) **then**
        Extract nonces as $nonce\_list$ from msg.
        if ($|nonces[k_p^v] - nonce\_list| \leq a\lg\hat{n}/125$) **then**
            $ids_{rep} \leftarrow ids_{rep} \bigcup \{k_p^v\}$.
for (every slot $1 \leq i \leq a\hat{c}\lg\hat{n}$) **do**
    $\texttt{broadcast}(k_p^u || \texttt{sign}(k_o^u, ids_{rep}), \texttt{random}(\hat{c}))$.

Client node u's pseudocode for the verification phase:

Let v denote the identity that should be the base station in this repetition.
Let $\hat{n}$ be the estimation on network size.
Let $count$ be the dictionary structure defined in Figure 4.7.
$msg \leftarrow k_p^u || \texttt{sign}(k_o^u, \texttt{encrypt}(k_p^v, nonce\_list))$, $\hat{c} \leftarrow \min\{c, \hat{n}\}$.
for (every slot $1 \leq i \leq a\hat{n}\lg\hat{n}$) **do**
    if ($\texttt{random}(\hat{n}) \leq \hat{c}$) **then**
        $\texttt{broadcast}(msg, \texttt{random}(\hat{c}))$.
for (every slot $1 \leq i \leq a\hat{c}\lg\hat{n}$) **do**
    $pkt \leftarrow \texttt{listen}(\texttt{random}(\hat{c}))$.
    if (pkt comes from v) **then**
        Extract identities trusted by v as ids_{rep} from pkt.
for (every $k_p^{v'} \in ids_{rep}$) **do**
    $count[k_p^{v'}] \leftarrow count[k_p^{v'}] \bigcup \{k_p^v\}$.

Figure 4.10: Verification phase of SYBILCASTVAR.

One point worth noting is that all verification messages should be encrypted by the base station's public key, as this stops Eve from eavesdropping nonces.

The pseudocode for the verification phase is summarized in Figure 4.10.

Analysis of Part Three of SYBILSIEVEOPT. We now analyze the third part of the SYBILSIEVEOPT protocol.

To begin with, we show if an honest node is playing the base station, then each honest node can securely and successfully obtain the seed by the end of the corresponding dissemination phase.

Lemma 4.15. *In a repetition where an honest node u is playing the base station, by the end of the corresponding dissemination phase, every other honest node will obtain its seed generated by u, w.h.p. Moreover, all these seeds will not be known by Eve, w.h.p.*

Proof. Since w.h.p. Eve does not know honest nodes' private keys (recall Eve can only

guess honest nodes' private keys), the confidentiality requirement follows.

We then consider the delivery requirement. Fix a repetition in which an honest node u is the base station, fix another honest node v. We calculate the minimum probability that v receives u's packet (and hence obtains its seed) in a slot during the dissemination phase. For this to happen: (a) u and v must have chosen the same channel, which happens with probability $1/\hat{c}$; and (b) u does not choose a channel that is disrupted by Eve, which happens with probability at least $1 - t/\hat{c} = 1 - t/\min\{c, \hat{n}\} > 1 - 2/\alpha$. Hence, in each slot, v will obtain its seed with probability at least $(1 - 2/\alpha)/\hat{c} = \Theta(1/\hat{c})$. Since each slot is independent, after the dissemination phase which has $a\hat{c}\lg\hat{n}$ slots, v will obtain its seed w.h.p. Take a union bound over the $O(n)$ honest nodes, and another union bound over the $O(n)$ repetitions, we can have the lemma. □

Similarly, when an honest node is playing the base station, the verification messages from the honest nodes can be received by the base station; and the base station's rep-list will also be received by all honest nodes.

Lemma 4.16. *In any repetition in which an honest node u is playing the base station, during the verification phase, every honest node's verification message can be received by u without Eve knowing any nonces inside, w.h.p. Moreover, by the end of the verification phase, every honest node will know u's rep-list, w.h.p.*

Proof. Since w.h.p. Eve does not know any honest nodes' private keys, the confidentiality of the nonces within the verification messages are guaranteed.

We now consider the requirement that every honest node will receive the base station's rep-list. Fix a repetition in which an honest node u is the base station, fix another honest node v. We calculate the minimum probability that v receives u's packet (and hence obtains the rep-list) in a slot during the second part of the verification phase. For this to happen: (a) u and v must have chosen the same channel, which happens with probability $1/\hat{c}$; and (b) u does not choose a channel that is disrupted by Eve, which happens with probability at least $1 - t/\hat{c} = 1 - t/\min\{c, \hat{n}\} > 1 - 2/\alpha$. Hence, in each slot, v will obtain the rep-list with probability at least $(1 - 2/\alpha)/\hat{c} = \Theta(1/\hat{c})$. Since each slot is independent, after the second part of the verification phase which has $a\hat{c}\lg\hat{n}$ slots, v will obtain the rep-list w.h.p. Take a union bound over the $O(n)$ honest nodes, and another union bound over the $O(n)$ repetitions, we have proved the claim.

We then consider the delivery requirement of the verification messages. Fix a repetition in which an honest node u is playing the base station, fix another honest node v. We calculate the minimum probability that u receives v's verification message in a slot during the verification phase. For this to happen: (a) v must choose to broadcast, which happens with probability $\hat{c}/\hat{n}$; (b) u and v must have chosen the same channel, which happens with probability $1/\hat{c}$; (c) the other $n-t-2$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1-(\hat{c}/\hat{n})(1/\hat{c}))^{n-t-2} \approx e^{-(n-t)/\hat{n}} > e^{-2(n-t)/n} \geq e^{-2}$; and (d) u does not choose a channel that is disrupted by Eve, which happens with probability at least $1-t/\hat{c} = 1-t/\min\{c,\hat{n}\} > 1-2/\alpha$. Hence, in each slot, u will hear v's verification message with probability at least $(1-2/\alpha)/(e^2\hat{n}) = \Theta(1/\hat{n})$. Since each slot is independent, after $a\hat{n}\lg\hat{n}$ slots, u will hear v's verification messages w.h.p. Take a union bound over the $O(n)$ honest nodes, and another union bound over the $O(n)$ repetitions, we can have the lemma. □

We then consider how many nonces an honest node can lose when another honest node is playing the base station.

Lemma 4.17. *In any repetition in which an honest node u is playing the base station, during the collection phase, any other honest node will lose at most $a\lg\hat{n}/125$ nonces due to adversarial jamming, w.h.p.*

Proof. Fix a repetition in which an honest node u is playing the base station, fix another honest node v.

In a slot during the collection phase, with probability $1/\hat{c}$, u will send a nonce that v should record. Hence, in expectation, v should record $a\lg\hat{n}$ nonces during the collection phase. Since each slot is independent, apply a Chernoff bound (in particular, Equation 4.1) and we know w.h.p. v should record at most $(1+\delta)a\lg\hat{n}$ nonces during the collection phase.

On the other hand, due to Lemma 4.15, we know w.h.p. Eve does not know u's random choices or v's seed. Conditioned on Eve does not know u's random choices or v's seed, for every nonce v should record, the probability that Eve can jam it is at most $t/\hat{c}$. Hence, in expectation, v will lose at most $(t/\hat{c})\cdot(1+\delta)a\lg\hat{n} < 2(1+\delta)a\lg\hat{n}/\alpha$ nonces. Apply a Chernoff bound (in particular, Equation 4.1) and we know, for $\delta =$

$1/100$, w.h.p. v will lose at most $2(1+\delta)^2 a \lg \hat{n}/\alpha < a \lg \hat{n}/125$ nonces.

Take a union bound over the $O(n)$ honest nodes, and another union bound over the $O(n)$ repetitions, we can have the lemma. □

We continue to show: when an honest node is playing the base station, if Eve creates too many sybil identities, then most of them will lose many nonces.

Lemma 4.18. *In any repetition in which an honest node is playing the base station, if there are $s \geq 12t+1$ sybil identities in the common-list, then at least $s - 12t$ of them will each lose at least $a \lg \hat{n}/5$ nonces during the collection phase, w.h.p.*

Proof. We first show in any slot during the collection phase, if there are $s \geq 12t$ sybil identities, then they will spread on at least $4t$ channel with probability at least $1-\delta$ (for $\alpha \geq 256$ and $\delta = 1/100$). To see this, let w be the event that they spread on at most $4t$ channels, we have:

$$\begin{aligned}
\mathbb{P}(w) &\leq \binom{\hat{c}}{4t}\left(\frac{4t}{\hat{c}}\right)^s \leq \left(\frac{e\hat{c}}{4t}\right)^{4t}\left(\frac{4t}{\hat{c}}\right)^{12t} \\
&= e^{4t}\left(\frac{4t}{\hat{c}}\right)^{8t} \leq e^{4t}\left(\frac{1}{32}\right)^{8t} \\
&= \left(\frac{e}{1024}\right)^{4t} \leq \left(\frac{e}{1024}\right)^{4} \ll \delta
\end{aligned}$$

We now prove the lemma.

Fix a repetition in which an honest node u is playing the base station. Fix a set that contains $12t+1$ sybil identities. According to our above claim, in a slot, these sybil identities will spread on at least $4t$ channels with probability at least $1-\delta$. When such an event happens, with probability at least $4t/\hat{c}$, u will send a nonce to some of these sybil identities. Moreover, if u indeed sends a nonce to some of these sybil identities, since Eve can monitor at most t channels simultaneously, we know with probability at least $3/4$, Eve will not be able to record the nonce. Hence, in a slot, Eve will lose a nonce for at least one of these $12t+1$ sybil identities with probability at least $(1-\delta)\cdot(4t/\hat{c})\cdot(3/4) = (1-\delta)3t/\hat{c}$. Since each slot is independent, apply a Chernoff bound (in particular, Equation 4.2) and we know, during the collection phase, these $12t+1$ sybil identities will in total lose at least $(1-\delta)^2 \cdot 3at \lg \hat{n}$ nonces, with probability at least $1 - n^{-\Theta(t)}$. Since each lost nonce affects at least one sybil identity, we know at

least one of these $12t+1$ sybil identities will lose at least $(1-\delta)^2 \cdot 3at \lg \hat{n}/(12t+1) \geq (1-\delta)^2 \cdot 3at \lg \hat{n}/13t = (1-\delta)^2 \cdot 3a \lg \hat{n}/13$ nonces, with probability at least $1-n^{-\Theta(t)}$.

Due to Lemma 4.14, we know w.h.p. there are at most $0.0667n$ sybil identities in the common-list. Hence, we can conclude there are at most $\binom{0.0667n}{12t+1} \leq (0.0667en/(12t+1))^{12t+1} < (en/12t)^{13t} < (en/12)^{13t} = n^{O(t)}$ different sets of $12t+1$ sybil identities. Take a union bound over all these sets and we know, for any set of $12t+1$ sybil identities, w.h.p. at least one of them will lose at least $(1-\delta)^2 \cdot 3a \lg \hat{n}/13 > a \lg \hat{n}/5$ nonces.

Hence, we know if there are $s \geq 12t+1$ sybil identities, then at least $s-12t$ of them will each lose at least $a \lg \hat{n}/5$ nonces, w.h.p. (Otherwise, with probability $1/n^{o(1)}$, we can find a set of $12t+1$ sybil identities each lose less than $a \lg \hat{n}/5$ nonces, which contradicts what we have just shown above.)

Take a union bound over the $O(n)$ repetitions and we can have the lemma. □

Finally, we show SYBILSIEVEOPT can improve the upper bound on the number of sybil identities to $O(t)$. The key intuition behind it is: when honest nodes are executing the last part of SYBILSIEVEOPT, in each repetition in which an honest node is playing the base station, SYBILCASTVAR ensures there are at most $O(t)$ sybil identities in the rep-list. On the other hand, Eve is only playing the base station (via her sybil identities) in at most a small constant fraction of all repetitions. Since honest nodes only accept an identity if that identity has appeared in the rep-list in a large constant fraction of all repetitions, we know in total at most $O(t)$ sybil identities will be accepted by honest nodes.

Theorem 4.19. *For any $t \leq \min\{n, c\}/\alpha$ where $\alpha \geq 256$, SYBILSIEVEOPT terminates within $O(n^2 \lg^2 n \cdot \max\{1, n/c\})$ slots, and guarantees the following, w.h.p.: (a) there are at most $O(t)$ sybil identities accepted by the honest nodes, collectively; (b) every honest node accepts all other honest nodes; and (c) all honest nodes terminate simultaneously and have same estimation on network size which is either $2^{\lfloor \lg n \rfloor}$ or $2^{\lfloor \lg n \rfloor + 1}$.*

Proof. The protocol execution time follows from the protocol description.

Due to Theorem 4.13, we know all honest nodes will have same estimate on network size which is either $2^{\lfloor \lg n \rfloor}$ or $2^{\lfloor \lg n \rfloor + 1}$, w.h.p.

We now show all honest nodes will be accepted. According to Lemma 4.17 and

protocol description, when an honest node is acting as the base station, every (other) honest node will not lose too many nonces to fail the verification, w.h.p. Moreover, according to Lemma 4.16, the verification packets sent by honest nodes will be received by the base station, w.h.p. Hence, in each such repetition, the base station will accept all honest nodes (including itself according to the protocol), and the base station's rep-list will be known by all honest nodes, w.h.p. On the other hand, due to Lemma 4.14, when $\alpha \geq 256$, there are at most $0.0667n$ repetitions in which a sybil identity is acting as the base station, w.h.p. Hence, each honest node will appear in at least $n - t \geq (n - t) + 0.0667n - n/12 \geq |ids_{list}| - \hat{n}/6$ different rep-lists, w.h.p. As a result, according to the protocol, by the end of SYBILSIEVEOPT, every honest node will accept all honest nodes, w.h.p.

Lastly, we prove the upper bound on the number of sybil identities. According to Lemma 4.14, after the second part of SYBILSIEVEOPT, w.h.p. all honest nodes will agree on a list of identities which contains all $n - t$ honest nodes and q sybil identities, where $0 \leq q \leq 0.0667n$. Let $l = (n - t) + q$ denote the length of this list. According to our protocol, to let an honest node accept a sybil identity v by the end of SYBILSIEVEOPT, that honest node must have seen v in at least $l - \hat{n}/6$ different rep-lists. This implies in at least $l - \hat{n}/6 - q = (n - t) - \hat{n}/6$ repetitions in which an honest node is acting as the base station, the corresponding base station accepts v by the end of the corresponding verification phase. Due to Lemma 4.18, we know when an honest node is acting as the base station, it will accept at most $12t$ sybil identities by the end of the corresponding verification phase, w.h.p. Therefore, we know by the end of SYBILSIEVEOPT, there are at most $12t \cdot (n - t)/((n - t) - \hat{n}/6) \leq 12t(n - t)/((n-t)-n/3) = 12t(1+(n/3)/((n-t)-n/3)) = 12t(1+(n/3)/(2n/3-t)) \leq 12t(1 + (n/3)/(2n/3 - n/256)) < 19t$ sybil identities that are accepted by any honest nodes, w.h.p. (As, there are $n-t$ repetitions in which honest nodes play as base station, and in each such repetition, at most $12t$ sybil identities will be in the corresponding rep-list. Since each sybil identities needs to appear in at least $(n-t)-\hat{n}/6$ different rep-lists which are sent by honest nodes, we know the total number of sybil identities that will eventually be accepted by at least one honest node is at most $12t\cdot(n-t)/((n-t)-\hat{n}/6)$.)

By now, we have proved the theorem. □

4.6 Extending SIMPLESYBILSIEVE to Other Model

The current version of SIMPLESYBILSIEVE relies on the availability of a collision detection mechanism (as nodes need to be able to distinguish between collision and silence to accurately count the number of silent rounds). We believe, however, that even without this assumption, our goal can still be achieved. In particular, in this section, we will describe and analyze a protocol named SIMPLESYBILSIEVEWCD (i.e., "SIMPLESYBILSIEVE" without collection detection), which is a variant of SIMPLESYBILSIEVE that does not rely on collision detection. When compared with SIMPLESYBILSIEVE, the SIMPLESYBILSIEVEWCD algorithm can provide the same asymptotic bounds in terms of running time and number of sybil identities accepted. Nevertheless, if we take the constants behind the asymptotic notations into consideration, SIMPLESYBILSIEVEWCD's performance is worse than SIMPLESYBILSIEVE. Towards the end of this section, we will discuss how these differences on constants can affect SIMPLESYBILSIEVEWCD's potential of being integrated with other algorithms (such as combing SIMPLESYBILSIEVEWCD with SYBILSENSUS and SYBILCASTVAR to form a more advanced protocol, just as we did with SIMPLESYBILSIEVE in previous section).

We will first describe the SIMPLESYBILSIEVEWCD algorithm, and then proceed to analysis and discussion. Notice, throughout this section, we assume $\alpha \geq 10$, and assume nodes cannot tell the difference between collision and silence. (Other aspects of the network model remain unchanged.)

4.6.1 Protocol Description

The general structure of SIMPLESYBILSIEVEWCD is identical to SIMPLESYBILSIEVE: the protocol proceeds in epochs, where in epoch i, nodes assume there are $n_i = 2^i$ nodes in total. In addition, in epoch i, nodes use $n_i/2$ channels; and if $c < n_i/2$, then nodes simulate the $n_i/2$ needed channels using $n_i/(2c)$ slots for each "round" of the protocol.

Similar to SIMPLESYBILSIEVE, nodes have two tasks during each epoch: (a) check whether the current estimate of n is accurate; and (b) if the estimate is (roughly) correct, then use the results of uncoordinated radio resource tests to spot honest identities and eliminate sybil identities. To achieve these goals, in each round, each honest node's behavior is to broadcast or listen—each with probability $1/2$—on a random channel

that is chosen uniformly from $[1, n_i/2]$.

The key difference between SIMPLESYBILSIEVE and SIMPLESYBILSIEVEWCD is the mechanism employed by honest nodes to determine the accuracy of the current estimate of n: in SIMPLESYBILSIEVEWCD, nodes rely on the fraction of clear message rounds (i.e., rounds in which they receive a message) they heard to decide whether their guess of n is correct or not. In particular, as we will later show, when the estimate is near n, the fraction of clear message rounds will reach a certain peak value. Notice, the malicious nodes cannot have big impact on this fractional value: when the estimate is too small, there is nothing Eve can do to make it looks correct, as the honest nodes alone generate enough contention on each channel to prevent too many clear message rounds; on the other hand, once the estimate becomes too large, Eve cannot fool the honest nodes as well since the number of channels that will be used is large yet Eve can only broadcast messages on a limited fraction of them simultaneously.

Once the honest nodes believe the current estimate of n is roughly correct, they will apply the same technique that is used in SIMPLESYBILSIEVE to distinguish honest identities and sybil identities: each node will re-examine the messages it has received during the current epoch and only accepts an identity if it has heard that identity sufficiently often in current epoch.

We now describe each honest node's behavior in more detail. In epoch i, there are $an_i \lg n_i$ rounds, where $a > 0$ is some sufficiently large constant. In each round, every node will go to a random channel that is chosen uniformly from $[1, n_i/2]$. Then, each node will broadcast or listen, each with probability $1/2$. If a node chooses to broadcast, it will broadcast its identity (i.e., its public key). If a node chooses to listen, it will record whether it has heard nothing or a packet from another identity. Notice, since collision detection is not available, a silent round may occur when nothing is being transmitted (on that channel) *or* a collision has happened (on that channel).

After each epoch, each node will calculate the fraction of clear message rounds it has heard. When this value reaches $\gamma = 2(1-\delta)(1-4/\alpha)e^{-2}$ for the first time, nodes will start accepting identities (in this and later epochs). Later, when this value falls below γ, nodes will terminate, without accepting any identities in that last epoch. In each epoch in which a node decides to accept identities, an identity will be accepted by that node if it has heard that identity for at least $a(1-\delta)(1-4/\alpha)\lg n_i/(2e^2)$ times.

Pseudocode of SIMPLESYBILSIEVEWCD executed at node u:

Generate asymmetric key pair. Let k_p^u be u's public key, and k_o^u be u's private key.
$ids \leftarrow \emptyset$. ▷ Set containing identities that u accepts.
$flag \leftarrow false$. ▷ Boolean variable indicating whether to accept identities in current epoch.
for (every epoch $i \geq 1$) **do**
    $n_i \leftarrow 2^i$, $count_{listen} \leftarrow 0$, $count_{msg} \leftarrow 0$.
    $ids_{count} \leftarrow \emptyset$. ▷ ids_{count} is a dictionary structure with value being an integer.
    for (every round $1 \leq j \leq an_i \lg n_i$) **do**
        $ch \leftarrow \texttt{random}(n_i/2)$.
        for (every slot $1 \leq k \leq \max\{1, n_i/2c\}$) **do**
            if ($\lfloor (ch-1)/c \rfloor = k-1$) **then**
                if ($\texttt{random}(2) = 1$) **then** ▷ Broadcast with probability $1/2$.
                    $\texttt{broadcast}(\langle k_p^u \rangle, ((ch-1) \textbf{ mod } c) + 1)$.
                else ▷ Listen with probability $1/2$.
                    $count_{listen} \leftarrow count_{listen} + 1$.
                    $msg \leftarrow \texttt{listen}(((ch-1) \textbf{ mod } c) + 1)$.
                    if ($msg \neq nil$) **then** ▷ Node has heard a message.
                        Let k_p^v be the public key inside msg.
                        if ($k_p^v \notin ids_{count}$) **then**
                            $ids_{count}[k_p^v] \leftarrow 1$.
                        else
                            $ids_{count}[k_p^v] \leftarrow ids_{count}[k_p^v] + 1$.
                        $count_{msg} \leftarrow count_{msg} + 1$.
    if ($count_{msg}/count_{listen} < 2(1-\delta)(1-4/\alpha)e^{-2}$ **and** $flag = true$) **then**
        return ids.
    else if ($count_{msg}/count_{listen} \geq 2(1-\delta)(1-4/\alpha)e^{-2}$ **and** $flag = false$) **then**
        $flag \leftarrow true$.
    if ($flag = true$) **then**
        for (every identity k_p^v in ids_{count}) **do**
            if ($ids_{count}[k_p^v] \geq a(1-\delta)(1-4/\alpha)\lg n_i/(2e^2)$) **then**
                $ids \leftarrow ids \bigcup \{k_p^v\}$.

Figure 4.11: Pseudocode of SIMPLESYBILSIEVEWCD.

The above description is summarized in the pseudocode in Figure 4.11.

4.6.2 Analysis

In this subsection, we show SIMPLESYBILSIEVEWCD can provide similar guarantees when compared to SIMPLESYBILSIEVE: all honest nodes will terminate at epoch $\lfloor \lg n \rfloor + O(1)$, accept all other honest identities and at most $O(t \cdot \max\{1, n/c\})$ sybil identities.

Termination and Correctness. To begin with, we argue that no honest nodes will accept any identities or terminate before (or during) epoch $\lg(n/(g \lg n))$, where g is a positive constant. This claim follows from the proof of Lemma 4.2: in these epochs, the number of broadcasting honest nodes exceeds the number of used channels, creating

contention and preventing clear message rounds.

We then consider epochs from $\lg(n/(g \lg n))$ to epoch $\lfloor \lg n \rfloor - 2$. For these epochs, we claim that all honest nodes will still not accept any identities or terminate, since they will not hear enough clear message rounds.

Lemma 4.20. *For any epoch i where $\lg(n/(g \lg n)) \leq i \leq \lfloor \lg n \rfloor - 2$, w.h.p. no honest nodes will accept identities during epoch i or terminate after epoch i.*

Proof. We do an induction on epochs. According to Lemma 4.2, we know w.h.p. no honest nodes will accept any identities or terminate before (and include) epoch $\lg(n/(g \lg n))$. Assume for every epoch $\lg(n/(g \lg n)) \leq i - 1 < \lfloor \lg n \rfloor - 2$, w.h.p. all honest nodes decide not to accept any identities during epoch $i - 1$, and decide to continue after epoch $i - 1$. We now consider epoch i.

In epoch i, there are at least $an_i \lg n_i = \Omega(n)$ rounds. Since each node chooses to listen with probability $1/2$ in every round, and since each round is independent, by using a Chernoff bound (in particular, Equation 4.2) and a union bound, we know every honest node will choose to listen in at least $\Omega(n)$ rounds, w.h.p.

Fix an honest node u, we calculate the maximum probability that it hears a message from another node in a round that it decides to listen. To calculate this probability, we fix another node v, and consider the maximum probability u hears a message from v.

In case v is a malicious node, for u to hear v's message: (a) u and v must choose the same channel, which happens with probability at most $1/\min\{c, n_i/2\}$; (b) the other $n - t - 1$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1 - (1/2) \cdot (2/n_i))^{n-t-1} = (1 - 1/n_i)^{n-t-1} \approx e^{-(n-t)/n_i}$; and (c) the other $t - 1$ malicious nodes must not broadcast on the channel chosen by u, which happens with probability at most one. Since there are at most t malicious nodes, the maximum probability that u hears a message from a malicious node in a round is $t \cdot (1/\min\{c, n_i/2\}) \cdot e^{-(n-t)/n_i} \leq t \cdot (2/n_i) \cdot e^{-(n-t)/n_i} = (2t/n_i) \cdot e^{-(n-t)/n_i}$.

Otherwise, in case v is an honest node, to let u hears v's message: (a) u and v must choose the same channel, which happens with probability $2/n_i$; (b) v must choose to broadcast, which happens with probability $1/2$; (c) the other $n-t-1$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1 - (1/2) \cdot (2/n_i))^{n-t-1} = (1 - 1/n_i)^{n-t-1} \approx e^{-(n-t)/n_i}$; and (d) the channel chosen by u is not

disrupted by malicious nodes, which happens with probability at most one. Since there are $n-t-1$ honest nodes, the maximum probability that u hears a message from an honest node in a round is $(n-t-1)\cdot(2/n_i)\cdot(1/2)\cdot e^{-(n-t)/n_i} \approx ((n-t)/n_i)\cdot e^{-(n-t)/n_i}$.

Therefore, in a round that u chooses to listen, it will hear a message with probability at most $((n+t)/n_i)\cdot e^{-(n-t)/n_i} \leq 4(n+t)/n\cdot e^{-4(n-t)/n} = 4(1+1/\alpha)\cdot e^{-4(1-1/\alpha)}$. Since we have previously shown that u will choose to listen in at least $\Omega(n)$ rounds, by a Chernoff bound (in particular, Equation 4.1), we know the fraction of clear message rounds u will hear is at most $4(1+\delta)(1+1/\alpha)\cdot e^{-4(1-1/\alpha)}$, w.h.p. Take a union bound over all honest nodes, we know this claim holds true for every honest node. Recall the protocol requires at least $2(1-\delta)(1-4/\alpha)e^{-2}$ fraction of clear message rounds to start accepting identities. When $\alpha \geq 10$, for sufficiently small δ (e.g., $1/100$), one can easily verify $4(1+\delta)(1+1/\alpha)\cdot e^{-4(1-1/\alpha)} < 2(1-\delta)(1-4/\alpha)e^{-2}$. As a result, we know in epoch i, no honest nodes will accept identities during this epoch or terminate after this epoch, w.h.p. By now, we have proved the lemma. □

Starting from epoch $\lfloor \lg n \rfloor - 1$, nodes may start to accept identities. In the following lemma, we show every honest node will accept all other honest identities by the end of epoch $\lfloor \lg n \rfloor$. Notice, this lemma, together with Lemma 4.2 and Lemma 4.20, also implies no honest nodes will terminate before the end of epoch $\lfloor \lg n \rfloor$.

Lemma 4.21. *Every honest node will accept all other honest nodes during epoch $\lfloor \lg n \rfloor$, w.h.p.*

Proof. Due to Lemma 4.2 and Lemma 4.20, we know no honest nodes will accept identities or terminate during epoch one to epoch $\lfloor \lg n \rfloor - 2$, w.h.p. Hence, for epoch $\lfloor \lg n \rfloor$, if we show every honest node will hear at least $2(1-\delta)(1-2/\alpha)e^{-2}$ fraction of clear message rounds among the rounds it chooses to listen, we know honest nodes will accept identities during epoch $\lfloor \lg n \rfloor$, and will not terminate after epoch $\lfloor \lg n \rfloor$, w.h.p.

To prove this, firstly, notice every honest node will choose to listen in at least $\Omega(n)$ rounds in epoch $\lfloor \lg n \rfloor$, w.h.p. Then, consider a round in which an honest node u chooses to listen, fix another honest node v, we calculate the minimum probability that u hears v's message in that round. To let this event happen: (a) v must choose to broadcast, which happens with probability $1/2$; (b) u and v must choose the same channel, which happens with probability $1/2^{\lfloor \lg n \rfloor - 1}$; (c) the other at most $n-t-1$

honest nodes must not broadcast on the channel u choose to listen, which happens with probability at least $(1-(1/2)(1/2^{\lfloor \lg n \rfloor -1}))^{n-t-1} \approx e^{-(n-t)/2^{\lfloor \lg n \rfloor}}$; and (d) the channel chosen by u is not disrupted by Eve, which happens with probability at least $1-t/\max\{c, 2^{\lfloor \lg n \rfloor -1}\} \geq 1-4/\alpha$. Since there are $n-t-1$ honest nodes in total, we know in a round that u chooses to listen, it will hear a message from another honest node with probability at least $(n-t-1)\cdot(1/2)\cdot(1/2^{\lfloor \lg n \rfloor -1})\cdot e^{-(n-t)/2^{\lfloor \lg n \rfloor}}\cdot(1-4/\alpha) \approx (1-4/\alpha)((n-t)/2^{\lfloor \lg n \rfloor})e^{-(n-t)/2^{\lfloor \lg n \rfloor}} \geq (1-4/\alpha)\cdot\min\{((n-t)/n)e^{-(n-t)/n}, (2(n-t)/n)e^{-2(n-t)/n}\} \geq (1-4/\alpha)\cdot\min\{(1-1/\alpha)e^{-(1-1/\alpha)}, 2e^{-2}\} \geq (1-4/\alpha)\cdot 2e^{-2}$. Since we have previously shown that w.h.p. u will choose to listen in at least $\Omega(n)$ rounds, apply a Chernoff bound (in particular, Equation 4.2) and we know, u will hear at least $2(1-\delta)(1-4/\alpha)e^{-2}$ fraction of clear message rounds, w.h.p. Take a union bound over all honest nodes, we know this claim holds true for them as well. By now, we have shown that all honest nodes will accept identities during epoch $\lfloor \lg n \rfloor$, and will not terminate before the end of epoch $\lfloor \lg n \rfloor$, w.h.p.

We then prove every honest node will accept all other honest nodes during epoch $\lfloor \lg n \rfloor$. To see this, fix two honest nodes u and v, we calculate the minimum probability that u hears v's identity in a round. To let such an event happen: (a) u must choose to listen, which happen with probability $1/2$; (b) v must choose to broadcast, which happens with probability $1/2$; (c) u and v must choose the same channel, which happens with probability $2/2^{\lfloor \lg n \rfloor}$; (d) the other $n-t-2$ honest nodes must not broadcast on the channel chosen by u, which happens with probability $(1-(1/2)(2/2^{\lfloor \lg n \rfloor}))^{n-t-2} \approx e^{-(n-t)/2^{\lfloor \lg n \rfloor}} > e^{-2(n-t)/n} \geq e^{-2}$; and (e) u must not choose a channel that is disrupted by Eve, which happens with probability at least $1-t/\min\{2^{\lfloor \lg n \rfloor -1}, c\} > 1-4/\alpha$. Hence, in a round, u will hear v's identity with probability at least $(1/2)\cdot(1/2)\cdot(2/2^{\lfloor \lg n \rfloor})\cdot(1/e^2)\cdot(1-4/\alpha) = (1-4/\alpha)/2e^2 2^{\lfloor \lg n \rfloor}$. Thus, during epoch $\lfloor \lg n \rfloor$, in expectation, u will hear v's identity at least $a(1-4/\alpha)\lfloor \lg n \rfloor/2e^2$ times. Since each round is independent, apply a Chernoff bound (in particular, Equation 4.2) and we know, w.h.p. u will hear v's identity at least $a(1-\delta)(1-4/\alpha)\lfloor \lg n \rfloor/2e^2$ times. Take two union bounds (each over $O(n)$ honest nodes) and we know, for any two honest nodes u and v, u will hear v's identity at least $a(1-\delta)(1-4/\alpha)\lfloor \lg n \rfloor/2e^2$ times, and hence accept v as a non-sybil identity, w.h.p. By now, we have proved the lemma. □

Lastly, we show all honest nodes will terminate at most four epochs later.

Lemma 4.22. *All honest nodes will terminate no later than the end of epoch* $\lfloor \lg n \rfloor + 4$*, w.h.p.*

Proof. Since we have shown in Lemma 4.21 that every honest node will accept identities during epoch $\lfloor \lg n \rfloor$, we only need to show that all honest nodes will hear less than $2(1-\delta)(1-4/\alpha)e^{-2}$ fraction of clear messages rounds during epoch $\lfloor \lg n \rfloor + 4$ to prove that all honest nodes will terminate after epoch $\lfloor \lg n \rfloor + 4$.

To prove this, fix an honest node u that is still active during epoch $i = \lfloor \lg n \rfloor + 4$, we calculate the maximum probability that it hears a message from another node in a round that it decides to listen. To calculate this probability, we fix another node v, and consider the maximum probability that u hears a message from v in a round.

In case v is a malicious node, to let u hears v's message: (a) u and v must choose the same channel, which happens with probability at most $1/\min\{c, n_i/2\}$; (b) the other honest nodes must not broadcast on the channel chosen by u, which happens with probability at most one (as there might have no other honest nodes at all); and (c) the other $t-1$ malicious nodes must not broadcast on the channel chosen by u, which happens with probability at most one. Since there are at most t malicious nodes, the maximum probability that u hears a message from a malicious node in a round is $t \cdot (1/\min\{c, n_i/2\}) \leq 2t/n_i$.

Otherwise, in case v is an honest node, to let u hears v's message: (a) u and v must choose the same channel, which happens with probability $2/n_i$; (b) v must choose to broadcast, which happens with probability $1/2$; (c) the other honest nodes must not broadcast on the channel chosen by u, which happens with probability at most one (as there might have no other honest nodes); and (d) the channel chosen by u is not disrupted by malicious nodes, which happens with probability at most one. Since there are at most $n-t-1$ honest nodes (beside u), the maximum probability that u hears a message from an honest node in a round is $(n-t-1) \cdot (2/n_i) \cdot (1/2) \approx (n-t)/n_i$.

Therefore, in a round that u chooses to listen, it will hear a message with probability at most $(n+t)/n_i \leq (n+t)/(8n) \leq (1+1/\alpha)/8$. Since we know w.h.p. u will choose to listen in at least $\Omega(n)$ rounds during epoch $\lfloor \lg n \rfloor + 4$, by a Chernoff bound (in particular, Equation 4.1), we know the fraction of clear message rounds u will hear

is at most $(1+\delta)(1+1/\alpha)/8$, w.h.p. Take a union bound over all honest nodes, we know this claim holds true for every honest node that is still active during epoch $\lfloor \lg n \rfloor + 4$. Notice, when $\alpha \geq 10$, for sufficiently small δ (e.g., $1/100$), one can easily verify $(1+\delta)(1+1/\alpha)/8 < 2(1-\delta)(1-4/\alpha)e^{-2}$. Since honest node needs at least $2(1-\delta)(1-4/\alpha)e^{-2}$ fraction of clear message rounds to continue into epoch $\lfloor \lg n \rfloor + 5$, we know all honest nodes will terminate after epoch $\lfloor \lg n \rfloor + 4$, w.h.p. □

Constraining Sybil Identities. In this part, we show SIMPLESYBILSIEVEWCD can ensure the total number of sybil identities that are accepted by any honest nodes is at most $O(t \cdot \max\{1, n/c\})$, which is (asymptotically) identical to SIMPLESYBILSIEVE.

To prove this, firstly, notice according to Lemma 4.2, Lemma 4.20, Lemma 4.21, and Lemma 4.22, we can summarize honest nodes' behavior as follows: (a) for every epoch before and including $\lfloor \lg n \rfloor$, no honest nodes will terminate; (b) during epoch $\lfloor \lg n \rfloor - 1$ to epoch $\lfloor \lg n \rfloor + 3$, honest nodes may accept identities, which include other honest nodes and sybil identities; (c) by the end of epoch $\lfloor \lg n \rfloor$, every honest node must have accepted all other honest nodes; and (d) starting from epoch $\lfloor \lg n \rfloor + 1$, (some or all) honest nodes may decide to terminate after the corresponding epoch, and by the end of epoch $\lfloor \lg n \rfloor + 4$, all honest nodes must have terminated.

Then, by a proof that is almost identical to the one for Lemma 4.7, we can obtain the following lemma, which states Eve can successfully create at most $O(t \cdot \max\{1, n/c\})$ sybil identities in each epoch from epoch $\lfloor \lg n \rfloor - 1$ to epoch $\lfloor \lg n \rfloor + 3$.

Lemma 4.23. *For epoch i where $\lfloor \lg n \rfloor - 1 \leq i \leq \lfloor \lg n \rfloor + 3$, at most $\frac{2(1+\delta)e^2 \cdot e^{-(n-t)/n_i}}{(1-\delta)(1-4/\alpha)} \cdot t \cdot \max\{1, n_i/2c\} = O(t \cdot \max\{1, n/c\})$ sybil identities will be accepted by honest nodes in that epoch, w.h.p.*

Combine these aforementioned lemmas, we can obtain the following theorem which states the guarantees that are provided by SIMPLESYBILSIEVEWCD.

Theorem 4.24. *For any $t \leq \min\{n, c\}/\alpha$ where $\alpha \geq 10$,* SIMPLESYBILSIEVEWCD *finishes within $O(n \lg^2 n \cdot \max\{1, n/c\})$ slots, and guarantees the following, w.h.p.: (a) there are at most $O(t \cdot \max\{1, n/c\})$ sybil identities accepted by the honest nodes, collectively; and (b) every honest node accepts all other honest nodes.*

4.6.3 Comparison and Discussion

As can be seen from Theorem 4.24, both the SIMPLESYBILSIEVE algorithm and the SIMPLESYBILSIEVEWCD algorithm can guarantee correctness (i.e., all honest nodes are correctly accepted); and the bounds they provide in terms of running time and maximum number of sybil identities (that are incorrectly accepted) are also asymptotically identical. In fact, these two protocols also share other similarities: they do not guarantee honest nodes will accept same sets of identities, or terminate simultaneously. Hence, from this point of view, one may infer SIMPLESYBILSIEVEWCD is *better* than SIMPLESYBILSIEVE, as it does not assume the availability of any collision detection mechanism.

Nevertheless, if we take the constants behind the asymptotic notations into consideration, SIMPLESYBILSIEVEWCD's performance becomes *worse* than SIMPLESYBILSIEVE. For example, SIMPLESYBILSIEVEWCD's restriction on the value of α is stronger. More importantly, in SIMPLESYBILSIEVEWCD, the range of epochs in which honest nodes can accept identities is wider, which in turn leads to the result that more sybil identities can potentially be accepted by honest nodes.

The impacts of these differences (on constants) are not limited to the scenario in which SIMPLESYBILSIEVEWCD is executed alone. More critically, they degrade SIMPLESYBILSIEVEWCD's potential to be integrated into other protocols. For example, one may want to composite SIMPLESYBILSIEVEWCD with SYBILSENSUS and SYBILCASTVAR to form a protocol that is similar to SYBILSIEVEOPT without the need for collision detection. Unfortunately though, several barriers exist to realize this plan: (a) SIMPLESYBILSIEVEWCD allows more sybil identities to be accepted (when compared with SIMPLESYBILSIEVE), which may exceed the threshold SYBILSENSUS can tolerate; and (b) SIMPLESYBILSIEVEWCD has a wider range in terms of epochs in which honest nodes may terminate, which may affect the correctness of SYBILSENSUS as it requires a relatively accurate estimation of n (recall, e.g., Lemma 4.10) to work. To overcome these obstacles, two possible solutions are: (a) modify/improve related protocols (e.g., SYBILSENSUS); and/or (b) further strengthen restrictions on α (i.e., α needs to be larger) when executing SIMPLESYBILSIEVEWCD, so that the range of epochs in which honest nodes may accept sybil identities or terminate is smaller.

Table 4.1: Comparison of the SYBILSIEVE family of protocols.

	Time Complexity	Sybil Bound	Sync. Term. and Same Estimate
SIMPLESYBILSIEVE and SIMPLESYBILSIEVEWCD	$O(n \lg^2 n \cdot \max\{1, n/c\})$	$O(t \cdot \max\{1, n/c\})$	no
SYBILSIEVE	$O(n^2 \lg^2 n \cdot \max\{1, n/c\})$	$O(t \cdot \max\{1, n/c\})$	yes
SYBILSIEVEOPT	$O(n^2 \lg^2 n \cdot \max\{1, n/c\})$	$O(t)$	yes

To sum up, it seems SIMPLESYBILSIEVEWCD is not a perfect *replacement* for SIMPLESYBILSIEVE. Instead, trade-offs exist between them (just like SIMPLESYBILSIEVE, SYBILSIEVE, and SYBILSIEVEOPT). In different scenarios, different algorithm may be the most appropriate one.

4.7 Summary and Discussion

In this chapter, we develop new algorithms that can effectively thwart sybil attacks in multi-channel ad hoc wireless networks. They allows wireless devices to securely and reliably establish identities in unknown and anonymous environment. These protocols can also serve as building blocks for other protocols. A comparison of these protocols are shown in Table 4.1.

One interesting point worth noting is, although our algorithm can only achieve consensus in unknown and anonymous networks *with high probability*, we suspect it is impossible to solve this problem with probability one. For deterministic algorithms, in a recent paper by Delporte-Gallet et al. [10], the authors show that synchronous Byzantine agreement is unsolvable if the number of distinct identifiers is less than $3t$, where t is the number of Byzantine nodes. Since it is hard for deterministic algorithms to break symmetry in our setting, this seems to imply consensus is impossible, here, for deterministic algorithms. As for randomized algorithms, with certain small probability, nodes may end up with same coins and hence fail to generate sufficient identities. In fact, under our model, even if nodes indeed generate unique identities, the fact that Eve can potentially keep jamming one honest node (with some small probability) and isolate it from the remaining honest nodes also seems to imply that consensus cannot always be guaranteed. We will discuss more details regarding these potential impossibility results in the next chapter.

Chapter 5

Some Impossibility Results in Noisy Wireless Networks

In previous chapters, we have developed several algorithms that can effectively thwart sybil attacks, jamming attacks, and spoofing attacks in many settings. Nevertheless, all of them suffer one drawback: these algorithms can only enforce their guarantees *with high probability*. This implies, with certain small but non-zero probability, the adversary can still successfully bypass these protocols and conduct various malicious behaviors thereafter. As a result, one natural and important question to ask is: is such chance of error artificial (due to imperfect design and/or analysis), or simply inherently inevitable (due to the nature of the problem).

To answer this question, in this chapter, we will focus on two problems—*the neighbor discovery problem* and *the counting problem*—and explore whether algorithms exist that can *guarantee* to solve them when malicious users are present. Consider a wireless network in which there are multiple users, each of which does not know the existence of others. In the counting problem, the goal is to let each user know *how many* users are there in total in the network. To solve the more difficult neighbor discovery problem, however, each user must know the *identities* of the other users.

Both of these two problems are important and fundamental. For example, the neighbor discovery process is a common bootstrap procedure for many advanced protocols, such as voting, leader election, and fair resource allocation. Knowing the size of the system (i.e., solving the counting problem), on the other hand, may help users determine key parameters that are crucial to the correctness and/or efficiency of the protocols that

will be executed. For instance, our SYBILSIEVE family of protocols rely on relatively accurate estimate of system size to work correctly.

These two problems are also closely related to the malicious behaviors we mentioned above. In particular, by creating sybil identities, an adversary can easily prevent honest users from knowing the correct system size. Similarly, by jamming messages, an adversary can prevent honest users from sending out identities. In this chapter, we will discuss only the impact of jamming on neighbor discovery and counting.

Results. We consider a single hop wireless network consisting of $c \geq 1$ channels. We divide time into synchronous slots and assume that $n \geq 2$ users (called *nodes* in the following) are activated simultaneously. We assume standard collision rules: if multiple nodes send on the same channel during the same slot, a collision occurs on that channel. However, in order to extend the range of scenarios our results can apply, several different collision detection/recovery models are considered. We assume each node has a pseudorandom number generator. Aside from these n honest nodes, we assume there also exist $t \geq 1$ malicious nodes which may collude. Again, we have considered different capabilities the adversary may posses (from the perspective of generating interference) to maximize the applicability of our results.[1]

We first show the fact that the counting problem is no harder than the neighbor discovery problem. This allows any impossibility results that are proved with respect to the counting problem to be applicable to the neighbor discovery problem as well.

We then present our impossibility results. In particular, our first impossibility result shows that when a jamming adversary is present, in most cases, no algorithm can guarantee solving the counting problem (and hence the neighbor discovery problem) within a finite period of time. Then, based on this finding, we further show that in many cases, no Las Vegas algorithm exists that can guarantee solving the counting problem (and hence the neighbor discovery problem) within a finite expected running time.

5.1 Model and Problem Statement

In this section, we will describe the model, and the problem we are interested in. As mentioned earlier, for flexibility, when introducing some aspects of the model, we will

[1] In fact, as we shall see in later sections, many other aspects of the model are flexible as well.

discuss several different possible settings. Nevertheless, later when we are claiming lemmas or theorems, we will clarify which of these settings we are referring to.

We consider a single hop wireless network which contains $c \geq 1$ channels (i.e., the network may be single-channel or multi-channel). Time is divided into synchronous *slots*. In the network, there are n honest nodes, each of which has a unique identity. However, we assume these honest nodes do not know the value of n. We assume honest nodes join the network and start execution simultaneously.

Each honest node has one radio transceiver, each radio transceiver can access all c channels. However, at any given time, a radio transceiver can only operate (i.e., broadcast or listen) on one channel. If no transceivers broadcast on a channel, then all nodes listing on that channel will hear silence. If only one transceiver broadcasts a message on a channel, then all nodes listing on that channel will hear that message. If two or more transceivers broadcast simultaneously on a channel, there is a collision. Depending on the collision detection model, honest nodes listing on that channel may hear a collision (which is distinguishable from silence and single message transmission), or just silence, or one of the message that is being transferred (which may be picked by an adversary or just at random).

Aside from honest nodes, there are $t \geq 0$ malicious nodes. Each of them also has one radio transceiver. We assume malicious nodes can collude with each other, hence we sometimes refer to them collectively as one adversary that has t radios, and we name her as Eve. Eve may be *oblivious* or *adaptive*. In particular, if she is oblivious, then at any time point, she does not know honest nodes' past protocol execution history. Otherwise, if Eve is adaptive, she is allowed to know honest nodes' past protocol execution history. Notice, when proving lower bounds or impossibility results, allowing Eve to be oblivious (instead of adaptive) makes the claim stronger.

Each malicious node's radio can also operate on only one channel at any given time. However, these radio's broadcasting capabilities may be different from the honest nodes' ones. In particular, for a "jamming radio", it can only broadcast noise that looks like a collision, and cannot transmit ordinary messages. (I.e., even if honest nodes can detect collisions, they cannot tell if a collision originates from jamming or a real collision.) On the other hand, for an "overwrite radio", whenever it broadcasts a message on a channel, it "overwrites" all messages that are currently being transferred

on that channel. I.e., nodes that are listening on that channel at that time will only receive the malicious node's message, even if there are other messages from honest nodes being transferred. Last but not least, a malicious node's radio may also just be an ordinary radio that is identical to honest nodes' ones.

We assume each honest node has a pseudorandom number generator that can generate pseudorandom bits. Each node's generator is "private" to itself, which means other nodes and Eve do not know the output of this generator, unless the node itself explicitly reveal them. Nevertheless, when Eve is adaptive, she is allowed to know the bits that are generated in the past.

For simplicity, in later discussion, we will use $\mathcal{N}(c, t, cd, eve, radio)$ to specify the model we are considering. In particular: (a) c denotes the number of channels; (b) t denotes the number of malicious nodes; (c) cd denotes the collision detection model, which can be $\sqrt{}$—honest nodes can detect collision, $\emptyset$—honest nodes will only detect silence, or $?$—honest nodes will accept a message specified by Eve; (d) eve denotes the type of the adversary, which can be `ob`—Eve is an oblivious adversary, or `ad`—Eve is an adaptive adversary; (e) $radio$ denotes Eve's radios' broadcasting capabilities, which can be `jam`—jamming radio, `ow`—overwrite radio, or `common`—ordinary radio which is identical to the honest nodes' ones. Notice, when a parameter is specified by $*$, it means we do not restrict that aspect of the model (i.e., it can be any of the potential settings). Moreover, for a parameter, we may also use notations like "$\cup, \cap, \setminus$" to express "or, and, not" of different settings.[2]

Problem Statement. We are interested in two problems: the *counting* problem, and the *neighbor discovery* problem. To solve the counting problem, each honest node needs to know the exact value of n. On the other hand, to solve the (harder) node discovery problem, each honest node needs to know the identities of all other honest nodes.

For a distributed algorithm $\mathcal{A}$ (that is executed at each honest node), we say it *guarantees* that it solves the counting problem if: when all honest nodes finish executing $\mathcal{A}$ (as they may terminate at different time), each honest node *always* correctly outputs the value of n (i.e., there is no chance of error). Similarly, we say a distributed algorithm $\mathcal{A}$

[2]For example, $\mathcal{N}(c \geq 1, t = 1, cd = * \setminus \{?\}, eve = \{\texttt{ad}\}, radio = \{\texttt{jam}\})$ specifies a network which contains at least one channel. The collision detection model can either be collision is detectable or collision is not detectable. (However, whenever a collision happens, all information is lost and no message will be delivered.) The adversary is adaptive; she controls one radio that is capable of jamming.

guarantees that it solves the neighbor discovery problem if: when all honest nodes finish executing $\mathcal{A}$, each honest node always correctly outputs all honest nodes' identities.

Notice, an algorithm $\mathcal{A}$ can either be *deterministic* or *randomized*. We say $\mathcal{A}$ is deterministic if nodes do not need to invoke their pseudorandom number generators before or while executing $\mathcal{A}$. Otherwise, the algorithm is randomized.

Lastly, we assume Eve knows the algorithm that is executed by the honest nodes.

5.2 Impossibility Results

In this section, we will present our impossibility results, showing that in most cases, no efficient algorithms exist that can guarantee solving the counting problem or the neighbor discovery problem.

To begin with, we show the fact that the neighbor problem is no easier than the counting problem; this allows impossibility results that are proven with respect to the counting problem to be applicable to the neighbor discovery problem as well.

Lemma 5.1. *For any algorithm $\mathcal{A}$, if it can solve the neighbor discovery problem in time T with probability p, then there must exist another algorithm $\mathcal{A}'$ which can solve the counting problem in time T with probability p, where $T \geq 0$ and $0 \leq p \leq 1$.*

Proof. According to the definition of the neighbor discovery problem, if an algorithm $\mathcal{A}$ can solve the neighbor discovery problem in time T with probability p, then by the end of time slot T, each honest node must have known all honest nodes' identities with probability p. Therefore, one can construct an algorithm $\mathcal{A}'$, which first runs $\mathcal{A}$; and by the end of time slot T, by counting the number of identities, honest nodes running $\mathcal{A}'$ can correctly know the total number of honest nodes that are in the network, with probability p. □

We then show our first impossibility result, stating that in most cases, if a jamming adversary is present, then no algorithm exists that can guarantee solving the counting problem within a finite time period. According to Lemma 5.1, this also implies no algorithm exists than can guarantee solving the neighbor discovery problem within a finite time period. The key observation which leads to the proof is: with some non-zero probability, Eve can successfully isolate some nodes by blocking communication

between these nodes and the remaining nodes. This effectively "blinds" these honest nodes and makes them unable to count correctly.

Theorem 5.2. *Under the* $\mathcal{N}(c \geq 1, t \geq 1, cd = \{\surd, \emptyset\}, eve = *, radio = \{\texttt{jam}, \texttt{ow}\})$ *model, for any given finite time period* $T > 0$*, no algorithm exists that can finish within* T *and guarantee solving the counting problem (or the neighbor discovery problem).*

Proof. We prove our claim by contradiction: assume algorithm $\mathcal{A}$ can guarantee solving the counting problem within a finite time period T. Consider two scenarios, in one scenario there are n nodes in total; and in another scenario, there are an additional n^2 nodes (and hence $n + n^2$ nodes in total). We call these two scenarios $\mathcal{E}$ and $\mathcal{E}'$, respectively. We focus on t honest nodes that appear in both scenarios.

Firstly, we consider scenario $\mathcal{E}$.

If algorithm $\mathcal{A}$ is deterministic, before the protocol execution, Eve can correctly predict the t honest nodes' behavior within the given period of time. Therefore, for each of these t honest nodes, she can assign one radio. Moreover, during protocol execution, assume Eve uses the strategy that whenever one of these t honest nodes listens on a channel, she lets the corresponding radio (controlled by her) broadcast noise (or a message that overwrites other messages). As a result, during protocol execution, the t honest nodes will not be able to receive any information from other honest nodes.

If algorithm $\mathcal{A}$ is randomized, notice once honest nodes' random bits have been generated, the corresponding algorithm becomes a deterministic one (for this execution). Moreover, notice algorithm $\mathcal{A}$'s execution time is bounded by a finite value T. Hence, before the protocol execution, Eve can always try to guess the outcomes of the pseudorandom number generators, and with certain non-zero probability (which can be extremely small), she will correctly guess all outcomes. In such case, she can use the same strategy as discussed in the previous paragraph to stop the t honest nodes from receiving information from other honest nodes.

By now, we can conclude, for scenario $\mathcal{E}$, during protocol execution, with some non-zero probability, Eve can stop the t honest nodes from receiving information from other honest nodes. Assume this (unfortunate) event indeed happens. In such case, since algorithm $\mathcal{A}$ grantees solving the counting problem, we know all honest nodes will still get count n.

We then consider scenario $\mathcal{E}'$.

By an analysis that is identical to the above, we know that during protocol execution, with some non-zero probability, Eve can stop the t honest nodes from receiving information from other honest nodes. Assume this (unfortunate) event indeed happens. In such case, $\mathcal{E}$ and $\mathcal{E}'$ looks identical to these t honest nodes. As a result, by the end of $\mathcal{E}'$, they must have gotten n as the count, just like what they did in $\mathcal{E}$. However, this count is incorrect, as there are in fact $n + n^2$ honest nodes in total.

As a result, we have proved the claim by contradiction. □

Next, we present our second impossibility result, which concerns Las Vegas algorithms. Las Vegas algorithms are randomized algorithms which guarantee correctness. However, the running time of the algorithm may vary, depending on the input and the random choices the algorithm has made.

In the following theorem, we prove that no Las Vegas algorithm exists that can guarantee solving the counting problem (and hence the neighbor discovery problem) within a finite expected running time, given that a jamming adversary is present. The core of the proof is showing that any algorithm with a finite expected running time will inevitably results in chance of error.

Theorem 5.3. *Under the* $\mathcal{N}(c \geq 1, t \geq 1, cd = \{\surd, \emptyset\}, eve = *, radio = \{\texttt{jam}, \texttt{ow}\})$ *model, for any Las Vegas algorithm* $\mathcal{A}$*, if its expected running time* T *is a finite number, then it cannot guarantee solving the counting problem (or the neighbor discovery problem).*

Proof. Consider the case in which there are two users (i.e., $n = 2$): Alice and Bob.

For ease of presentation, we divide each time slot into two *steps*: the *communication step* and the *processing step*. In the first step, nodes can send messages (on a particular channel) if they choose to broadcast; otherwise, nodes can listen (on a particular channel) if they choose to listen. Notice, nodes can also do nothing during the first step. We assume by the end of the first step, all messages that can be successfully delivered in this slot (i.e., sender and receiver are on the same channel and there is no interference) are received by the correspond receivers. In the second step, nodes do local computing, such as process received messages, generate random bits. By the end of the second step, each node has decided what action to take during the next time slot.

Consider a randomized algorithm $\mathcal{A}$ which can guarantee solving the counting problem (i.e., $\mathcal{A}$ is a Las Vegas algorithm). We model all possible executions of algorithm $\mathcal{A}$ (on all nodes as a whole) as a forest, which consists of one or multiple trees. In particular, for each tree, each level represents a time slot, and time goes from top level to bottom level. Each root node (of a tree) is at level one. For any level i, a node (of a tree) at that level represents the communication step in time slot i. (I.e., for each level i node of a tree, it represents one possible execution—according to the random choices wireless nodes have made—of the communication step of time slot i.) The link connecting a node in level i and a node in level $i+1$ represents the processing step in time slot i. (I.e., for each edge connecting a level i node of a tree and a level $i+1$ node of a tree, it represents one possible execution—according to the random choices wireless nodes have made—of the processing step of time slot i.) Notice, since algorithm $\mathcal{A}$ utilizes randomization and may require initial processing before sending out any messages in time slot one, the forest may contain multiple (but finite) trees. Similarly, due to randomization, each node in the tree may have multiple (but finite) children.

Notice, for the sake of simplicity, the processing step of the last time slot (during protocol execution) is omitted in the forest (so that the leaf nodes of the trees do not have "tail" links that link to nothing).

In the reminder of this proof, to distinguish a node that is executing the protocol and a node in the tree, we call the prior (i.e., a node that is executing the protocol) a user.

We now prove the claim by contradiction. Without lose of generality, assume randomized algorithm $\mathcal{A}$ can solve the problem with a finite expected running time. Let forest $\mathcal{F}$ denote all possible executions of $\mathcal{A}$. Due to the definition of expectation, we know there must exist a tree $\mathcal{T}$ in $\mathcal{F}$ which contains a finite length path $\mathcal{P}$ from $\mathcal{T}$'s root to one of $\mathcal{T}$'s leaf node. (To see this, notice the expected running time of $\mathcal{A}$ can be expressed as $\sum_i (p_i \cdot |\mathcal{P}_i|)$, where $\mathcal{P}_i$ is a path in a tree in the forest, and p_i is the probability that $\mathcal{P}_i$ is the actual execution. If every $|\mathcal{P}_i|$ is infinite, then without lose of generality, assume $|\hat{\mathcal{P}}|$ is the smallest among them. We know $\sum_i (p_i \cdot |\mathcal{P}_i|) \geq \sum_i (p_i \cdot |\hat{\mathcal{P}}|) \geq |\hat{\mathcal{P}}| \cdot \sum_i p_i = |\hat{\mathcal{P}}|$. I.e., if every $|\mathcal{P}_i|$ is infinite, then the expected running time of $\mathcal{A}$ will be infinite too. Hence, if the expected running time of $\mathcal{A}$ is finite, there must exist some finite length path in the forest.)

We argue the last node on $\mathcal{P}$ can be removed, and the resulting new algorithm $\mathcal{A}'$

can still solve the problem with a finite expected running time. To see this, assume the contrary: if the last node on $\mathcal{P}$ is removed, then in that particular execution, Alice (and/or Bob) will incorrectly terminate (after $|\mathcal{P}| - 1$ time slots since the beginning of protocol execution) with an incorrect count, or Alice (and/or Bob) will never terminate.

If it is the case that removing the last node on $\mathcal{P}$ will result in some user incorrectly terminating without knowing the correct count, then without lose of generality, assume Bob is one victim. In such a scenario, it must be the case that in the last time slot, Alice has sent a message to Bob, and Bob can only correctly terminate upon successfully receiving this message. Notice, however, that Alice cannot tell if this last message has been successfully delivered or not—as Eve can potentially jam or overwrite this message with non-zero probability—and will always terminate after sending this message. Hence, for algorithm $\mathcal{A}$, chances exist that Bob may incorrectly terminate. This contradicts with the assumption that $\mathcal{A}$ can always correctly solve the problem. Thus, the assumption that the last node on $\mathcal{P}$ cannot be removed is incorrect.

On the other hand, if it is the case that removing the last node on $\mathcal{P}$ will result in some user never terminating, then without lose of generality, assume Bob is one victim. In such scenario, again, it must be the case that in the last time slot, Alice has sent a message to Bob, and Bob can only correctly terminate upon successfully receiving this message. By an argument that is identical to the one described in the previous paragraph, we know in this case, the assumption that the last node on $\mathcal{P}$ cannot be removed is incorrect as well.

At this point, we have shown the last node on $\mathcal{P}$ can be removed, and the resulting new algorithm $\mathcal{A}'$ can still solve the problem with a finite expected running time. By a similar argument, we can further show that $\mathcal{A}''$, which is obtained from $\mathcal{A}$ by removing the last two nodes on $\mathcal{P}$, will solve the problem with finite expected running time as well. Since $|\mathcal{P}|$ is finite, in the end, we can show that the execution which has no nodes at all will solve the problem as well. However, this is clearly not true, as Alice and Bob must each send at least one message.

By now, we have proved our claim by contradiction. □

5.3 Summary and Discussion

In this chapter, we have briefly explored some impossibility results a jamming adversary can bring in. In particular, we have shown that in many cases, when a jamming adversary is present, no algorithm can guarantee solving the counting problem or the neighbor discovery problem within a finite time period. Moreover, the same claim holds true for Las Vegas algorithms (with a finite expected running time) as well.

These impossibility results delineate the border of what an algorithm can achieve when solving counting or neighbor discovery in a noisy wireless network. They also suggest our algorithms which are proposed in previous chapters are near optimal in terms of correctness.

Chapter 6

Conclusion

Nowadays, wireless telecommunication is being widely used in people's daily life, and in the foreseeable future, there is no doubt this trend will continue. As a result, providing security guarantees for various wireless networks is of great importance.

In this thesis, we consider three security threats that may exist in wireless networks: jamming attacks, spoofing attacks, and sybil attacks. While examining related work, we find that existing solutions suffer several drawbacks, which is especially true in the case of sybil attacks.

Motivated by this finding, we propose several randomized algorithms that can effectively thwart sybil attacks in multi-channel wireless networks. The core of these algorithms is a simply strategy called radio resource testing, yet as we have seen, generalizing it to practical scenarios is not easy. The proposed algorithms can work in both centralized and ad hoc wireless networks, and can guarantee correctness with high probability. These algorithms can usually provide asymptotically optimal bound on the number of sybil identities. Nevertheless, for some of them, trade-offs do exist between sybil bound and time complexity (e.g., SIMPLESYBILSIEVE and SYBILSIEVEOPT). These algorithms can also tolerate jamming attacks and spoofing attacks.

To the best of our knowledge, these algorithms are the first ones that can effectively thwart sybil attacks in multi-channel wireless networks with provable time and correctness guarantees that are expressed with respect to the actual number of users (instead of the number of identities, which may include a lot of sybils). Another notable contribution of this thesis is several useful subroutines which may be of independent interest (e.g., network size estimation, consensus in sybil-prone environment).

We have also briefly explored how jamming may affect the solvability of the counting problem and the neighbor discovery problem. Unfortunately, the results indicate that even when a simple jamming adversary is present, no algorithm exists that can guarantee to solve these two problems within a finite (expected) running time.

In terms of future research direction, we believe several avenues are available.

For the sybil resistance algorithms, notice, most existing proposals (including the ones proposed in this thesis) only work in single-hop networks. Therefore, a natural and immediate next step is to develop anti-sybil algorithms that can work in multi-hop wireless networks. However, we should note that the multi-hop topology can bring issues that one would not meet in the single-hop scenario. For example, in some areas of a multi-hop wireless networks, the malicious nodes may take a dominate portion. Hence, how can identities be correctly identified (as honest or sybil), and how can such identification outcomes be reliably (e.g., not changed by the malicious nodes) disseminated to the other parts of the network, is a particularly challenging problem.

On the other hand, following the impossibility results which we have shown in Chapter 5, one immediate and interesting next step is to develop lower bounds—for solving the counting problem and the neighbor discovery problem—in terms of time complexity which are expressed with respect to certain correctness probability. Another related direction worth exploring is the impact of sybil attacks regarding these two problem; e.g., how accurate can the solution be when sybil attacks are present, and what would be the minimum time consumption to get such accuracy.

It would also be interesting to see how the proposed algorithms actually perform in practical settings. In particular, in theoretical analysis, we usually ignore the constants behind the asymptotic notation. In real world, unfortunately, these hidden constants can be too large, and may have serious negative impact on the performance, especially when the system size is small. On the other hand, however, these constants—which are derived from theoretical analysis—can deviate from the real figure, due to, e.g., imperfections in the analysis process. As a result, implementing the various protocols in this thesis can be rewarding for both evaluation purposes and further theoretical analysis.

Last but not least, as we have previously discussed in Chapter 2, the impact of new hardware to our protocols is also an interesting future work direction. For example, software defined radios are usually capable of scanning a wide frequency band at the

same time, potentially violating the key assumption that the adversary cannot monitor all channels simultaneously, which is critical to the effectiveness of radio resource testing and the correctness of our algorithms. On the other hand, however, the good news is that the "scanning" power of software defined radios is highly dependent on the processing power of the embedded CPU, and more and more frequency bands are being released for public use. Notice, fundamentally, the ultimate goal of defense is to let the attacker gains (significantly) less than what he or she pays during the process of an attack. Therefore, even if new hardware can favor adversary, it would still be valuable to find and develop "cost-effective" defense strategies.

Bibliography

[1] IEEE standard for information technology–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, 2009.

[2] Bluetooth specification version 4.1. December 2013.

[3] C. Adams and S. Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Longman Publishing, 2nd edition, 2002.

[4] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged byzantine impostors. 2007.

[5] B. Awerbuch, A. Richa, and C. Scheideler. A jamming-resistant mac protocol for single-hop wireless networks. In *Proceedings of the 27th ACM symposium on Principles of distributed computing*, pages 45–54. ACM, 2008.

[6] A. Back. Hashcash - a denial of service counter-measure. Technical report, 2002.

[7] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa. On the performance of ieee 802.11 under jamming. In *INFOCOM 2008*, pages 1265–1273, 2008.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. `http://datatracker.ietf.org/doc/rfc5280`, 2008. RFC5280.

[9] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[10] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, A.-M. Kermarrec, E. Ruppert, and H. Tran-The. Byzantine agreement with homonyms. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 21–30, New York, 2011. ACM.

[11] M. Demirbas and Y. Song. An rssi-based scheme for sybil attack detection in wireless sensor networks. In *Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks*, pages 564–570. IEEE Computer Society, 2006.

[12] S. Dolev, S. Gilbert, R. Guerraoui, D. R. Kowalski, C. Newport, F. Kohn, and N. Lynch. Reliable distributed computing on unreliable radio channels. In *Proceedings of the 2009 MobiHoc S3 workshop on MobiHoc S3*, pages 1–4. ACM, 2009.

[13] S. Dolev, S. Gilbert, R. Guerraoui, F. Kuhn, and C. Newport. The wireless synchronization problem. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 190–199. ACM, 2009.

[14] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Gossiping in a multi-channel radio network. In A. Pelc, editor, *Distributed Computing*, volume 4731 of *Lecture Notes in Computer Science*, pages 208–222. Springer Berlin / Heidelberg, 2007.

[15] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Secure communication over radio channels. In *Proceedings of the 27th ACM symposium on Principles of distributed computing*, pages 105–114. ACM, 2008.

[16] J. R. Douceur. The sybil attack. In *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg, 2002.

[17] D. P. Dubhash and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[18] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147. Springer-Verlag, 1993.

[19] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Artech Print on Demand, second edition edition, 2007.

[20] S. Gilbert, R. Guerraoui, D. Kowalski, and C. Newport. Interference-resilient information exchange. In *IEEE INFOCOM 2009*, pages 2249–2257, 2009.

[21] S. Gilbert, R. Guerraoui, and C. Newport. Of malicious motes and suspicious sensors. *Theor. Comput. Sci.*, 410(6-7):546–569, 2009.

[22] S. Gilbert, V. King, S. Pettie, E. Porat, J. Saia, and M. Young. (near) optimal resource-competitive broadcast with jamming. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 257–266. ACM, 2014.

[23] S. Gilbert, J. Saia, V. King, and M. Young. Resource-competitive analysis: A new perspective on attack-resistant distributed computing. In *Proceedings of the 8th International Workshop on Foundations of Mobile Computing*. ACM, 2012.

[24] S. Gilbert and M. Young. Making evildoers pay: resource-competitive broadcast in sensor networks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 145–154. ACM, 2012.

[25] Z. Golebiewski, M. Klonowski, M. Koza, and M. Kutylowski. Towards fair leader election in wireless networks. In *Ad-Hoc, Mobile and Wireless Networks*, volume 5793 of *Lecture Notes in Computer Science*, pages 166–179. Springer Berlin Heidelberg, 2009.

[26] R. Goodwins. Next-generation wireless networks: from gigabit WiFi to white space, 2013.

[27] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. *SIGCOMM Comput. Commun. Rev.*, 37(4):385–396, 2007.

[28] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42:1:1–1:31, 2009.

[29] A. Juels and J. Brainard. *Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks*, volume 99, pages 151–165. 1999.

[30] V. King, J. Saia, and M. Young. Conflict on a communication channel. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 277–286. ACM, 2011.

[31] M. Klonowski, M. Koza, and M. Kutyowski. Repelling sybil-type attacks in wireless ad hoc systems. In *Information Security and Privacy*, volume 6168 of *Lecture Notes in Computer Science*, pages 391–402. Springer Berlin Heidelberg, 2010.

[32] J. Komlos and A. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Transactions on Information Theory*, 31(2):302–306, 1985.

[33] M. G. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[34] K. Ma, Y. Zhang, and W. Trappe. Mobile network management and robust spatial retreats via network dynamics. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 2005.

[35] D. J. Malan, M. Welsh, and M. D. Smith. Implementing public-key infrastructure for sensor networks. *ACM Trans. Sen. Netw.*, 4(4):22:1–22:23, 2008.

[36] D. Meier, Y. A. Pignolet, S. Schmid, and R. Wattenhofer. Speed dating despite jammers. In *Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems*, pages 1–14. Springer-Verlag, 2009.

[37] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[38] D. Mónica. Thwarting the sybil attack in wireless ad hoc networks. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2009.

[39] D. Mónica, J. Leitão, L. Rodrigues, and C. Ribeiro. On the use of radio resource tests in wireless ad-hoc networks. In *Proceedings of the 3rd Workshop on Recent Advances on Intrusion-Tolerant Systems*, pages F21–F26, 2009.

[40] D. Mónica, J. Leitão, L. Rodrigues, and C. Ribeiro. Observable non-sybil quorums construction in one-hop wireless ad hoc networks. In *Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2010.

[41] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. Using channel hopping to increase 802.11 resilience to jamming attacks. In *26th IEEE International Conference on Computer Communications*, pages 2526–2530, 2007.

[42] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 259–268. ACM, 2004.

[43] G. Noubir and G. Lin. Low-power dos attacks in data wireless lans and countermeasures. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):29–30, 2003.

[44] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176. ACM, 2006.

[45] C. Piro, C. Shields, and B. Levine. Detecting the sybil attack in mobile ad hoc networks. In *Securecomm and Workshops*, pages 1–11, 2006.

[46] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. A jamming-resistant mac protocol for multi-hop wireless networks. In *Proceedings of the 24th international conference on Distributed computing*, pages 179–193. Springer-Verlag, 2010.

[47] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. Competitive and fair medium access despite reactive jamming. In *31st International Conference on Distributed Computing Systems*, pages 507–516, 2011.

[48] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996.

[49] L. G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, 1975.

[50] F. R. Schreiber. *Sybil*. Henry Regnery, 1973.

[51] R. Shirey. Internet security glossary (version 2). `http://datatracker.ietf.org/doc/rfc4949`, 2007. RFC4949.

[52] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, 1994.

[53] R. Smith. *Authentication: From Passwords to Public Keys*. Addison-Wesley, 2002.

[54] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

[55] W. Stallings. *Cryptography and Network Security: Principles and Practice (5th Edition)*. Pearson Education, 2011.

[56] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the 1988 Winter USENIX Conference*, pages 191–202, 1988.

[57] M. Strasser, C. Pöpper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 64–78, 2008.

[58] The WhiteSpace Alliance. `http://www.whitespacealliance.org/InTheNews.html`, 2015.

[59] N. Tran, J. Li, L. Subramanian, and S. Chow. Optimal sybil-resilient node admission control. In *INFOCOM 2011*, pages 3218–3226, 2011.

[60] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 15–28. USENIX Association, 2009.

[61] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications*, pages 324–328, 2005.

[62] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64. ACM, 2004.

[63] A. Wood, J. Stankovic, and S. Son. Jam: a jammed-area mapping service for sensor networks. In *24th IEEE Real-Time Systems Symposium*, pages 286–297, 2003.

[64] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47, 2006.

[65] W. Xu, W. Trappe, and Y. Zhang. Channel surfing: Defending wireless sensor networks from interference. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 499–508. ACM, 2007.

[66] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57. ACM, 2005.

[67] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 80–89. ACM, 2004.

[68] H. Yu. Sybil defenses via social networks: a tutorial and survey. *SIGACT News*, 42(3):80–101, 2011.

[69] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.

[70] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: Defending against sybil attacks via social networks. In *Proceedings of the 2006 Conference*

on Applications, Technologies, Architectures, and Protocols for Computer Communications, pages 267–278. ACM, 2006.

[71] P. R. Zimmermann. *The official PGP user's guide*. MIT Press, 1995.