INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Thwarting The Sybil Attack in Wireless Ad Hoc Networks

Diogo Mónica

Dissertação para obtenção do Grau de Mestre em
Engenharia de Redes de Comunicação

Júri

Presidente:	Prof. Doutor Rui Jorge Morais Tomaz Valadas
Orientador:	Prof. Doutor Carlos Nuno da Cruz Ribeiro
Co-Orientador:	Prof. Doutor Luís Eduardo Teixeira Rodrigues
Vogais:	Prof. Doutor Nuno Fuentecilla Maia Ferreira Neves

July 2009

Agradecimentos

From the formative stages of this thesis, to the final draft, I owe an immense debt of gratitude to both my advisers, Carlos Ribeiro and Luís Rodrigues. Their sound advices and careful guidance were invaluable during the course of the whole work.

I would also like to thank João Leitão for his patience, dedication, and for the endless jokes that kept me going.

This work was partially supported by FCT under grants PTDC/EIA/65588/2006 and PTDC/EIA/71752/2006, and by LEMe in the context of project WiMesh.

Lisboa, July 2009

Diogo Mónica

For my family, who offered me
unconditional love and support
throughout the course of this thesis.
Specially for my father, whose
untiring encouragement was crucial
during this last year.

Resumo

As redes ad hoc sem fios são uma tecnologia que permite a instalação rápida, fácil e económica de redes de comunicação. Infelizmente, estas vantagens também fazem com que sejam mais passíveis de serem atacadas, uma vez que permitem a instalação fácil de nós maliciosos no ambiente. De forma a tornar as redes ad hoc seguras, são tipicamente utilizadas técnicas de tolerância a faltas bizantinas, que normalmente dependem de protocolos de segurança baseadas na utilização de quorums. Uma forma de ataque que permite facilmente disromper a operação da rede consiste na utilização de multiplas identidades por um único adversário, um comportamento denominado como Ataque Sybil. Esta dissertação aborda o problema do Ataque Sybil em redes ad hoc sem fios. Em particular, é proposto um algoritmo que permite que os nós correctos numa dada vizinhança rádio conheçam um conjunto comum de identidades não Sybil. Este algoritmo é baseado na combinação de vários tipos de testes aos recursos, desenvolvidos a partir de uma análise comparada do trabalho anterior nesta área.

Abstract

Wireless ad hoc networking is a technology that allows fast, easy, and inexpensive network deployment. Unfortunately, these advantages also make the task of an attacker simpler, as it is also becomes easier to deploy a malicious node in the environment. To make the ad hoc network secure, one has often to rely on Byzantine fault-tolerance techniques, which typically rely on quorum based security protocols. However, quorums may be easily defeated if a single adversary can participate in the network with multiple identities, a behavior known as the Sybil Attack. This thesis addresses the problem of preventing the Sybil Attack in wireless ad hoc networks. In particular, it is proposed an algorithm that allows the correct nodes in an one-hop neighborhood to have a common set of non-Sybil identities. This algorithm is based on the combination of several types of resource tests, which were developed from a comparative analysis of the previous work in the literature.

Palavras Chave
Keywords

Palavras Chave

Redes sem Fios

Ataque de Sybil

Redes sem fios Ad hoc

Testes de Recursos

Testes de Recurso Rádio

Testes de Recurso Computational

Keywords

Wireless Networks

Sybil Attack

Wireless Ad hoc networks

Resource Tests

Radio Resource Tests

Computational Resource Tests

Index

List of Figures

List of Tables

Acronyms

RRT Radio Resource Test

DoS Denial-of-Service

CRT Computational Resource Test

TDMA Time Division Multiple Access

SST Simultaneous Sender Test

oSST Optimized Simultaneous Sender Test

SRT Simultaneous Receiver Test

FCT Forced Collision Test

MANET Mobile Ad hoc Network

1 Introduction

Wireless ad hoc networking is a technology that allows for fast, easy and inexpensive network deployment. Ad hoc networks are substantially different from infrastructured wireless networks, where nodes never communicate directly amongst themselves and all communication is performed via specialized nodes known as Access Points. Despite their known limitations in terms of scalability and overall capacity (Gupta & Kumar 2000; Li et al. 2001), the decentralized nature, minimal configuration, and self-healing abilities of wireless ad hoc networks, make them suitable for a variety of situations like search and rescue operations, recovery from natural disasters, or military conflicts.

Unfortunately, these advantages also make the task of an attacker simpler, as it is also easier to deploy a malicious node in these environments. To start with, the legitimate nodes of an ad hoc network are typically more vulnerable to tampering than the nodes of a fixed wired network. Also, the membership and topology of a wireless ad hoc network can be very dynamic, making it easier for a malicious node to be inserted in the system. Thus, to make ad hoc networks secure, one has often to rely on Byzantine fault-tolerance techniques.

Generally, most Byzantine fault-tolerance technics rely on some form of quorum (Malkhi & Reiter 1998). In the context of wireless ad hoc networks, Byzantine quorum systems have been used for multiple purposes, including auto-configuration of IP addresses (Xu & Wu 2007), node location (Haas & b. Liang 1999), power saving protocols (Jiang, Tseng, Hsu, & Lai 2005), mobility management (Haas & b. Liang 1999) and reliable storage (Luo, pierre Hubaux, & Eugster 2003).

However, quorums may easily be defeated if a single adversary can participate in the network with multiple identities, a behavior known as the Sybil Attack (Douceur 2002). Therefore, finding efficient techniques to defeat the Sybil Attack is fundamental to build secure wireless ad hoc networks. This is the problem addressed in this work.

1.1 Goals

The primary goal of this thesis is to design a protocol that allows the mitigation of the sybil attack in an one-hop wireless ad hoc network, in a scalable way.

This thesis aims at analyzing and designing efficient techniques to mitigate the Sybil Attack in wireless ad hoc networks. Most methods that attempt to mitigate these attacks are based on a centralized authority, needing a pre-shared secret and thus, pre-configuration. However, this requirement makes such solutions unfeasible to implement in many civilian environments, since one cannot assume a common administrative entity with access to every node, and trusted by all participants. The main goal of this thesis is, therefore to propose a technique which can reliably test for sybil identities, without requiring the a priori configuration of nodes.

1.2 Contributions

This thesis is focused on a promising technique to mitigate sybil attacks called resource tests. We analyze and compare two specific kinds of resource tests, Radio Resource Tests (RRTs), and Computational Resource Tests (CRTs). Then we introduce some distributed nonce generation algorithms, that jointly with both kinds of resource tests, will be part of a final algorithm, the Non-Sybil Quorum (NSQ) algorithm, whose aim is to efficiently mitigate the sybil attack in an one-hop wireless network.

The contributions of the thesis can be enumerated as follows:

- Two new radio resource tests, an optimization to an existing test, and a comparative framework to assess the power and performance of the different resource tests.
- A brief survey, analysis and comparison of computational resource tests, including a new optimization to an existing test, given a set of relevant metrics.
- Two methods of creating a collaborative nonce in a completely distributed fashion, and an analysis on the guarantees that both of them provide.
- An algorithm that uses radio and computational resource tests as well as the nonce generation method, to mitigate the existence of sybil identities in an one-hop neighborhood of a wireless ad hoc network, in a scalable and robust fashion.

1.3 Research History

In its early stages, the main objective of this thesis was to study how to build a sybil-free quorum in a completely decentralized fashion, for multi-hop wireless ad hoc networks. However, this construction of the sybil free quorum proved to be much more challenging than expected, and we ended up being able to devise an algorithm only for the single-hop setting. We focused on the use of resource tests, since they allowed to create protocols that were truly decentralized, being therefore good building blocks for our techniques. The message cost of these tests, lead us to combine several types of resource tests to achieve an increased efficiency, resulting in the design of the NSQ protocol, which is the core of this thesis. In the process of realizing the difficulties of the original challenge, and finding adequate solutions for the vast number of problems that came up along the way, I have benefited from very fruitful discussions with all the members of the GSD team at INESC-ID. Moreover, the collaboration with João Leitão was of particular importance, since it allowed me to grow as a researcher.

The analysis and design of the Radio Resource Tests as been published in (Mónica, Leitão, Rodrigues, & Ribeiro 2009b). An overview of the NSQ algorithm has also been published, in (Mónica, Leitão, Rodrigues, & Ribeiro 2009a).

1.4 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2 presents the related work from a security perspective, addressing the vulnerabilities of wireless ad hoc networks, attacks, and possible countermeasures.

Chapter 4 describes Radio Resource Tests, and presents a framework to assess the power and performance of the different tests.

Chapter 5 describes Computational Resource Tests and several types of crypto-puzzles, presenting a framework that allows us to assess the limitations of the different tests, and make a comparison between them.

Chapter 6 describes and compares two distributed algorithms for the collaborative generation of a nonce.

Chapter 7 presents the Non-Sybil Quorum (NSQ) protocol, based on three essential modules: radio resource tests, computational resource tests and nonce generation algorithms.

Chapter 8 concludes the thesis and proposes some directions for future work.

2 Context and Related Work

This thesis addresses security issues on wireless ad hoc and mesh networks, with special emphasis on the sybil attack. This section introduces fundamental concepts, starting with a brief overview of the security issues present in this type of networks. Afterwards, the Sybil Attack will be introduced, followed by the description of the main existing techniques to address this attack.

2.1 Wireless Ad Hoc Networks

Infrastructured wireless networks typically require a fixed network structure, with centralized administration, for their operation. In contrast, wireless ad hoc networks consist of a collection of wireless nodes that communicate without using any infrastructure or administrative support. There are essentially three kinds of ad hoc networks: Mobile Ad Hoc Networks (MANET), Sensor Networks and Mesh Networks. While the basic principles of these three wireless networks remain the same, they have a few differences that make them worth of being separately addressed.

In the case of the MANETs, their main differentiating characteristic is the fact that the nodes are free to move without constraints, and organize themselves arbitrarily. Therefore, MANETs may have a highly dynamic membership and mobility, which means that the network topology may change rapidly and unpredictably. In sensor networks, nodes are typically static and resource constrained. In order to save resources, nodes may put themselves in sleep mode, which also causes changes in the network topology, since this sleep mode affects the communication patterns among nodes. Finally, mesh networks combine mobile wireless nodes with energy-unconstrained static wireless nodes that support routing in the network. Mesh networks are, for instance, appropriate to expand the wireless network connectivity in regions where there is limited access to an infrastructured network.

While the methods developed in this thesis will be applicable to all of these kinds of networks, we mostly focus on scenarios with low dynamics, and furthermore assume that nodes have sufficient computational power to use asymmetric cryptography, which puts our environment closer to that of a Mesh Network.

2.2 Security

The fact that ad hoc networks do not necessarily rely on a fixed infrastructure, raises many challenges for their security architecture (Vesa 2000). Issues that make hard to secure ad hoc networks are the vulnerability of the links, limited physical protection of each node, the absence of a certification authority, and the lack of centralized monitoring or management points (Hubaux et al. 2001), among others. Unfortunately, the differences between ad hoc and wired networks make the former not eligible for the same security solutions as the latter. While the basic security requirements such as confidentiality, integrity and authenticity remain the same, ad hoc networks restrict the set of feasible security mechanisms that can be used, mostly due to performance issues.

The security requirements of any network depend vastly on the type of application. While there are some networks that operate in a safe and friendly environment, most of them are deployed in hostile environments, subject to constant threats from attackers. An example of such a hostile environment can be found in networks deployed for military communications in direct conflict areas. In this case, all network components are physically vulnerable to tampering and must satisfy very stringent requirements regarding confidentiality and resistance to denial-of-service attacks. While not all applications of ad hoc networks have this kind of strict security requirements, every security solution may have to address the limited power, memory, and CPU available in each node, while still managing to provide strong protection against threats. This trade-off between performance and protection makes the design and implementation of strong protective measures in ad hoc networks a non-trivial problem.

The design of a security scheme requires, in general, the discussion of the vulnerabilities that need to be addressed, the description of the fundamental security components, and must take into consideration the attacks that currently exist to exploit those vulnerabilities. In what follows, several aspects will be addressed: the main vulnerabilities of wireless ad hoc networks;

the essential security needs of such networks; the main threats that violate such security needs (generally called *attacks*).

Vulnerabilities of ad hoc Networks From a security point of view, there are several reasons why wireless ad hoc networks are more vulnerable than their wired counterparts. Many of these reasons derive from vulnerabilities of the wireless medium of communication. Some characteristics that make these networks particularly vulnerable to attacks, will now be discussed. In this context, attacks are defined as procedures launched by unauthorized entities or nodes within the network, that exploit vulnerabilities with the intent of disrupting the network operation.

Lack of Secure Boundaries In a wired network, attackers need to have physical access to a node or to the medium in order to perform malicious activities. On the other hand, in wireless networks it is not possible to create the same sort of secure *boundary*. Once an attacker is in the radio range of any group of nodes, it can communicate with these nodes and attempt to join the network. As a result, ad hoc networks cannot rely on boundary lines of defense, such as firewalls and gateways, to protect the network from potentially harmful network accesses.

Furthermore, since all communications are performed over the air, ad hoc networks are specially vulnerable to attacks such as passive eavesdropping, active interference, leaking of secret information, data tampering, impersonation, message replay, message distortion, and denial-of-service (Mishra 2008). These vulnerabilities raise serious concerns, since there are many applications in which confidentiality is one of the major issues. For example, in military applications, confidentiality of the information is one of the most important attributes, as discussed in Hubaux et al. (2000). Also, without any authenticity and integrity protection (these definitions will be provided further ahead in the text), an attacker is able to destroy, create or even manipulate messages, allowing it to, ultimately, compromise the entire network. Availability is also a central issue in ad hoc networks, that must operate under dynamic and unpredictable conditions. In civilian scenarios, availability has the greatest relevance for the user (Stajano & Anderson 2002), and is also one of the most difficult properties to preserve, since the wireless medium is particularly susceptible to denial-of-service attacks, including resource exhaustion and jamming.

Threats from Compromised Nodes An attacker may attempt to compromise the links or the nodes of the ad hoc network. If the attacker gains control of one or more nodes in

the network, he can then use these compromised nodes to execute further malicious actions. One of the challenges in face of this attack is to detect accurately the nodes that have been compromised, as discussed in several works of intrusion detection (Veríssimo et al. 2006; Zhang & Lee 2000; Nadkarni & Mishra 2004). Usually, compromised nodes are detected by monitoring their behavior. Unfortunately, in wireless environments, it is difficult to distinguish a truly misbehaving node from a node with a poor link quality (Mishra 2008). In addition, and since the nodes that compose the ad hoc network can have a high behavioral diversity, it is hard to create effective policies that prevent all the possible malicious behaviors from every kind of existent node. Finally, the fact that nodes can join or leave the network with freedom, allows an attacker to frequently change target, and attack a different node in the network. This makes the task of tracking the malicious behavior performed by any compromised node inside the network, even more difficult.

Compromised nodes in the network may deliberately cause Byzantine faults (Veríssimo & Rodrigues 2001). Furthermore, a set of nodes may be compromised in such a way that their malicious incorrect behavior cannot be detected (Mishra 2008), making them a serious threat to the network. While compromised nodes may appear to be operating correctly, they can, for example, create new routing messages or advertise non-existent links, thus inflicting Byzantine faults on the system.

Lack of Centralized Management Facility The lack of centralized management has a significant impact on the design of security mechanisms for ad hoc networks, in particular because attack detection becomes a very hard task. For instance, attack detection based on traffic monitoring is challenging in highly dynamic and large scale ad hoc networks (Ilyas & Dorf 2003), since the malicious activities may be obfuscated by the frequent benign failures in wireless networks, such as path breakages, transmissions impairments and packet dropping (especially when attackers frequently change their targets and attack patterns).

On the other hand, the absence of an authorization facility provided by a pre-existing infrastructure, makes it very hard to distinguish trusted from untrusted nodes. This distinction is a line of defense that may be implemented by requiring trusted nodes to carry credentials that can be validated by the remaining trusted nodes. Unfortunately, in the case of ad hoc networks, no prior security association can be assumed for all network nodes. Consequently, algorithms

that rely on the cooperative participation of all nodes can be attacked by adversaries that make use of this vulnerability to undermine decentralized decisions (Krishnamurthy 2004).

Restricted Resources Typically, in wireless ad hoc networks, some or all of the network nodes rely on limited power sources (e.g. battery). Therefore, security mechanisms for such environments need to take into consideration the fact that power is a limited resource. For instance, the limited power of a node may be the basis to launch denial-of-service attacks (Anthony & John 1992) as follows. The attacker, being aware that his targets are battery-restricted, may, continuously send packets to their targets, asking to route those additional packets, or induce the target to perform time-consuming computations. This sort of activities will exhaust the battery of the attacked nodes and consequently, prevent them from answering legitimate service requests, as they exhaust their power supply and become out of service.

Power may not be the only scarce resource in wireless networks. Another such resource can be found in sensor networks, where besides power issues, nodes also have very limited processing capacity (Akyildiz et al. 2002). Sometimes, the working memory of a sensor node can be insufficient, even to hold the variables required for asymmetric cryptographic algorithms (Mishra 2008). This restrictions impose the use of other, potentially less secure, security mechanisms.

Scalability The size of an ad hoc network is a variable that changes frequently (Mishra 2008). This mostly happens due to node mobility, and network partitions or merges. As a result, protocols and services for ad hoc networks, such as routing protocols, must operate efficiently regardless of the system size, which can be in the order of just dozens or even hundreds of nodes.

Lack of Physical Security In wired networks, it is often possible to physically secure the access to nodes (for example, by keeping nodes in rooms with limited controlled access). This is rare in wireless networks. For instance, military nodes in a hostile battlefield scenario cannot use security mechanisms that rely on their physical security, due to the risk of being captured and compromised. Thus, security mechanisms must be able to operate in face of compromised nodes.

2.2.1 Security Criteria

Security in networks must address different issues such as confidentiality, data integrity, legitimate use and availability of services (Ford 1994). These issues need also to be addressed in wireless ad hoc networks (Mishra 2008). The fundamental concepts that will be used to discuss the network security aspects in the remainder of the text, will now be introduced:

- *Availability* is the ability of the network to continuously provide service, irrespective of attacks (Vesa 2000; Ilyas & Dorf 2003) (including denial-of-service attacks (Lau, Rubin, Smith, & Trajkovic 2000), like radio jamming or battery exhaustion). In Zhou and Haas (1999) *availability* is identified as one of the key attributes related to the security of networks;

- *Integrity* is the guarantee that a delivered message contains exactly the information that was originally sent (Mishra 2008; Ilyas & Dorf 2003). This guarantee precludes the possibility of messages being altered in transit. The causes of integrity violation may be accidental or malicious but, in practice, it is impossible to distinguish one from the other;

- *Authenticity* is the guarantee that participants in communication are genuine and not impersonators (Mishra 2008; Ilyas & Dorf 2003). To achieve authenticity, participants in the communication are required to prove their identities. Without this authentication, an attacker could impersonate a legitimate participant, allowing him to obtain access to confidential resources or disturb the normal network operation by propagating fake messages.

  In some cases, it is possible to wave authenticity if end-to-end integrity is assured. For instance, in a wireless sensor network, if the messages arriving at the destination reflect the sensed environment, it does not matter if they were sent by legitimate nodes;

- *Confidentiality* is the guarantee that certain information is only readable by those who have been authorized to do so. This prevents information from being disclosed to unauthorized parties. If authentication is performed properly, confidentiality is a relatively simple process (Mishra 2008; Ilyas & Dorf 2003);

- *Nonrepudiation* is the guarantee that the sender of a message cannot later deny having sent the information nor the receiver can deny having received it (Mishra 2008; Ilyas &

Dorf 2003). This can be useful in the detection of compromised nodes, since it makes it possible to prove the malicious behavior of a specific node, by presenting any erroneous message it may have sent;

- *Self-healing* refers to a protocol that is able to recover automatically from an erroneous state, in a finite amount of time, without human intervention. For instance, it should not be possible to permanently disable a network by injecting a small number of malicious packets at a given point in time. If a protocol is self-healing, an attacker must remain in the network and inflict continuous damage in order to prevent the protocol from recovering, a behavior that makes the attacker easier to locate (Mishra 2008);

- *Byzantine Robustness* is a property which ensures a protocol should be able to function correctly, even if some of the nodes participating intentionally attempt to disrupt its operation. Byzantine robustness can be seen as a stricter version of the self-healing property: the protocol must not only automatically recover from an attack; it should not cease its operation, even if performance is hampered during the attack.

2.2.2 Attacks

Taking into consideration the essential security criteria, the various kinds of possible attacks against ad hoc networks will now be discussed. The discussion will provide the basis for, proposing defenses and countermeasures against these attacks later in the thesis. The attacks can be classified into two broad classes, namely (Mishra 2008):

- *External attacks* initiated from outside the network, in which the attacker attempts to cause congestion in the network, propagate incorrect routing information, prevent services from working properly, or shut down the network completely;

- *Internal attacks* initiated from within the network, in which the attacker gains normal access to the network by compromising directly or by impersonating an existing legitimate node. The attacker then uses the access to the network to engage in malicious behaviors.

In the two categories shown above, the external attacks are somewhat similar to typical attacks in wired networks, in which the attacker can exchange messages with network nodes but

it is not a trusted node. These attacks can, therefore, be prevented and detected by conventional security methods such as membership authentication. On the other hand, internal attacks are far more dangerous, because the compromised nodes are originally legitimate nodes of the network and they can, therefore, pass the authentication and get protection from the security mechanisms (Mishra 2008).

Another way of classifying an attack can be done by the focus of the attack itself. Ad hoc networks are typically subjected to two different levels of attacks:

- *Passive Attacks* which consist on the attacker *eavesdropping* on the data that is being communicated in the network. Examples of passive attacks include traffic analysis and sniffing information, allowing an attacker to compromise secrets and keys in the network;

- *Active Attacks* which involve specific actions performed by adversaries, for example, the modification, replication, or deletion of the exchanged data among network nodes.

External attacks are typically active attacks in which an adversary attempts to change the behavior of the operational mechanisms of the network. This is opposed to passive attacks in which the adversary will be subtle on his activities, while gathering information that may be later used to launch an active attack.

Denial-of-Service (DoS) can either be produced by an unintentional failure or by malicious action. The usual way to create a DoS attack is to *flood* a centralized resource with an abnormal number of requests, preventing it from operating correctly, or even cause it to crash. But, the fact that ad hoc networks do not have any centralized resource and distribute responsibilities throughout all the nodes in the network, makes them a difficult target to this kind of attack (Zhou & Haas 1999; Hubaux et al. 2000). On the other hand, by using a distributed denial-of-service attack, an attacker that has compromised enough nodes, can congest the network rather easily, rendering it useless. Better yet, a motivated and resourceful attacker can completely deny the service to the nodes by using radio jamming, an attack which makes the communication medium unusable at the physical level, or by using battery exhaustion (Anthony & John 1992), leveraging on the constrained power source available to nodes.

Eavesdropping The goal of eavesdropping is to obtain confidential information from messages exchanged among legitimate nodes. This attack is facilitated by the use of wireless links, since any node in the radio range of the communicating participants can eavesdrop their exchanged messages. To prevent eavesdropping, every critical data passing in the network, including control data, should be encrypted with strong cryptographic mechanisms.

Attacks on Information in Transit Any compromised or malicious node can utilize the information it forwards, for example, by executing routing protocols, to launch attacks. The attacker can maliciously intercept, modify, or fabricate routing messages that pass through him. These attacks can lead to the corruption of information, disclosure of sensitive information, theft of legitimate service from other protocol entities, or denial of network service to protocol entities (Mishra 2008). Several attacks against routing protocols have been studied and are now well known (Papadimitratos & Haas 2002; Hu et al. 2002; Dahill et al. 2002; Hu et al. 2002).

Node Hijacking It is possible for a malicious node to masquerade as the base station and encourage users to connect to it. That node will then be in a privileged position to collect private data, such as: passwords, secret keys, logon names, etc. This is an example of a node hijacking where a legitimate base station has been hijacked by an attacker. There can be also other kind of node hijacking called "route hijacking", where the attacker modifies the routing information in order to hijack traffic to and from selected nodes (Mishra 2008).

Impersonation attacks pose a serious security risk to ad hoc networks. If the security mechanisms cannot support proper node authentication, compromised nodes may be able to impersonate trusted nodes. This kind of threat can be mitigated by the use of strong authentication mechanisms like digital signatures (Vesa 2000). Due to the fact that digital signatures are implemented with public-key cryptography, they require high computational power and efficient and secure key management (Mishra 2008), something that most ad hoc network nodes are unable to provide due to the lack of resources. Due to this fact, hybrid encryption mechanisms like Message Authentication Codes (MAC) (Bellare et al. 1996), can be used.

The Sybil Attack A Sybil attack is essentially an impersonation attack, in which a malicious device illegitimately fabricates multiple identities, behaving as if it were a larger number of nodes

(instead of just one) (Douceur 2002). A malicious device multiple identities are referred to as *Sybil identities* or *Sybil nodes*. According to the taxonomy presented by Newsome et al (2004), there are three possible orthogonal dimensions for this attack: direct vs indirect communication; fabricated vs stolen identities; and simultaneity. In the worst case, an attacker can create an unlimited number of Sybil identities, with only one malicious device.

Direct vs. Indirect Communication

- *Direct communication* One way to perform the Sybil attack is for the Sybil nodes to communicate directly with the legitimate nodes. This means that, when a legitimate participant sends a radio message to a Sybil node, the malicious device listens to the message. Symmetrically, when any of the sybil nodes sends a message, the messages are actually sent from the malicious node device;
- *Indirect communication* In this version of the attack, the legitimate participants cannot directly communicate with the Sybil nodes. One or more malicious devices simply claim to be able to reach a number of Sybil nodes. This way, every message sent to a Sybil node is routed through one of these malicious node, which pretends to pass them to the final destination.

Fabricated vs. Stolen Identities There are essentially two different ways in which a Sybil node can get an identity: it can fabricate one (for instance, creating an arbitrary identifier) or it can *steal* an existing valid one from a legitimate node.

- *Fabricated Identities* If there is no network restriction to the allowed identities, or some way of verifying that an identity is legitimate, a malicious node can simply generate an arbitrary identity, and use it to join the network;
- *Stolen Identities* If there are mechanisms to prevent bogus identities from joining the network (for example, a limited namespace to prevent attackers from inserting new identities), the attacker may try to assign legitimate identities to the Sybil nodes. This identity theft may go unnoticed, if the attacker can, somehow, disable the impersonated nodes.

Note that the sybil attack with stolen identities, is in its essence a pure impersonation attack, and as was previously seen, can be solved with the use of cryptographic mechanisms,

like digital signatures, or MAC codes. Our thesis will thus focus on the sybil attack with fabricated identities.

Simultaneity

- *Simultaneous* While a particular hardware entity can only advertise one identity at a time, it can cycle through these identities to make them appear to be present simultaneously. This way the attacker can have all his Sybil identities participating in the network at the same time;
- *Non-simultaneous* Alternately, the attacker can present a large number of identities over a period of time while only acting as a smaller number of identities at a given time. Also, if the attacker has several compromised nodes, he can make the nodes swap identities periodically, making detection even harder.

Specific Attacks There are several known applications of Sybil attacks for wireless ad hoc networks (Newsome et al. 2004; Karlof & Wagner 2003).

- *Routing;* Sybil attacks have been shown to be effective against routing protocols in ad hoc networks (Karlof & Wagner 2003). One specially vulnerable mechanism is *multipath* or *dispersity* routing, where seemingly disjoint paths could, in fact, go through different Sybil identities of the same malicious node. Geographic routing is another vulnerable mechanism, where a Sybil node could appear in more that one place at once (Karlof & Wagner 2003);
- *Data aggregation;* Sensor networks make use of query protocols, which compute aggregates of values, obtained through sensor readings within the network, to conserve energy (rather than return each sensors individual reading)(Newsome et al. 2004). In scenarios with a small number of malicious nodes reporting erroneous sensor readings, the overall result may not be affected by a wide margin. Still, if the malicious sensors make use of the Sybil Attack, they can fabricate enough Sybil Nodes to alter significantly the outcome of the reading aggregation;
- *Voting;* The Sybil attack may be used to alter the outcome of a voting scheme. If, for example, there is a voting scheme to determine node misbehavior in a network, an attacker

can create enough false Sybil nodes to be able to expel any target node from the network. Conversely, if there is a vote on whether the attacker's identities are legitimate, the attacker may use his Sybil nodes to *vouch* for each other;

- *Misbehavior detection;* If there is a mechanism in the ad hoc network to detect malicious behavior, an attacker launching a Sybil attack can escape detection by "spreading the blame" throughout all the Sybil nodes. If the mechanism requires several observations of this behavior to take action, by using different nodes, the attacker can escape detection completely. Even if, somehow, some Sybil Nodes are expelled from the network for malicious behavior, the attacker can always create more identities, and avoid being caught;

- *Fair resource allocation;* Some network resources may be allocated on a per node basis. If, for example, the radio channel allocation is done by using time slots (TDMA MAC, for example), with the use of a Sybil Attack, the attacker can gain access to more radio resources. This both denies service to legitimate nodes by reducing their share of the resource, and gives the attacker more resources to perform other attacks.

2.3 Countermeasures Against the Sybil Attack

As just described, the Sybil attack is a fundamental problem in many systems, for which no universally applicable solution has been devised. Analysis of the Sybil attack has been done in the context of peer-to-peer applications in a wired context (Cheng & Friedman 2005; Douceur 2002). A number of approaches that can be used to protect from, or detect, this attack, as surveyed in Levine et al. (2006), are summarized below.

2.3.1 Trusted Certification

Trusted certification is the most common solution, mainly due to its potential to completely eliminate Sybil attacks (Douceur 2002). However, trusted certification relies on a centralized authority, that must provide guarantees that each node is assigned exactly one identity, as indicated by possession of a certificate. In fact, Douceur (2002) offers no method for ensuring such uniqueness, and in practice, it has to be performed by a manual configuration. This manual procedure can be costly, and create a performance bottleneck in large-scale systems. Additionally,

and in order to be effective, the certifying authority must guarantee the existence of a mechanism to detect and revoke lost or stolen identities. These requirements make trusted certification very difficult to implement in ad hoc networks, which lack, by definition, a centralized authority that can provide the certification service.

While there are some solutions that reduce the network dependency on a centralized authority, for instance, requiring the presence of a certification authority only in the bootstrap of the network (Martucci et al. 2008), there is still an additional problem with the use of a centralized authority: the possible existence of multiple administrative entities. If, there is only one common administrative entity managing the whole network, the implementation of a trusted certification, while having the problems stated above, can be a viable, if not perfect, solution. However, different administrative entities usually have different certification authorities. For example, consider an ad hoc network composed of nodes that do not belong to the same entity and, perhaps, have never met before. In this scenario, even if a node possesses a legitimate certificate from some certificate authority, it would not be recognized as legitimate by any other node in the network, since they are not under the administration of that entity. This limits the environments in which the implementation of a trusted certification mechanism is possible.

2.3.2 Trusted Devices

The use of trusted devices can be combined with trusted certification, binding one hardware device to one network entity. While this can effectively mitigate the Sybil attack, the main issue with this approach is that there is no efficient way to prevent one entity from obtaining multiple hardware devices other than manual intervention (Martucci et al. 2008).

2.3.3 Domain Specific

There are some countermeasures that are application-domain specific. For example, in Piro (2006), a detection mechanism for mobile ad hoc networks is proposed, based on the location of each node. For an attacker with a single device, all Sybil identities will always appear to move together. However, the defense is not applicable beyond mobile networks.

2.3.4 Resource Testing

The main goal of resource testing is to attempt to determine if a number of identities possess fewer aggregated resources than would be expected if they were independent. In resource testing, it is assumed that each physical entity has a limited amount of a given resource (e.g., limited bandwidth). The verifier then tests whether identities correspond to different physical entities by verifying that each identity has as much resources as an independent physical device should have. These tests include checks for computing power, storage ability and network bandwidth (Douceur 2002). A type of resource test is employed by the SybilGuard technique (Yu et al. 2008), which relies on the limited availability of real-world *friendship* edges between nodes.

2.3.5 Recurring Costs and Fees

There are several works in the literature that describe mechanisms in which identities are periodically re-validated using resource tests (Maniatis et al. 2003; Maniatis et al. 2005). This technique is a variation of the normal resource testing, and can limit the number of Sybil nodes an attacker, with constrained resources, can introduce in a period of time. Recently, it was shown (Boris & Levine 2008), that charging a recurring fee for each participating identity is more effective as a disincentive against Sybil attacks. For many applications, recurring fees can incur a cost to the Sybil attack that increases with the total number of identities participating; whereas one-time fees incur only a constant cost.

2.3.6 Computational Resource Testing

Computational resource tests are a specific type of resource testing that relies on the assumption that the network nodes possess a limited computational power. The use of the technique was firstly proposed in Dwork et al. (1993) as a method for combating junk email ("spam"), and later used as a defense against denial-of-service attacks (A. Juels 1999; Back 2002). In Aspnes et al. (2005) the authors introduced a computational resource test (CRT), to verify if participants own an expected amount of computational power. The test, entitled crypto-puzzle, consists on having identities solving a cryptographic problem, solvable only by brute force calculation, in a certain amount of time. This way, a node with constrained computationally power has a limit

on the number of crypto-puzzles it can solve in a given time period, thus setting an upper bound on the number of sybil identities it can present to the network.

2.3.7 Radio Resource Testing

In this context, radio resource testing, is another specific type of resource testing, which relies on the assumption that nodes only possess one radio device, building upon the limitations of these devices. A particular example of such an limitation is the incapability of a radio simultaneously transmitting in two different frequencies. This idea has been used in Newsome et al. (2004), to counteract the sybil attack. However, the authors do not address the details that would allow them to build a protocol capable of operating in real world scenarios. Therefore, they do not present a comprehensive study on the cost and complexity of solutions based on this technique.

In Piro et al. (2006), a somewhat different kind of test was proposed. Contrarily to the one used in Newsome et al. (2004), this test is completely passive, in the sense that it does not require the active participation of nodes. Despite this difference, the main assumptions remain the same. The authors proposed a protocol, named PASID-GD, that identifies sybil identities, by comparing the number of expected and observed collisions. However, this kind of passive approach has a practical problem: it allows the normal participation of sybil identities in the network until they are detected. This means that an attacker can continuously generate new sybil identities to take the place of those that are detected, thereby continuously participating in the network with multiple identities. Conversely, the use of active tests allow us to guarantee that no sybil identities participate in the network, before being properly tested.

Additionally, a number of techniques that rely on location information to detect sybil identities have been proposed (Bazzi & Konjevod 2005; Demirbas & Song 2006; Piro et al. 2006; Xiao et al. 2006). Such information can be either inferred using radio signal strength indication (Demirbas & Song 2006), or by relying on external components to provide such information *e.g.* GPS (Xiao et al. 2006). While the existence of external location sources is plausible for vehicular networks, they are not typical for ad hoc networks[1]. On the other hand, radio strength based approaches can be easily attacked, by varying the transmission power, leading

[1]For instance, it is not typical for laptops to be equipped with GPS receivers.

to inaccurate detection of sybil identities.

Summary

In this chapter we introduced the fundamental concepts, starting with a brief overview of the security issues present in wireless ad hoc networks. We presented a series of security metrics, that will be used to discuss the network security aspects in the remainder of the text. Furthermore, we have described the main attacks to which these networks are vulnerable, with emphasis on the Sybil Attack, and some known countermeasures against this attack, including those that we will further explore in this thesis: Radio Resource Tests and Computational Resource Tests.

3 Model

In this chapter we will present and detail the system model that will be used throughout the remainder of the text. All necessary prior setup conditions will be stated, relevant terminology will be established, and the underlying assumptions will be discussed. The chapter ends with the formalization of the problem statement.

3.1 System Model

In our system, we assume a pre-existent wireless network, constituted by N nodes, all of them located in the one-hop neighborhood of all other nodes. Further assumptions on the characteristics of nodes, communication channels and network operations, are detailed below.

3.1.1 Correct and Byzantine

Nodes in the network can be either malicious (byzantine) or correct. We will use the terms malicious and byzantine interchangeably. Byzantine nodes may exhibit arbitrary behavior, such as sending arbitrary messages, and may collude with each other. At most, f nodes may be byzantine ($f < N$).

3.1.2 Synchronism

We assume that, during algorithm execution, the network operation is synchronous. The synchronism does not need to be established at physical level. Less stringent types of synchronism, operating at the logical level will suffice, as discussed later. Furthermore, we model time as a sequence of successive time-periods (steps). In each step, all nodes may send or receive a message using the shared wireless medium. However, in each step, at most one message may be

sent in any of the available radio channels. If, at any given step, two or more nodes attempt to send a message in the same radio channel, a collision occurs.

3.1.3 Limited Resources

All nodes, correct or byzantine, are assumed to have a single radio device, and limited computational power. The radio devices can operate in K different channels. However, radio devices are assumed to be incapable of simultaneous operation in more than one channel. We limit the number of transmissions c_i that all nodes (including the byzantine), are able to perform, in a given number of consecutive slots; otherwise, network operations could be inhibited by simple frequency jamming, from a malicious node. This effectively limits the maximum number of collisions byzantine nodes are able to generate. Correct nodes can, however, still generate non-intentional collisions.

3.1.4 Communications

Nodes communicate through message passing over a one-hop shared wireless medium. We assume that all messages are sent in broadcast mode, in a single channel, and received by every other node listening to that channel, in the one-hop neighborhood. Unless stated otherwise, communication among nodes is performed using a single, default, channel. As will be seen, the remaining channels are used to perform radio resource tests. We assume also that the transmission channels are reliable: when there are no collisions the messages are delivered without loss or corruption. Furthermore, we assume that, whenever a collision happens in the wireless medium, all nodes, including the senders, are able to detect it. Whenever a collision happens, no node is able to receive any message.

3.1.5 Nodes and Identities

All messages sent by a node must be identified with the node's *identity*. Correct nodes are assumed to use a single identity that is not used by any other node. The receiver of a message can always correctly associate the received message with the originating identity, thus avoiding identity spoofing. This can be guaranteed by using public-key cryptography, and having each identity signing outgoing messages. Note that the use of public-key cryptography does not

require the use of a PKI: public-private asymmetric key pairs may be locally and randomly generated by a node without coordination with other nodes; the public key may then be used as the node's identity, and the message signed with the private counterpart.

Malicious nodes may try to use more than one identity for communication. These multiple identities, are called sybil identities. We denote by $ids(n_i)$ the set of identities used by node n_i. Conversely, we denote by $uses(id_i)$ the set of nodes that use identity id_i. Thus, for identities used by correct nodes, we have: $id_i = ids(n_i)$ and $n_i = uses(id_i)$. There is no limit to the number of identities $|ids(b)|$ that a byzantine node b may use. Also, since byzantine nodes can collude, the same identity may be used by multiple byzantine nodes. Thus, if id_b is an identity used by a byzantine node, we have $1 \leq |uses(id_b)| \leq f$.

3.1.6 On the Coverage of the Assumptions

As was stated, we assume that when a collision happens in the wireless medium, all nodes are able to detect it. This can be achieved by using a simple mechanism as follows: Upon detecting a collision, correct participants transmit for enough time to ensure that all nodes that transmitted the colliding messages are able to detect the collision. A similar method was used with this same objective in (Fullmer & Garcia-Luna-Aceves 1995).

We also assume a limitation on the number of transmissions the nodes are able to perform in a given time period. This is typically due to power constraints of the nodes. Note that, otherwise, the communication medium could be targeted by denial-of-service attacks; for instance, by having a byzantine node continuously transmitting in a communication channel. We thus assume no such attacks are possible, and refer the reader to some techniques that are able to mitigate this kind of attack, to some extent (Khattab, Mosse, & Melhem 2008; Noubir 2004).

Channels are assumed to be reliable. This assumption simplifies the description and the analysis of our algorithms. We believe that faults in the communication may be encapsulated by using as a building block protocols that mask such faults using retransmissions, such as (Koo, Bhandari, Katz, & Vaidya 2006). However this direction has not been addressed in the thesis.

Finally, we assume that all communication is one-hop. An attacker may control the power of its radio device to cover just a subset of the nodes. Solutions to address this problem are left for future work.

3.2 Problem Statement

Our objective, is returning in each correct node n_i, a quorum of identities NSQ_i whose target (and maximum) size is given by a parameter q. The identities in each NSQ_i belong to correct or byzantine nodes. However, since faulty nodes may stop operating at any point, the returned quorum may include less identities (but at least $q - f$). Furthermore, the intersection of the quorum returned to different nodes contains at least $q - f$ identities from correct nodes. More precisely, the problem can be defined using the following properties:

Sybil-Free Partial Consistent Quorum Set There is a set of identities from correct nodes NSQ, common to every NSQ_i, such that $q \geq |NSQ| \geq q - f$, with a probability arbitrarily close to 1. Let $\Gamma()$ be defined as:

$$\Gamma(n_i) = \begin{cases} 1, \text{ if } n_i \text{ is a correct node} \\ 0, \text{ otherwise} \end{cases} .$$

Let $NSQ = NSQ_1 \cap NSQ_2 \cap ... \cap NSQ_{N-f}$ be the set of identities common to all NSQ_i of the correct nodes. Then,

$$\sum_{n_i}^{uses(NSQ)} \Gamma(n_i) \geq q - f.$$

Probabilistic Termination Every node returns the quorum with a probability arbitrarily close to 1, given a finite number of steps.

The target size of the NSQ, q, is a protocol parameter. Notice that we cannot prevent byzantine nodes from joining the quorum. If a byzantine node behaves correctly, it is undistinguishable from a correct node, and thus, it may be possible for it to become part of the final quorum. Consequently, the size of q must be selected according to the type of applications of the quorum returned by the NSQ algorithm. Typically, q must be set high enough to ensure that number of faulty nodes is not enough to defeat the application, for instance $q = 3f + 1$ in (Lamport, Shostak, & Pease 1982).

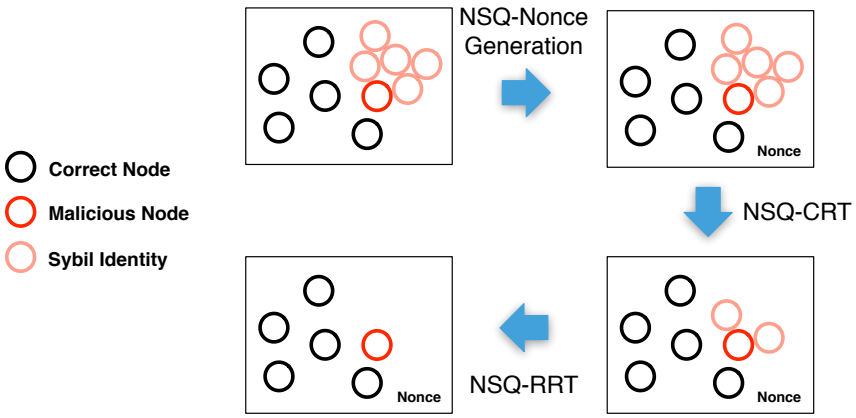

Figure 3.1: Building blocks used for the Non-Sybil Quorum Algorithm.

3.3 Overview of the Solution

An algorithm to solve the stated problem will be provided in Chapter 7. Our solution relies on a combination of different resource tests and requires the use of a number of building blocks that are going to be gradually introduced in the next chapters. These blocks are represented in Figure 3.1, in the order they are used in the NSQ. The first block being used is the Nonce Generation (Chapter 6). The nonce generation algorithm outputs in every node a random and previously unknown nonce. This nonce is required to execute a computational resource test. Then, a Computational Resource Test (Chapter 5) is used to eliminate most of the existing Sybil identities in the network. Unfortunately, using this test alone is hard to ensure that no sybil identities remain. Finally, the Radio Resource Test (Chapter 4) eliminates the remaining Sybil identities in the system, testing the identities from the output of the CRT. As it will become clear in the text, this last test is robust but very expensive to be used on a large population of identities without the filtering provided by the previous two steps.

Summary

In this chapter, we introduced the system model that will be used throughout the remainder of this thesis, and described precisely the problem that will be addressed. In the next chapters we are going to present three different building blocks, that will ultimately be used by our Non-Sybil Quorum (NSQ) algorithm to mitigate the Sybil attack.

4 Radio Resource Tests

As previously discussed, Radio Resource Tests (RRTs) are a particular case of the more general class of arbitrary resource tests. RRTs are of particular interest, since they have the potential to support protocols that do not require pre-configuration nor pre-shared secrets, improving the scalability of the network. In this chapter, we will present two novel RRT tests and an optimization to a previously proposed test. Furthermore, we provide a framework to analyze and compare RRTs, for a set of relevant metrics. Finally, we will finish the chapter by further detailing one of the proposed radio resource tests, and providing proof of some of its relevant properties properties.

4.1 Framework Definition

A radio resource test is typically based on the following assumptions about radio devices (Newsome, Shi, Song, & Perrig 2004): *A1)* Each node in the network owns at most a single radio device; *A2)* Each device can operate over, at most, K different channels; *A3)* No radio device can simultaneously transmit on two different channels; *A4)* No radio device can listen simultaneously on two different channels; *A5)* A node cannot detect a collision while transmitting.

Note that, while it seems that assumption *A5* contradicts our own assumptions on Chapter 3, it does not. In our model we assume that collisions are possible to be detected, since the other non-transmitting network nodes are able to detect them, and can, therefore, warn the senders by reinforcing the collision. However, in assumption *A5* we state that a single node is unable to detect a collision while transmitting (by itself).

Each particular test may use only a subset of these assumptions. Moreover, note that even if assumption *A1* does not hold, we can model nodes with more than one radio devices as multiple colluding nodes.

Each *RRT* is characterized by a set of parameters $RRT(h, c, w)$ as follows. Parameter h is the size of the set $S = \{s_1, s_2, ..., s_h\}$ of distinct identities that can be tested simultaneously, in a single test. Parameter c is the number of *challenger* identities (not in S) that need to actively participate in the test. Parameter w is the number of *tester* nodes that can extract information from the test.

Simultaneous Sender Test (SST) As described before, in (Newsome, Shi, Song, & Perrig 2004), the authors proposed a RRT based on having the entities to be tested transmit simultaneously on different channels. In their original paper, the authors call their test "radio resource test" but, here, we dubbed it Simultaneous Sender Test (or simply: SST) to distinguish it from the remaining tests. As originally proposed, and in light of our framework, SST is a $RRT(K, 1, 1)$, *i.e.*, a test that allows a single node to test simultaneously as many identities as the number of channels available to the radio devices.

The test operates as follows: The challenger assigns a different channel to each identity being tested. Then, these identities start transmitting simultaneously. According to *A3*, sybil identities will be unable to simultaneously transmit on their assigned channels. The challenger can then listen to a channel at random, to verify if the corresponding identity is actually transmitting. If one of the tested identities does not transmit on the assigned channel, it is assumed to be a sybil identity. Since the challenger can only check one channel at a time, the reply transmissions have to be repeated r times, to achieve a desired probability of detection. The required number of rounds r will be derived at a later section.

4.2 Tests Being Proposed

4.2.1 Optimized Simultaneous Sender Test (oSST)

As the name implies, the Optimized Simultaneous Sender Test (oSST) is an optimization of SST. We do not list this test as a novel test, since the optimization is incremental (although powerful, as shown later in this chapter).

The oSST is based on the fact that the test proposed in (Newsome, Shi, Song, & Perrig 2004) can be used as a $RRT(K, 0, N - K)$, where N is the number of nodes in the one-hop

neighborhood of the nodes being tested. In fact: *i)* for a set of k identities ($k < N$) to be tested, it is possible to devise a deterministic channel assignment algorithm, thus avoiding the need for an explicit challenger ($c = 0$), and; *ii)* any node in the one-hop neighborhood of the nodes being tested can be a tester, *i.e.* several nodes can detect, at the same time, the existence (or nonexistence) of sybil identities in the set being tested.

4.2.2 Simultaneous Receiver Test (SRT)

Both the SST and the oSST have the disadvantage of being very asymmetrical in resource usage: all the tested identities need to transmit during the test. A set of malicious nodes may, therefore, drain the power resources of the network by issuing successive challenges, possibly with distinct sybil identities.

To avoid the problem, we now propose a novel $RRT(K, 1, 1)$, the Simultaneous Receiver Test (SRT). As with SST, the challenger assigns a different channel to each of the K tested identities. However, in the SRT test, tested identities have to listen in those channels. The challenger then sends a message in one of these channels, chosen at random. The corresponding identity is then required to echo this message. If one or more of the identities being tested are sybil, accordingly to *A4*, they will be unable to listen in all channels simultaneously, and there is a probability that the message will not be echoed. As before, the challenger may need to perform multiple rounds, to ensure that sybil identities can be reliably detected.

4.2.3 Forced Collision Test (FCT)

All the tests described so far require the radio devices to operate in more than one channel (i.e., $K \geq 2$). We now propose a test that can be performed in settings where radio devices are limited to a single channel.

The test is based on assumption $A5$, a known limitation of radio devices (Bar-Yehuda, Goldreich, & Itai 1991). The Forced Collision Test (FCT) is a $RRT(2, 1, 1)$, where one challenger can test two different identities, s_1 and s_2. The test operates as follows: s_1 is required to transmit a message M to s_2. If s_2 receives M, it should retransmit it. During the transmission of M by s_1, the challenger randomly decides to i) cause a collision on the wireless medium, or ii) listen to the medium to verify compliance of s_1. If s_1 and s_2 are different identities, s_2 will be able

to retransmit M if there was no collision, and unable to do so otherwise. If s_1 and s_2 are sybil identities, the malicious node that controls them will have to guess if a collision was generated or not. As in all previous tests, the test must have r rounds, to achieve the desired detection probability.

4.3 Analysis

We now discuss the power and performance of each of the four tests presented earlier: SST, oSST, SRT and FCT. This analysis considers the following metrics: vulnerability to collusion, message cost, ratio of resource consumption between legitimate and sybil nodes, and synchronization requirements.

4.3.1 Vulnerability to Collusion

Collusion occurs when two or more malicious nodes coordinate their efforts to protect one or more sybil identities. For instance, some malicious nodes may vouch for the sybil identities of other malicious nodes being tested, making it impossible to identify such identities as being sybil.

Intuitively, one can circumvent colluding nodes by testing simultaneously more identities than the existing number of colluding nodes. This would ensure that, in these tests, all colluding nodes would have to vouch for one of their own identities, and, thus, all remaining sybil identities would eventually be identified as so. In this context, we will use m to denote the size of the largest group of colluding malicious nodes ($m \leq f$).

More precisely, to ensure that a radio resource test $RRT(h, c, w)$ operates correctly in environments with at most m colluding nodes[1], we must have $h > m$. Due to this fact, SST, oSST, and SRT can tolerate as many as $h - 1$ colluding nodes. Notice that these protocols are limited by parameter h, whose maximum value depends on the total number of radio channels available to nodes. On sharp contrast, because FCT can only be performed on a pair of nodes, this protocol cannot operate correctly in the presence of colluding nodes. A single pair of colluding nodes can vouch for an arbitrary large number of sybil identities.

[1]Notice that, in scenarios without colluding nodes: $m = 1$.

4.3.2 Message Cost

We now discuss the message cost of each test. We also derive the number of rounds (r) required to detect sybil identities with a given target probability. Before discussing r, let us look at the cost of a single round for each test (mt). In SST and oSST, each round requires every tested node to send one message ($mt = h$)[2]. In SRT, at most two messages ($mt = 2$) are exchanged (one from the challenger and its echo from the tested node). In FCT, two messages are generated ($mt = 2$), one from one of the tested nodes and another from the other tested node (no forced collision) or from the challenger (forced collision case).

The probability of detecting a given sybil identity in S after r rounds of a $RRT(h, c, w)$ test (p_d) is, for each of the w tester nodes, given by:

$$p_d = 1 - \left(1 - \frac{1}{h}\right)^r.$$

Solving for r, one can calculate the number of rounds required to attain a specific detection probability in a single test of r rounds:

$$r = \frac{log\,(1 - p_d)}{log\left(1 - \frac{1}{h}\right)}.$$

The number of messages can be obtained by multiplying the number of rounds r by the number of messages in each round mt. This aspect will be further pursued ahead.

4.3.3 Resource consumption

We now discuss the onus that a $RRT(h, c, w)$ test imposes on legitimate nodes when a malicious node is involved in the test. The malicious node can be the challenger or the owner of one sybil identity being tested. In this context, we define the *resource consumption cost* as the difference Δ between the number of messages sent by correct nodes and messages sent by the malicious node. Notice that RRTs with higher cost levels are more vulnerable to denial-of-service (DoS) attacks.

[2]Depending on the particular implementation, we could also need to consider the challenging message.

	Malicious Challenger	Malicious and Sybil in S	Only Malicious in S
SST (oSST)	$rh-1$	$rh+1-3r$	$rh+1-2r$
SRT	0	$\frac{2h-3}{h} \cdot r$	$\frac{2h-2}{h} \cdot r$
FCT	r	$-0.5 \cdot r$	$0.5 \cdot r$

Table 4.1: Δ of resource consumption, for each RRT.

In SST, if the challenger is malicious, it sends a single message to initiate the test and then each of the h correct nodes sends a message on the assigned channel. Since there are r transmission rounds, the value of Δ is $rh-1$. If both the malicious node and its sybil identity are in S, than the sybil will not reply to the challenger query and, therefore, $\Delta = rh+1-3r$. If a malicious node is being tested, but its sybil is not in S, then $\Delta = rh+1-2r$.

In SRT, if the malicious node is the challenger, then Δ has a value of 0 (zero), since the challenger has to send a message for each reply. If both a malicious node and its sybil identity are being tested, $\Delta = \frac{2h-3}{h} \cdot r$. If a malicious node is being tested, but its sybil is not, then $\Delta = \frac{2h-2}{h} \cdot r$.

Finally, for FCT Δ has a value of r, if the malicious node is the challenger, $-0.5 \cdot r$ if both the malicious node and its sybil are being tested, and $0.5 \cdot r$ if a malicious node is being tested but its sybil is not.

Thus, we have that for $h \geq 5$, the SST always has the worst cost of all the tests, since it is possible for an attacker to consume a large number of network resources, with a low corresponding effort. These results are summarized in Table 4.1.

4.3.4 Synchronization Requirements

All RRTs compared in this paper assume that the participants in the test have exclusive access to the medium for the duration of the test. Otherwise, nodes not participating in the test might generate an unbounded number of collisions that, in turn, would make the tests inconclusive.

Also, some tests (such as SST and oSST) require nodes to transmit "simultaneously". However, in practice, nodes are not required to have a perfect synchronization; it is enough to ensure that the time to transmit a message is orders of magnitude larger than the allowed amount of

desynchronization among nodes (such that a node cannot leverage on the desynchronization to send a message on both channels).

4.4 Using RRTs for Population Control

For any $RRT(h, c, w)$, the number of simultaneously tested identities h cannot be more than the number of available channels K ($h \leq K$). Naturally, the number of identities that need to be tested may be greater than h. Therefore, to test a population $\mathcal{U}$ composed of N identities, one has to execute a given $RRT(h, c, w)$ several times, to cover all possible combinations. Thus, the final cost of using a given RRT to test an entire population depends on both the cost of each individual test and also of the number of required tests.

As previously stated, it will be assumed that all nodes in the system are in radio range (*i.e.* a single hop scenario), and that a common Time Division Multiple Access (TDMA) scheduler exists, to avoid collision and simplify the scheduling of individual tests.

In order to check if there are any sybil identities in $\mathcal{U}$, each node i must test every group of size h ($h < N$) in $\mathcal{U} \setminus \{i\}$. However, detection of a sybil identity can only occur in groups that include all the colluding malicious nodes ($m < h$) and that sybil identity. There are $\mathcal{G}$ such groups, where:

$$\mathcal{G}(N, h, m) = \binom{N - m - 2}{h - m - 1}.$$

Taking this into consideration, the probability that a challenger node detects a particular sybil identity becomes:

$$p_d^+ = 1 - \left(1 - \frac{1}{h}\right)^{r.\mathcal{G}(N,h,m)}. \tag{4.1}$$

Since $1 - \frac{1}{h} < 1$, we have that $p_d^+ \geq p_d$ for the same r, which means that the probability of a sybil identity being detected when a whole population is tested is bigger than when a single group of h nodes is tested. This is so because, when a population is tested, every combination of h identities must be tested.

In order to ensure a consistent view of $\mathcal{U}$ by all non-malicious nodes in $\mathcal{U}$, we require each sybil identity to be detected by every non-malicious node. The overall probability of detection

of a particular sybil identity is thus given by:

$$p_d^* = (p_d^+)^{N-m-1}. \tag{4.2}$$

One could avoid requiring all nodes to perform all the tests, since a node could, upon detecting a sybil identity, simply broadcast a warning. That would make the remaining nodes ignore the sybil identity and avoid further testing. Unfortunately, although this approach allows large improvements on the performance of RRTs, one has to consider that it also creates the opportunity for a simple attack against the group membership of correct participants. If a malicious node broadcasted sybil notifications concerning correct participants, they would be expelled from the group, which can be used at a later time by the malicious node to attack the system (*e.g.* to attack some majority based protocol being executed in the system).

Figure 4.1 shows the value of p_d^* for a network composed of 10 identities ($N = 10$) and $m = 1$, for distinct values of r and h. For reference, we also represent $p_d^* = 0.95$, which is the value we will use in the following sections as the minimum acceptable level of detection probability[3].

Solving equations (4.1) and (4.2) for r allows the specification of the number of rounds required for a desired probability of detection.

$$r = \mathcal{R}(N, h, m, p_d^*) = \left\lceil \frac{log\left(1 - \sqrt[N-m-1]{p_d^*}\right)}{\mathcal{G}(N, h, m).log\left(1 - \frac{1}{h}\right)} \right\rceil$$

4.4.1 Number of Tests

We now express the total number of tests ($\mathcal{NT}$) as a function of N, h, m and p_d^*.

As previously described, each node will perform r rounds of tests to each possible combination of h identities of all remaining nodes in the system ($N - 1$). Considering this, one can easily derive $\mathcal{NT}$ for this protocol as being:

$$\mathcal{NT}(N, h, m, p_d^*) = \mathcal{R}(N, h, m, p_d^*) \cdot \binom{N-1}{h} \cdot N$$

[3]Although we consider $p_d^* \geq 0.95$ as a case study, tests can be configured to any desired target value of p_d^*.

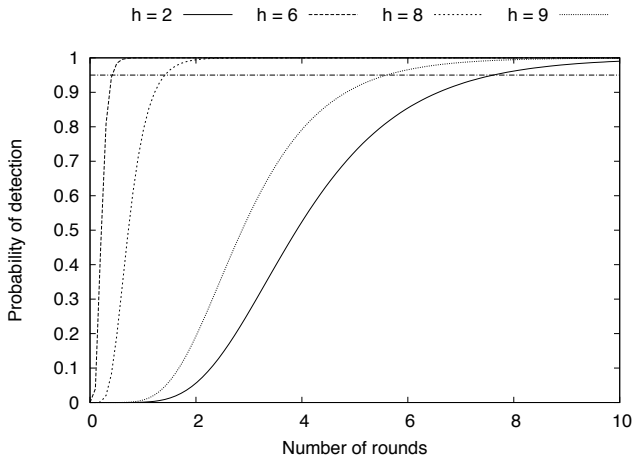

Figure 4.1: Probability of detection (p_d^*) as a function of the number of rounds (r)

If we consider the optimized version of SST (oSST), the total number of tests decreases substantially, since nodes can avoid testing combinations of h identities that have already been monitored when other nodes performed their tests. If, on each test, there are $0 < w \leq N - h$ tester nodes (the challenger node and the passive tester nodes), the function that describes $\mathcal{NT}$ becomes:

$$\begin{aligned} \mathcal{NT}(N, h, m, w, p_d^*) &= \mathcal{R}(N, h, m, p_d^*) \cdot \binom{N-1}{h} \cdot \frac{N}{w} \\ &= \mathcal{R}(N, h, m, p_d^*) \cdot \binom{N}{h} \cdot \frac{N-h}{w} \end{aligned} \tag{4.3}$$

This equation clearly shows the advantage of oSST, when configured with the maximum allowable value for w (*i.e.* $w = N - h$), in relation to the remaining RRTs (where $w = 1$), for the number of required tests:

$$\mathcal{NT}^{SST/SRT/FCT} = \mathcal{NT}^{oSST} \cdot (N - h)$$

4.4.2 Total Message Cost

The message cost of a RRT is defined as the total number of messages transmitted to complete the protocol. This metric is closely associated with the energy consumption in the

system due to execution of each RRT. The number of messages transmitted by each protocol ($\mathcal{MT}$) is simply the product between the number of tests $\mathcal{NT}$ and the number of messages transmitted on each test (mt):

$$\mathcal{MT}(N, h, m, w, mt, p_d^*) = \mathcal{NT}(N, h, m, w, p_d^*) \cdot mt \tag{4.4}$$

Table 4.2 parameterizes the $\mathcal{MT}$ function for each type of test, given the specificities of their operation. In SST, SRT and FCT the test is only carried by one tester at a time, $w = 1$. On the other hand, in the optimized version of SST (oSST) every node not being tested is testing the group ($w = N - h$). FCT can only test two nodes at a time ($h = 2$) and, thus, can not handle collusion ($m = 1$).

Test	Messages transmitted	Test parameters
SST	$\mathcal{MT}(N, h, m, 1, h, p_d^*)$	$w = 1,\ mt = h$
oSST	$\mathcal{MT}(N, h, m, N-h, h, p_d^*)$	$w = N - h,$ $mt = h$
SRT	$\mathcal{MT}(N, h, m, 1, 2, p_d^*)$	$w = 1,\ mt = 2$
FCT	$\mathcal{MT}(N, 2, 1, 1, 2, p_d^*)$	$h = 2$,w $= 1,$ $m = 1, mt = 2$

Table 4.2: Number of transmissions per test

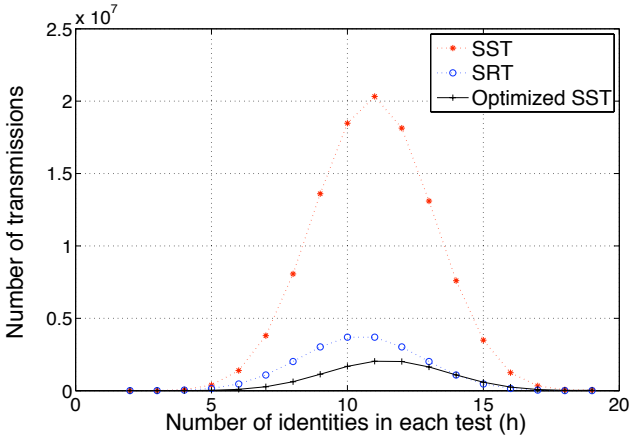

Figure 4.2: Total number of message transmissions ($\mathcal{MT}$) as a function of the number of simultaneously tested identities (h).

Figure 4.2 plots the functions in Table 4.2 as a function of h (FCT is not plotted because

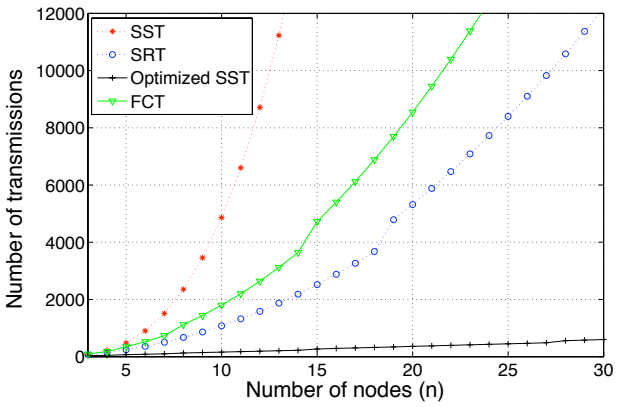

Figure 4.3: Total number of message transmissions ($\mathcal{MT}$) as a function of the number nodes (N).

it cannot handle $h \neq 2$), with $N = 20, m = 1$ and $p_d^* = 0.95$. All three plots show the same behavior. The number of transmitted messages is higher for intermediate values of h. Therefore, the number of simultaneously tested identities (h) should be either very low ($h = 2$) or very high ($h = N - 1$). However, the choice of h is also dependent on the maximum number of colluding nodes (m) that we want to tolerate, and the number of channels K available ($m < h \leq K$). The non-optimized version of SST is always worse than the remaining two, but the optimized version (oSST) is better than SRT for $h < \frac{2N}{3}$, and worse otherwise. Therefore, SRT is better for high collusion scenarios, and oSST is better for scenarios where fewer channels are available.

Figure 4.3 plots the $\mathcal{MT}$ in table 4.2 as a function of N with $m = 1$, $p_d^* = 0.95$ and $h = N-1$ for SRT, and $h = 2$ for the others (best case for each of them). FCT is always worse than the remaining tests. As expected, for $h = 2$, the optimized version of SST behaves better than the remaining tests.

4.5 Discussion

None of the proposed solutions is better than all the others in every scenario. Table 4.3 characterizes the scenarios where each solution performs better, when compared with other RRTs. The optimized version of SST (oSST) is the most adequate on scenarios with low and medium collusion and where there is no danger of denial-of-service attacks, because it requires the lowest number of messages of all RRTs. SRT is the best one for scenarios with high levels

Test	Best application context
oSST (SST)	Low collusion and no DoS threat
SRT	High collusion and/or DoS threat
FCT	One channel

Table 4.3: Best application context for each test

of collusion, or where denial-of-service attacks need to be taken into account, because it has the lowest resource cost level. Finally, FCT is best suited for scenarios where there is only one channel available, since all the other RRT require the simultaneous use of more than one channel.

One has to consider, however, the feasibility of testing a complete population of nodes. As depicted in Figure 4.3, even for a low collusion environment ($m = 1$) and a probability $p_d^* = 0.95$ of sybil detection, the number of transmissions quickly rises to the thousands. Further taking into consideration the possible increase of both these parameters, and the increase in the total number of nodes in the network, we quickly reach an unacceptable number of transmissions.

There is yet another disadvantage of testing the whole population of nodes: the fact that all the nodes joining the network have to be tested. This leaves the network vulnerable to denial-of-Service attacks, in which an attacker continuously presents new identities, with the sole intent of spending network resources. This vulnerability is further exacerbated in highly-dynamic networks, in which the entrance of many correct nodes can also contribute to a huge delay in the network operation. From this we conclude that the usage of radio resource tests should be reserved for a smaller group of network nodes, where the participants change rarely.

In the following chapters, this idea will be pursued further, with the inclusion of a second type of test prior to RRT execution, and the restriction of the universe to be tested to a subset (quorum) of the N nodes.

4.6 The NSQ-RRT Algorithm

This section presents and describes in more the implementation of the algorithm we chose for the NSQ-RRT, which will later be used as a building block for our NSQ algorithm. Both the oSST and the SRT could be used as a building block, given that the same abstractions still

apply, however, we choose oSST due to the fact that it allows $N - K$ testers, which is useful to increase the scalability of the NSQ protocol.

As previously described, oSST is a $RRT(K, 0, N - K)$, and is based on the assumption that no radio is able to simultaneously transmit on more than one channel ($A3$). The algorithm for the NSQ-RRT is depicted in Figure 4.4. The algorithm iterates through a fixed number of steps, given by the function "rrt.length()", to test a certain set of identities C. In each step, the node verifies if it is supposed to transmit in that specific step. If so, it transmits a VALIDATE message in the channel specified by the test scheduler, which is given by the function "rrt.schedule()". If not, it listens in a randomly chosen channel. Nodes that do not belong to C must listen in every step. If, while listening to a channel, a node detects radio silence (there was no transmission), the identity which was scheduled to transmit in that channel is added to the exclusion list. Note that the existence of a collision is interpreted as being a valid response. This is done to avoid an attacker from eliminating correct identities, by generating intentional collisions while they are being tested. No attacker will be able to exploit this fact to vouch for additional sybil identities, since if the attacker has an additional radio to generate a collision, it is also able to vouch for the sybil identity being tested.

```
// Executed at every node i
algorithm NSQ-RRT (C) is
    excluded ⟵ ∅;
    I ⟵ rrt.schedule(C);
    for (j = 0 to rrt.length (C)) do
        if id_i ∈ I[j] then
            broadcast.send (I[j].indexOf(id_i), VALIDATE);
        else
            channel ⟵ rand(1,K);
            if broadcast.receive (channel) = null then
                excluded ⟵ ∪ {I[j][channel]};
    return [sort(C \ excluded)];
```

Figure 4.4: NSQ-RRT Algorithm.

The function "rrt.length (C)" returns the total number of steps needed, and is used to iterate through all the necessary tests. The function "rrt.schedule(C)" returns a list of ids, which contains the identities that are supposed to transmit on each step of the test. We assume that the channels are assigned to the identities, from channel 1 to channel h, in lexicographic order.

The function "rrt.length()", can be trivially obtained, since it merely returns the total number of tests $\mathcal{NT}$ (Equation 4.3). The function "rrt.schedule()" is slightly more complex, and is shown in Figure 4.5.

algorithm rrt.schedule ($\mathcal{C}$) **is**
    $list \longleftarrow \emptyset$;
    for ids in $\binom{\mathcal{C}}{K}$ **do**
        for round = 0 to r **do**
            list $\longleftarrow$ list $\cup$ $\{ids\}$;
    return list;

Figure 4.5: RRT Schedule Algorithm.

The scheduling algorithm represented in Figure 4.5, essentially generates a list of every possible combination of identities in which every combination is repeated r times. Accessing index i of the list, we obtain the pairs (ids,channels), specifying the identities that are supposed to transmit on step i (ids), and their corresponding channel numbers.

The algorithm ends by returning a set with all the identities that passed the test, in lexicographic order.

4.6.1 Properties

4.6.2 Safety

Lemma 1: With a probability arbitrarily close to 1, the radio resource test limits the number of identities proposed by the m malicious nodes to m, if $m < h \leq K$, and there will be therefore, no sybil identities in the output of the RRT.

Proof sketch: This property of the RRT derives directly from the limitations imposed on radio devices, and the design of the RRT. Recall the following two assumptions: *A1)* each node possesses only one radio device; *A3)* no radio device can transmit simultaneously in more than one channel. From these two assumptions, we have that a set of m colluding malicious nodes, can only transmit simultaneously at most in m distinct radio channels. This way, and given that $m < h$, there will be tests in which the malicious nodes will only be able to vouch for at most m identities. If these tests are repeated an appropriate number of times (rounds), the probability

of detecting sybil identities can be arbitrarily close to 1. □

Lemma 2: No correct node's identity is excluded by the RRT.

Proof sketch: Assume the opposite, that the identity of a correct node is excluded from the test by another correct node, while executing the RRT. In our algorithm, an identity is only excluded when there is radio silence on the wireless medium, since a collision is considered as a valid transmission. Therefore, the node that owns the identity either stopped communicating due to a fault, or is defending some other identity. In either case, that node is considered not to be correct under the algorithm, which contradicts the assumption. □

4.6.3 Termination

Lemma 3: Assuming a finite set of identities to be tested, RRT terminates in a finite number of steps.

Proof sketch:

If the set of identities to be tested is finite, the number of possible combinations of h elements is also finite, and it will be possible to obtain a deterministic schedule for the RRT. The number of tests to be performed is, therefore, finite, and since the progression of the algorithm does not depend on any external condition, the RRT will thus terminate after a finite number of steps. □

Summary

In this chapter we further detailed the Radio Resource Tests, proposing a framework that captures the limitations of these tests, thus allowing the comparison of their performance. We have also proposed two novel RRTs and an optimization to a RRT previously proposed in the literature. Furthermore, we have analyzed these tests both in isolation, and when used to test an one-hop population. Finally, we detailed the implementation of the NSQ-RRT building block, providing proof of some relevant properties.

5 Computational Resource Tests

As was previously discussed, Computational Resource Tests (CRTs) have a specific type of resource test that relies on the assumption that network nodes possess limited computational power. This assumption is usually explored resorting to some sort of crypto-puzzle. In this chapter, we will analyze some crypto-puzzles and their properties. Furthermore, we will present a few previously proposed CRT tests and a new optimization of one of these tests. Finally, we provide a framework to analyze and compare CRTs, with a set of relevant metrics, finishing the chapter by detailing one of the computational resource tests, and providing proof of some of its relevant properties.

5.1 Crypto-Puzzles

A computational resource test is based on the assumption that nodes have restrictions on their computational power (Aspnes, Jackson, & Krishnamurthy 2005; Douceur 2002). These tests rely on the use of crypto-puzzles that are expensive to compute, yet not intractable (Back 2002; A. Juels 1999; Rivest, Shamir, & Wagner 1996). These puzzles allow to test the computational resource of network nodes, by requiring each new identity proposal to be accompanied by a solution to a moderately hard cryptographic puzzle. Since network nodes have limited computational power, this will effectively limit the number of sybil identities that each malicious node is able to propose in a given finite proposal period.

Cryptographic puzzles were first proposed by Merkle in public key protocols (Merkle 1978). Since then, puzzles have been applied in specific applications such as, for instance, authentication protocols (Ahn, Blum, Hopper, & Langford 2003; Aura, Nikander, & Leiwo 2001), e-mail protocols (Back 2002; Cynthia Dwork 1993), and generic network layer protocols (A. Juels 1999; Wang & Reiter 2003).

We define a crypto-puzzle as being a cryptographic challenge that requires the challenged

nodes to calculate a moderately hard puzzle, given some parameters. There are two parameters commonly present in the crypto-puzzles: the *work-factor*, that regulates the difficulty of each puzzle, and the *nonce*, a value usually given by the challenger node, that makes each crypto-puzzle unique. Note that, while the work-factor can be an *a priori* known parameter, the nonce is a dynamic value that changes with every puzzle.

We will now present some crypto-puzzles and discuss some of their properties.

5.1.1 Hash Reversal

Juels et al. (1999) proposed a crypto-puzzle approach, in which the challenged nodes are required to reverse a cryptographic hash calculated by the challenger, given the original random input (nonce) with b bits erased. The work-factor of this crypto-puzzle can be increased or decreased by varying b. The challenged nodes perform a brute-force search on the erased bits, by hashing each pattern in the input space until they find the answer. Note that both the generation and verification of a puzzle can be performed quickly, since they require a single hash, making the time taken to generate and verify a puzzle significantly small.

Multiple Hash Reversal In multiple hash reversal, the puzzle is divided into smaller hash reversal puzzles. This optimization was also proposed by Juels et al. (1999), and decreases the variance in the total solution time. Furthermore, using sub-puzzles of varying difficulty allows a finer control of the overall puzzle difficulty. On the other hand, the generation time of the puzzles increases, which represents a disadvantage, specially for high workloads.

5.1.2 Partial Hash Collision

Back (Back 2002) proposed a crypto-puzzle similar to the previously presented hash-reversal. In this puzzle, the challenged nodes are required to find a string (of unknown length), such that the concatenation of the nonce with that string, returns a hash in which the first b bits are 0. The work-factor of this crypto-puzzle can also be increased or decreased by varying parameter b. The challenged nodes perform a brute-force approach to the string, by hashing the concatenation of the nonce with a random value, until they find an answer.

In terms of performance, this crypto-puzzle is slightly better than the hash reversal scheme. While the verification of the puzzle can also be done by performing only one hash, the generation of the puzzle is only dependent on the creation of the random value (nonce), needing no cryptographic operations.

5.1.3 Time-Lock

Rivest et al. (1996) proposed a different kind of crypto-puzzle, introducing the concept of "time-release crypto". The objective is to encrypt a message so that it cannot be decrypted by anyone, until a pre-determined amount of time has passed. They created a crypto-puzzle whose fastest solution method is a sequential squaring process, forcing the solver of the puzzle to compute in a tight loop for a controllable amount of time. With time-lock puzzles, the challenger can decide the amount of time it wants the challenged node to spend solving the puzzle. The work-factor is measured in the number of squaring operations a challenged node needs to perform.

Time-lock puzzles constitute an attractive type of puzzle, since they require an exact, fixed amount of work. However, they consume a lot of resources on the challenger side, since they require the generation of two large prime numbers for each puzzle.

5.1.4 Properties

Low Variance Ideally, crypto-puzzles should take a predictable amount of computing resources to be solved. However, different puzzles will yield different results. We can divide the crypto-puzzles into two different types: fixed cost, and probabilistic cost.

A fixed cost crypto-puzzle takes a fixed amount of resources to compute. This means that the fastest algorithm for solving the puzzle will be a deterministic algorithm; for example the time-lock is fixed cost, since the number of squaring operations a node has to compute to solve the puzzle is known.

Probabilistic cost crypto-puzzles have a random solving time. There are two types of probabilistic cost puzzles: bounded probabilistic cost and unbounded probabilistic cost.

In the bounded probabilistic cost crypto-puzzles, there is an upper limit to the maximum number of computations a node may be required to do before finding the answer. The hash

reversal has a bounded probabilistic cost, since there is a well defined key space on which it has to find the solution for the puzzle (all the strings with b bits).

Finally, the unbounded probabilistic cost crypto-puzzles can, in theory, take forever to compute, even though the probability of taking significantly longer than the expected value may decrease rapidly towards zero. As an example, take the number of times needed to achieve a tail while throwing a fair coin; there is no bound to the required number of tries, but the probability of not getting a tail in z trials, tends towards 0 rapidly with increasing z. The partial hash collision crypto-puzzle also has a probabilistic unbounded cost, since the key space on which it has to find the solution for the puzzle is infinite (all the possible strings, with every possible size).

Comparing the previously presented types of crypto-puzzles, we have that the time-lock is the best to use in what concerns the variance of the amount of resources a puzzle takes to compute. The difference between the hash reversal and the partial hash collision is not so clear. Even though the hash reversal has a limit to the theoretical worst case running time, the practical difference between the two types, concerning the variance and typical running time is limited, as discussed in (Back 2002). Multiple hash reversal optimization can also be applied to the partial hash collision.

Interactivity Until now, we assumed that in all crypto-puzzles there was a challenger and a challenged node. The challenger essentially provided the parameters needed by the challenged node to solve the crypto-puzzle, and verified the correctness of the answer. However, this challenger may not be required, since the parameters (the nonce, for example), can be obtained in a non-interactive fashion.

Both the hash reversal and the time-lock are interactive crypto-puzzles. With the hash reversal, the challenger has to provide both the hash of the nonce and the value of the nonce with the final b bits erased, something that requires calculation to obtain. With the time-lock, the challenger needs to provide the challenged node with a multitude of parameters, and calculate two prime numbers to obtain them.

Using the partial hash collision, however, we can obtain either an interactive or a non-interactive crypto-puzzle. This is due to the fact that the only required dynamic parameter in the partial hash collision is the nonce. In an interactive setting, the challenger would give the

challenged node the nonce from which the puzzle should be calculated. However, one could use any commonly known parameter as nonce, like the identity of a node, the current time, or a nonce computed cooperatively, as is the case of the nonces that will be presented in Chapter 6. This eliminates the need for an explicit challenger, and creates a non-interactive crypto-puzzle.

Known Solution A crypto-puzzle is said to have a known solution, if the challenger knows the answer to the puzzle. Both the hash reversal and the time-lock are known solution puzzles. In the case of the hash reversal, the challenger knows the value of the b bits that are removed from the original nonce, on which the challenged node needs to perform brute-force. Similarly, in the time-lock, the challenger node knows the message that the challenged node is required to decipher.

However, the partial hash collision is an unknown solution crypto-puzzle, since the challenger only provides a nonce to the challenged node, and doesn't know the answer to the puzzle (the additional string that concatenated with the nonce gives a hash with the first b bits set to zero).

Public Auditability The solution of a publicly auditable crypto-puzzle can be efficiently verified by any third party, without access to secret information. Note that this property means implicitly that the crypto-puzzle is efficiently publicly auditable, since all crypto-puzzles are publicly auditable by definition, as the auditor can just repeat the work done by the challenged node.

Both the hash reversal and partial hash collision have this property. In the case of the hash reversal, a reply to the crypto-puzzle that contains: the original hash; the first bits of the nonce; and the last b bits missing, can be verified by any node only listening to the answer. In a similar way, in the case of the partial hash collision, a reply to a crypto-puzzle that contains the nonce and the answer string can also be easily verified by any node. However, in the case of the time-lock, no other node but the challenger has the original message that was encrypted and sent to the challenged node, nor the private key used to decipher it. Therefore, no other node is able to determine if the answer is correct.

Public Trust While public auditability is an important property for a crypto-puzzle, it can only be useful if the nodes in a network can trust the answer. Imagine that two nodes A and B

are testing each other using a hash reversal crypto-puzzle. If the nodes are malicious, and since the hash reversal is a known solution crypto-puzzle, the malicious nodes can just exchange the answers, and easily create a valid crypto-puzzle answer.

One might think that the usage of the partial hash collision crypto-puzzle is the answer to this problem, since it is an unknown solution crypto-puzzle. However, while the malicious nodes in this case do not know the answers, they can simply pre-compute as many valid crypto-puzzle answers as they want.

The fundamental problem lies with the nonce. The nonce guarantees the freshness and randomness of the crypto-puzzle, avoiding replay and/or pre-computation attacks on the puzzles. If the nonce is known *a priori*, all the malicious nodes are able to pre-compute multiple answers to the puzzle, defeating the whole purpose of the CRT.

This means that, in our environment, the only publicly trusted crypto-puzzle is one with an unknown solution, based on a trusted (randomly generated, and previously unknown) nonce. A nonce with these characteristics will be called a *correct nonce*.

5.2 Framework Definition

In this section we will describe three different Computational Resource Tests that have the objective of limiting the number of sybil identities proposed to a network. We will present two previously existing CRTs, and a new optimization. All three CRTs use crypto-puzzles with an unknown solution, more specifically, the partial hash collision crypto-puzzle. This fact will be crucial in our optimization, to be able to obtain a publicly trusted crypto-puzzle.

To capture the limitations of the CRTs, and ease the comparison between them, we characterize the $CRTs$ by a set of parameters $CRT(h, c, w)$ as follows. Parameter h is the size of the set $S = \{s_1, s_2, ..., s_h\}$ of distinct identities that can be tested simultaneously, in a single test. Parameter c is the number of *challenger* identities (not in S) that need to actively participate in the test. Parameter w is the number of nodes that can extract information from the test.

Note that, while our focus will be on CRTs with unknown solution crypto-puzzles, this framework can be applied to any kind of CRT.

5.2.1 Democracy

Aspnes et al. (2005) proposed the use of a CRT called Democracy, as a preamble for any standard byzantine agreement algorithm. They create a virtual network, where identities are priced by computing power, so that consensus algorithms can safely run. This technique solves the byzantine agreement problem if the adversary controls less than a third of the total computational power in the system, and in the specific case where all machines have equal computational power.

The Democracy algorithm takes three rounds to validate identities. In the first round, each node sends the individualized crypto-puzzle parameters to every other node. In the second round, all nodes computes as many solutions as possible in a given time, and send the puzzles and its solutions back to the challenger. In the final round, the challenger verifies the received solutions and assigns the correct number of identities to that node.

If we analyze each test individually, it is easy to classify Democracy as a $CRT(N-1,1,1)$ in our framework. First, the number of simultaneously tested identities (h), is the total number of existing identities in the network, minus the identity of the challenger ($N-1$). Secondly, and since the crypto-puzzle is interactive, the number of active challengers in this test is one. Finally, the fact that in each test there is a challenger providing the parameters to be used, makes the crypto-puzzles not publicly trusted, and only the challenger is able to trust its correctness.

5.2.2 Hashcash

Back (Back 2002) proposed the use of a CRT as a mechanism to throttle systematic abuse of unmetered internet resources such as email. This CRT works by requiring mail clients to solve a crypto-puzzle before being able to send an email. The parameters of the crypto-puzzle are given by the service-name, which can be any bit-string that uniquely identifies the service (e.g. host name, email address, etc).

While originally being proposed to be used in a client-server fashion, and to stop spam e-mail, Hashcash can be easily adapted to apply to a distributed model and to hinder the entrance of sybil identities in a network. If we assume that each node has an identifier, similar to the described service-name, and that each node acts both as a client and a server (all nodes calculate puzzles for every other nodes' identity), we can have the Hashcash CRT working in a distributed

way. Note, however, that while in Democracy each node replies to the crypto-puzzles sent by a challenger node, in Hashcash, each node takes the initiative and sends a crypto-puzzle answer to other nodes, parametrized with their identities.

This way, Hashcash becomes a $CRT(N-1,0,1)$. The number of simultaneously tested identities (h) is the total number of existing identities in the network (excluding the identity currently solving the puzzle), which is $N-1$. The number of challenger nodes is 0, given that the crypto-puzzle is non-interactive (the parameters of the puzzles are given by the bit-string that uniquely identifies every node). Finally, the number of testers in this test is also one, since the crypto-puzzle used, while publicly auditable, is not publicly trusted (since the answers can be pre-computed by malicious nodes).

5.2.3 Trusted Hashcash

The Democracy and Hashcash CRTs possess a few disadvantages. The fact that Democracy is based on an interactive crypto-puzzle creates a huge number of unnecessary transmissions, since the challengers have to distribute the puzzle parameters to all the other nodes. While Hashcash does not posses this disadvantage, the fact that it is non-interactive and uses the identities of the nodes as the nonce for the crypto-puzzle, makes it vulnerable to a pre-computation attack, in which an attacker calculates multiple answers for every node's identity, being thus able to propose multiple sybil identities. Additionally, both Hashcash and Democracy require every node in the network to solve one crypto-puzzle for every other node in the network, since the puzzle answers are not publicly trusted.

Trusted Hashcash is the name of our optimized computational resource test. This test was created with the main objective of possessing a publicly trusted crypto-puzzle, so that identities are able to trust every valid crypto-puzzle reply, and thus avoid the explosion of the number of necessary puzzles. In this CRT, we were also able to remove the other referred disadvantages of Hashcash and Democracy.

In Trusted Hashcash we assume the existence of a correct nonce, that becomes at some point known by every node. This assumption will allow us to have a publicly trusted, non-interactive crypto-puzzle that is not vulnerable to pre-computation attacks. We will describe in Chapter 6 two distinct methods to create such a commonly known correct nonce in a distributed fashion.

When the correct nonce becomes available, every node will have a given time period to calculate a non-interactive partial hash collision crypto-puzzle, the crypto-puzzle parameter being given by the concatenation of the known correct nonce and the node's own identity. As soon as a node calculates the crypto-puzzle, it broadcasts the solution.

This Trusted Hashcash test is, therefore, a $CRT(N, 0, N-1)$. The number of simultaneously tested identities will be N, since all the identities in the network will be tested simultaneously. The number of challenger nodes will be zero, since the test is non-interactive. Finally, the number of testers in this test will be $N - 1$, since the crypto-puzzle is publicly trusted, and, therefore, every node will be able to trust the answers of other nodes.

5.3 Analysis

5.3.1 Message Complexity

To analyze the performance and scalability of the presented CRTs, the relation between the number of sent messages and the number of nodes in the network must be obtained.

We will now analyze the message complexity of the three presented CRTs.

Democracy As previously seen, Democracy is a $CRT(N-1, 1, 1)$. In Democracy, every node sends an individualized crypto-puzzle to every other node in the network, since the crypto-puzzle used is interactive. Additionally, and since we have $w = 1$, only the challenger will be able to extract information from the test, which means that there will be N different combinations of tests (each of the N network nodes acts once as a challenger). Therefore, since in every test there will be $(N-1) \cdot 2$ messages sent, we will have a total of exchanged messages of $N \cdot (N-1) \cdot 2$. From this, we have that the complexity of the Democracy algorithm in terms of messages is $O(N^2)$.

Hashcash Hashcash is a $CRT(N-1, 0, 1)$. The main difference between this CRT and Democracy, is the fact that the crypto-puzzle is non-interactive ($c = 0$), and, thus, will only require $N - 1$ messages per test. In Hashcash, we also have $w = 1$, which means we will have to repeat it N times, once for every node. Therefore, we can easily see that the number of messages sent is given by $(N-1) \cdot N$, which also has a $O(N^2)$ complexity.

Trusted Hashcash The Trusted Hashcash CRT is a $CRT(N, 0, N-1)$. Besides having a non-interactive crypto-puzzle ($c = 0$), the number of testers is the maximum ($w = N - 1$), which will greatly reduce the number of necessary messages. In this CRT, the only message is the reply to the crypto-puzzle, and this reply will be trusted by every other node (given the existence of a correct nonce). From this, we have a total of N exchanged messages, and thus, a $O(N)$ complexity.

5.3.2 Sybil Mitigation

The discussed CRTs have a time period (timeout) in which the nodes have to answer the crypto-puzzle. This is necessary, in order to prevent malicious nodes from solving an infinite number of puzzles. This timeout period is, naturally, dependent on the crypto-puzzle being used, since, as seen earlier, crypto-puzzles may have different levels of variance. We will, however, focus on the analysis of probabilistic-cost crypto-puzzles, being the kind of puzzle used by the described CRTs.

The definition of the timeout value will always imply a difficult tradeoff, due to the randomness of the time taken by a node to solve one puzzle through brute-force: we wish to establish a timeout that, with a probability arbitrarily close to 1, accepts a minimum of replies from correct nodes, but also, hinders byzantine nodes from being able to solve more than one puzzle. These two objectives may not be compatible.

To solve the cryptographic puzzle of any of the three previously described CRTs, each node has to find a chain of characters that, concatenated with the nonce, outputs a hash in which the first b bits are zero (as stated, we are only considering the partial hash collision crypto-puzzle). This method was seen earlier when we described the partial hash collision crypto-puzzle. Assuming a perfect hash function, the probability of a node being able to solve ν crypto-puzzles in T tries, is given by a binomial distribution:

$$p(\nu) = \binom{T}{\nu} (p_{te})^{\nu} (1 - p_{te})^{(T-\nu)}, \tag{5.1}$$

where $p_{te} = (1/2)^b$ is the probability of the resulting hash having the first b bits set to zero, for any randomly character chain.

To validate this conclusion using a standard hash function, we created a program in python,

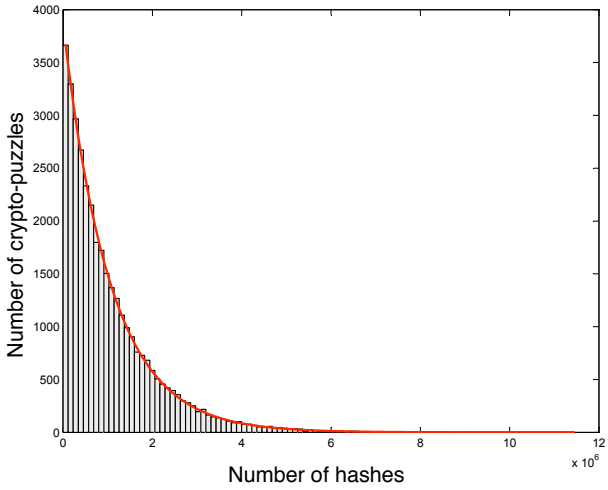

Figure 5.1: Distribution function and histogram, with 100 classes, of the simulated crypto-puzzle answers.

to simulate partial hash collision crypto-puzzles, saving the number of hash operations needed to solve each one of the 350000 different randomly generated crypto-puzzles, with $b = 5$. The results are shown in Figure 5.1, where we can see both the resulting histogram, with 100 classes, and the analytic solution (Equation 5.1). The simulations were done in a computer with a 2.33GHz Pentium processor, and with 8GB of RAM.

Let us now assume that we have a network with N nodes, f of them being byzantine, and that we want to allow at least $f + 1$ correct nodes to be able to reply to the crypto-puzzle. The probability p_c that all the $N - f$ correct nodes are able to propose at least $f + 1$ identities is given by

$$p_c = \sum_{i=f+1}^{N-f} \binom{N-f}{i} (1 - p(\nu = 0))^i (p(\nu = 0))^{(N-f-i)}. \tag{5.2}$$

On the other hand, the probability p_{sb} that the f byzantine nodes are able to propose at least one sybil identity is given by $p_{sb} = 1 - (p(\nu = 0) + p(\nu = 1))^f$.

Both these equations are represented in Figure 5.2, for N=50, f=5, and b=20, by varying the maximum number of tries that each node has to propose an identity. As can be seen, to achieve a probability near 1 that 6 $(f + 1)$ or more correct identities are proposed, we have to allow a probability of almost 0.2 of sybil identities being proposed. These numbers do not

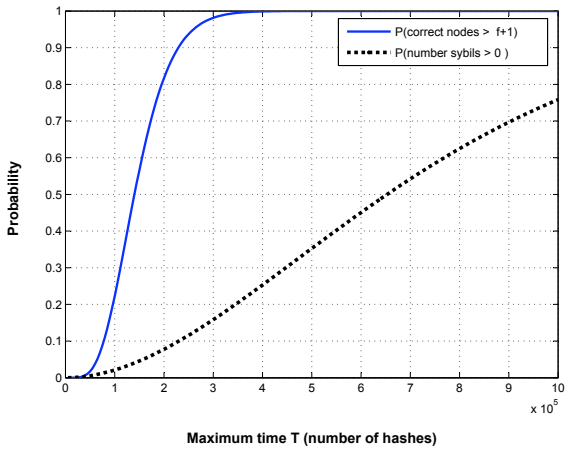

Figure 5.2: Probability of correct and sybil identities entering the network, for $N = 50$, $f = 5$, $w = 20$.

change by varying b, but change significantly by changing the number of byzantine nodes f.

This means that, to increase the capability to collect an adequate number of responses from correct nodes, we will unavoidably increase the probability of having sybil identities proposed. This problem could be avoided by using a time-lock crypto-puzzle, since that would allow perfect control of the time taken to solve a puzzle. However, since time-lock crypto-puzzles are not publicly auditable, the use of multiple hash reversal optimization seems preferable, due to the fact that it allows a decrease on the variance of the computing times, while maintaining public trust. We can try to decrease the variance of the results by using, for example, the multiple hash reversal optimization on our crypto-puzzle. However, this optimizations will decrease the performance of the solution and will never be able to completely avoid the entry of sybil identities.

To obtain the probability distribution of the total number of identities that the f byzantine nodes will be able to propose, one has only to realize that this total number of identities is the sum of the number of identities proposed by each byzantine node. It is, thus, a random variable which results from the sum of independent binomial random variables with the same individual probability of success (p_{te}). This sum is, therefore, also binomial, with the same p_{te} and a total number of tries equal to $f \cdot T$ (Ibe 2005). The probability of having up to s total identities proposed by the f byzantine nodes ($p_{ILT}(s)$) thus becomes

$$p_{ILT}(s) = \sum_{i=0}^{s} \binom{f \cdot T}{i} p_{te}^{i} \cdot (1 - p_{te})^{(f \cdot T - i)}. \tag{5.3}$$

5.4 Discussion

Even though, as seen, CRTs cannot completely avoid the entry of sybil identities they play an important role, since they severely limit the number of such identities. They are, thus, very useful as a first step towards sybil mitigation. Also, they possess high performance and scalability (something that Radio Resource Tests do not possess), and are, thus, usable in dense networks.

The fact that they are fallible depends on the stochastic nature of the times needed to solve the used crypto-puzzle. Has as been already discussed, this problem could be avoided by using a time-lock crypto-puzzle. However, time-lock crypto-puzzles are not publicly trusted, and therefore, as was seen, score worse on scalability. Scalability is, however, an important feature that we wish to maintain. Therefore the use of Trusted Hashcash, possibly with the multiple hash reversal optimization, constitutes a preferable option.

5.5 The NSQ-CRT Algorithm

This section presents and describes in more detail the NSQ-CRT, which will later be used as a building block for our NSQ algorithm. Even though any of the CRTs could easily be used as the NSQ-CRT, given that the same abstractions still apply, we chose Trusted Hashcash due to its better performance.

The NSQ-CRT has the following interface: a node has to invoke crt.solve(nonce,id) to solve the crypto puzzle, taking the nonce and its own identity as input parameters; this function returns the response to the crypto-puzzle. There is also a method crt.verify(nonce,id, a) that allows any node to validate an answer (a) to the crypto-puzzle. We assume that a response can be verified in a single step.

In Figure 5.3, the algorithm for the NSQ-CRT is presented. This algorithm uses two tasks. The first task is used to solve the crypto-puzzle and return the solution in the variable a. The

other task is used to receive the answers from the other nodes or, if the solution to the crypto-puzzle has already been discovered, to send the crypto-puzzle answer. A PROPOSAL message includes the identity of the node replying to the crypto-puzzle and the respective solution. When a node receives a crypto-puzzle answer, it validates the solution. If the answer is valid, the node adds that identity to the identity list. In this algorithm, p_t is used to randomize the transmissions of nodes possessing the solution to the puzzle.

As was discussed earlier, there needs to exist a time period (timeout) in which the nodes have to answer the crypto-puzzle. This timeout is chosen to allow, with a probability arbitrarily close to 1, the existence of a minimum of replies from correct nodes. This minimum number of replies from correct nodes, is given by $q - f$, where q is a parameter to the CRT that gives the total number of identities we wish our quorum to possess. However, and as we can see in the algorithm, besides the timeout period, there is another condition that allows the algorithm to terminate and return the proposal list: receiving σ proposals. This can be seen as an optimization for the algorithm, that allows the nodes not to wait for the timeout period to expire, if enough proposals have already been received. Note that σ has to be chosen carefully, since there is the possibility that the malicious nodes succeed in filling the whole proposal list with sybil identities. If we require at least $q - f$ correct identities in the output set, we should use $\sigma = (q - f) + s$, with s being the maximum number of crypto-puzzles that, with a probability arbitrarily close to 1, the byzantine nodes are able to solve (Equation 5.3).

Although our algorithm is not tied to a specific crypto puzzle, for self containment, we describe more thoroughly the partial hash collision crypto-puzzle used in NSQ-CRT. Given a one-way hash function $\mathcal{H}$ that returns a bit string, and an operator $\mathbf{left}_b(s)$ that returns a sub string composed by the b leftmost bits of s, the crypto puzzle can be defined as follows:

$$crt.solve(nonce, id) = \{a \in \Re : left_b(\mathcal{H}(nonce||id||a)) = 0^b\}$$

$$crt.verify(nonce, id, a) = \begin{cases} true, \ \mathbf{left}_b(\mathcal{H}(nonce||id||a)) = 0^b \\ false, \ \mathbf{left}_b(\mathcal{H}(nonce||id||a)) \neq 0^b \end{cases}$$

where a is the solution for the crypto puzzle and b is the crypto-puzzle work-factor that controls the difficulty of the puzzle. When solving the crypto puzzle, a node has to find a bit string a such that the hash resulting from the concatenation of the nonce, the identifier of the node, and

```
// Executed at every node i
algorithm NSQ-CRT(nonce,q) is
    C ⟵ ∅;
    done ⟵ 0;
    σ ⟵ number of required proposals;
    t ⟵ 0;
    Task 1: // Crypto-puzzle resolution
        a_i ⟵ crt.solve(nonce, id_i);
        t ⟵ 1;

    Task 2: // Answer verification
        while |C| ≤ σ | timeout do
            if t = 1 and rand() < p_t and done = 0 do
                if broadcast.send(id_i,a_i) ≠ collision do
                    C ⟵ C ∪ {id_i};
                    done ⟵ 1;
            else
                m ⟵ broadcast.receive();
                if m = (PROPOSAL) do
                    if crt.verify(nonce,m.id, m.a) do
                        C ⟵ C ∪ {id};
        return C;
```

Figure 5.3: NSQ-CRT Algorithm.

a is a bit string with b zeros as the leftmost bits. Note that, since we assume that the nonce is a commonly known value, we cannot simply request the string that, concatenated with the nonce, would output the desired hash, since all nodes would then have the same puzzle solution, and malicious nodes could take advantage of that to propose multiple sybil identities. We thus use the nodes' own identity to parametrize the crypto-puzzle, making a truly unique puzzle for every distinct identity.

The fastest known algorithm for computing such a *partial collision* for a one-way hash function is brute force (Pointcheval & Stern 2000). It should be noted that the puzzle is not interactive, since it is assumed that the nonce becomes globally known in some previous phase, and id is the identity of the node solving the puzzle.

Solutions can be easily verified by computing the hash with the answer provided by the nodes, and verifying if the b leftmost bits are zeros. Additionally, our crypto puzzle is publicly trusted, since the answer to the puzzle can be independently verified and trusted by every participant.

5.6 Properties

5.6.1 Safety

Lemma 4: Assuming the existence of a correct nonce, the CRT limits the maximum number of identities that, with a probability arbitrarily close to 1, f malicious nodes are able to propose.

Proof sketch: This property derives from the fact that the proposal period is finite. Given a finite proposal period, there is a maximum value (s) of identities that with probability arbitrarily close to 1, the malicious node will be capable of proposing (Equation 5.3). Since the number of sybil identities is given by $s - f$, the Lemma follows. □

5.6.2 Output Consistency

Lemma 5: Assuming the existence of a correct nonce, there will be at least $q - f$ correct identities in the CRT output, with a probability arbitrarily close to 1.

Proof sketch: The algorithm finishes by one of two possible reasons. Either the timeout has been exceeded or $\sigma = (q - f) + s$ valid proposals have been received. In the first case, if the timeout has been chosen according with Equation 5.2, the number of correct replies will have been received with the chosen probability (p_c). In the second case, since s is (again, with a probability arbitrarily close to 1), the maximum number of identities proposed by malicious nodes, it follows that the $q - f$ remaining identities belong to correct nodes. □

5.6.3 Termination

Lemma 6: Assuming the existence of a correct nonce, the CRT terminates in a finite number of steps.

Proof sketch: Since there is a maximum timeout, and the progression of the algorithm does not depend on any external condition, in the worst case, the CRT returns after a finite time period. □

Summary

In this chapter we further detailed the Computational Resource Tests, introducing the concept of crypto-puzzle, and presenting some different kinds of crypto-puzzles and their properties. We also utilized the same framework from Chapter 4, to capture the limitations of CRTs, allowing us to make a comparison between them. Furthermore, we presented an optimization to an existing CRT, and detailed the implementation of the NSQ-CRT building block, providing proof of some relevant properties.

6 Nonce Generation

As seen earlier, crypto-puzzles depend on some initial nonce. The goal of our nonce generation algorithm is to create a random string of bits that depends on the input from n identities. In this way, we create a commonly known random value in a completely distributed manner, allowing nodes to trust the outcome as correct (e.g. not influenced by byzantine nodes). A correct nonce must be collaboratively generated, random and previously unknown. In this chapter we will present and analyze two distinct nonce generation algorithms.

6.1 Algorithms

Our correct nonce generation problem is somewhat similar to the distributed group key management problem (Rafaeli & Hutchison 2003). In distributed key management, nodes have to agree on a common group key (or keys) to be used for the encryption of the communication. The distributed key management is characterized by having no group coordinator, and being done in a completely distributed fashion. The group key can be either generated in a contributory fashion, where all members contribute their own share to the computation of the group key, or generated by a single participant. Note that solutions that assign greater importance to some group members (the latter case), are undesirable, since privileged members might behave maliciously; furthermore, they are also attractive targets for attacks.

The peer-to-peer nature of both the key management, and our problem results in some common properties. First, every member is both a sender and a receiver. Second, there is no hierarchy, and all members enjoy the same status. However, our solution focus on two particular properties of the shared values: their randomness and freshness; these properties are common to some key management algorithms (Becker & Wille 1998; bo Guo & feng Ma 2003).

These key distribution techniques have some disadvantages that make them inappropriate for our environment. For example, in (Becker & Wille 1998), the authors propose the octopus

protocol, in which N nodes are split into four subgroups ($\frac{N}{4}$ members each), and a leader is elected in each subgroup, being responsible for collecting all contributions, from all its subgroup members, and calculating an intermediate value. This dependency on privileged members is undesirable. Other examples of key distribution protocols that depend on privileged nodes can be seen in (Rafaeli & Hutchison 2003). Another disadvantage of such an approach, is the vulnerability to byzantine behavior. For example, all the solutions based on the Diffie-Hellman protocol (Perrig ; Steiner, Tsudik, & Waidner 1996; Kim, Perrig, & Tsudik 2000; Becker & Wille 1998), require an *a priori* agreement on two prime numbers p and q. However, they assume that the participant nodes are correct nodes, and thus, that an agreement will always be possible, something that may not be true. Furthermore, if voting schemes are used to reach this agreement, these protocols become vulnerable to the sybil attack.

Additionally, since our problem has weaker properties, and needs to cope with collisions in the communication channel, the solutions used for the key management will be less efficient (due to the use of many cryptographic operations), and some of them, that assume fully reliable channels, may not work at all. This motivates the need for algorithms that are able to guarantee the generation of correct nonces in a completely distributed fashion.

6.1.1 With collision detection

This algorithm iterates over a sequence of communication steps. In each step, a node either attempts to provide an input to the nonce or just listens to other proposals. When two or more nodes attempt to contribute to the nonce in the same step, a collision is generated. As previously described in Chapter 3, we assume that, whenever a collision happens in the wireless medium, all nodes, including the senders, are able to detect it. Furthermore, when a collision happens, no node is able to receive any message.

The nonce generation algorithm is simple. It operates by sequentially collecting contributions from participants, in the order with which they are announced. The algorithm ends as soon as the set of collected contributions reaches a target size n.

Initially, the nonce has no contributions. In each step, each node that has not yet added its own identifier to the set might try to broadcast its contribution (the identifier, followed by random bit string). To avoid unnecessary collisions nodes only transmit with a given probability

p_t. If a node transmits its contribution and no collision is generated, the node adds its own identifier, followed by the random bit string, to the nonce set.

Nodes that do not try to add their own identifier in a step, will listen to the wireless medium, for other node's advertisements. If a node receives an advertisement from another participant, it will simply add the advertised contribution to the nonce set. Notice that this can be done since we assume that, if a node receives a message, then all nodes also receive the message (we are assuming that no omissions may occur). It is possible that, at a given step, no valid advertisements are made. This could happen as all nodes can decide not to transmit their contribution in a given step, or because a collision happened.

This mechanism ensures that all (correct) participants will compute the same nonce set, therefore, being able to return the same value deterministically (by resorting to a one-way hash function such as SHA-1 (Eastlake & Jones)).

6.1.2 Without Collision detection

Since the previous algorithm requires the existence of collision detection, we now present a different nonce generation algorithm that does not.

As before, this algorithm iterates over a sequence of communication steps, and in each step a node either transmits or just listens to the wireless medium. However, instead of a node trying to contribute to the nonce with it's own identifier, in this algorithm the contribution to the nonce is simply the occupation of the wireless medium in that step.

Every node keeps a bit sequence with the steps that were occupied, and those who were idle (with values 1 and 0, respectively). It should be noted that the transmission of more than two nodes simultaneously (a collision) is regarded as a normal transmission. This removes the need for the collision detector, since if a node transmits, there is necessarily at least one transmission in the wireless medium, and that slot is thus marked as occupied by every node.

Initially, the nonce is an empty set. In each step, each node either transmits, with a given probability p_t, or listens to the wireless medium. In the first case, a value of 1 is added to the nonce, given that, independently of the behavior of the other nodes, there is at least one transmission in the medium. In the second case, the node listens for transmissions from other

nodes. A 0 is added to the nonce if the medium is idle during that step, and a 1 is added otherwise.

This mechanism guarantees that all correct participants will converge to the same nonce set, thus being able to return the same deterministic value (by resorting to a one-way hash function such as SHA-1 (Eastlake & Jones).

6.2 Analysis

In both algorithms, although all nodes agree on the same nonce, if faulty nodes have no resource limits (i.e., they can send an arbitrary number of messages) it is possible that the resulting nonce is completely defined by the faulty nodes. Such a of nonce is not acceptable. A correct nonce must be random and previously unknown (or predictable).

This is possible because, without resource restrictions, faulty nodes may attempt to communicate in every step. Thus, faulty nodes either succeed to include the desired contribution or cause a collision (preventing correct nodes from provide contributions to the nonce). This would allow the byzantine nodes to determine the nonce, thus enabling pre-computation of the crypto-puzzle. We will assume that all nodes have such a resource restriction.

We will now consider the number of steps needed to guarantee the correctness of the output of the nonce, for both algorithms, given a limitation on the number of transmissions a node is able to perform.

6.2.1 With collision detection

In the case of the algorithm with collision detection, and since the contribution of the nodes is done with the identity of the contributing nodes, as well as a random bit string, it is enough to guarantee that at least one correct node is able to participate in the nonce generation for it to be correct.

Assume a network with N nodes, in which every node possesses a bound on the number of transmissions they are able to perform c_i, in a given number of steps ($\mathcal{P}$). This assumption effectively limits the number of intentional collisions f malicious nodes are able to generate to $c_i \cdot f < \mathcal{P}$. For a correct node to be able to contribute successfully with it's own identifier, we

have to take into account both the probability of collision between contributions from correct nodes, and the intentional collisions faulty nodes may generate. Therefore, we need to obtain the minimum number of communication steps require to achieve an arbitrarily high probability that at least one of the $N - f$ correct nodes will be able to contribute successfully, thus guaranteeing a correct algorithm outcome.

By hypothesis, the byzantine nodes are able to generate at most $c_i \cdot f$ collisions in $\mathcal{P}$ steps, while the correct nodes will generate non-intentional collisions that depend on the frequency with which they try to transmit. Considering p_t as the probability of a correct node trying to transmit in each step, we can obtain the probability of only one of the $N - f$ correct nodes trying to transmit in a given step p_s:

$$p_s = p_t \cdot (1 - p_t)^{N-f-1} \cdot (N - f).$$

We can now obtain the probability that the nonce generation succeeds in having a contribution from a correct node, in a number of steps $\mathcal{S}$:

$$p_{sp} = 1 - (1 - p_s)^{\mathcal{S}}.$$

Note that p_{sp} can have any desired value. By solving $\mathcal{S}$ in order to p_{sp}, and adding the number of slots where faulty nodes can create collisions, we obtain the minimum number of slots ($\mathcal{MS}$) needed for the nonce generation algorithm to succeed

$$\mathcal{MS} = \frac{log(1 - p_{sp})}{log(1 - (p_t \cdot (1 - p_t)^{N-f-1} \cdot (N - f)))} + f \cdot c_i. \tag{6.1}$$

The number of transmission steps (S) must, therefore, be greater than $\mathcal{MS}$ and lower than $\mathcal{P}$.

6.2.2 Without collision detection

In the case of the algorithm without collision detection, the randomness comes exclusively from the set containing the occupied and idle steps. Therefore, and contrarily to what happened in the previous example, instead of only needing one contribution from a correct node, we have

to guarantee that there are enough steps (bits) in the nonce contributed by the correct nodes, to avoid the possibility of pre-computation of the nonce. Bits contributed by correct nodes will be called hereafter as *correct bits*.

As before, assume a network with N nodes, where each of the network nodes possesses a bound on the number of collisions c_i it can generate in a given number of steps ($\mathcal{P}$). We have that, the maximum number of intentional collisions f byzantine nodes are able to generate is given by, $c_i \cdot f < \mathcal{P}$. For the output of the nonce to be correct, we have to take into account both the collisions caused by byzantine nodes and the size of the remaining number of steps. We will assume that n correct bits are enough to avoid pre-computation attacks.

Therefore, adding the number of steps needed for the n correct bits, to the number of intentional collisions byzantine nodes may generate, the minimum number of slots $\mathcal{MS}'$ needed for the nonce generation to succeed is given by:

$$\mathcal{MS}' = n + f \cdot c_i. \tag{6.2}$$

The number of transmission steps (S) must, therefore, be greater than $\mathcal{MS}'$ and lower than $\mathcal{P}$. Note that if the probability of the nodes transmitting in a given slot, is greater than not transmitting, the assumption that n steps are enough to provide n random bits to the nonce is not necessarily true, since the value 1 will be more common than the value 0. We need, therefore, to increase the amount of randomness (entropy) of these n correct bits, thus guaranteeing a truly random nonce outcome. The probability of each one of these bits being zero (p_{idle}) should, thus, be $\frac{1}{2}$, so that each step possesses the maximum value of entropy.

This probability of an idle step can be related to the individual probability of transmission p_t as follows:

$$p_{idle} = (1 - p_t)^{N-f}.$$

Solving this equation for p_t, we have

$$p_t = 1 - \sqrt[N-f]{p_{idle}}.$$

This p_t can be used to guarantee that the probability of an idle step is equal to the probability of a occupied step ($\frac{1}{2}$), and thus, that each step provides a truly random bit.

6.3 Discussion

Even though the nonce generation algorithm without collision detection has the obvious advantage of not requiring a collision detection mechanism, one has still to consider the number of steps needed for each of the algorithms to finish.

To be able to make a comparison between the algorithms, both of them need to have the same degree of security. More specifically, we need to ensure that both of them have the same resistance to the guessing attack. In the guessing attack, a malicious node tries to guess the value of some unknown parameter. The nonce can be seen as an example of such a parameter. This attack can be addressed by guaranteeing that the domain from which the parameter is chosen is large enough, being therefore unfeasible for the malicious node to guess its value. Furthermore, there must be no easier way to guess the nonce value, for example, by guessing the node's contribution for the nonce, instead of the nonce itself.

With collision detection In the algorithm with collision detection, the randomness comes from the content of the contribution of the nodes (the node's identity and the random value), and the place in the nonce set where the contribution appears. The most straightforward way of increasing the difficulty of guessing this contribution, is by increasing the size of the random value. We assume that one contribution from a correct node is enough to provide a sufficient number of random bits, such that the difficulty of guessing one contribution is higher than the difficulty of guessing the nonce.

The minimum number of steps ($\mathcal{MS}$) to guarantee, with a configurable high probability (p_{sp}), that the nonce has the contribution of at least one correct node, is given by Equation 6.1. This equation results from the addition of two elements: the number of steps occupied by byzantine nodes transmissions, and the number of steps required by the non-byzantine nodes to contribute to the nonce, with a probability p_{sp}. Figure 6.1, depicts the number of steps required by the latter element, in order to the probability of a correct node transmitting in each step (p_t), for a network with $N = 5, f = 2$, and a probability of success of $p_{sp} = 0.95$. The function

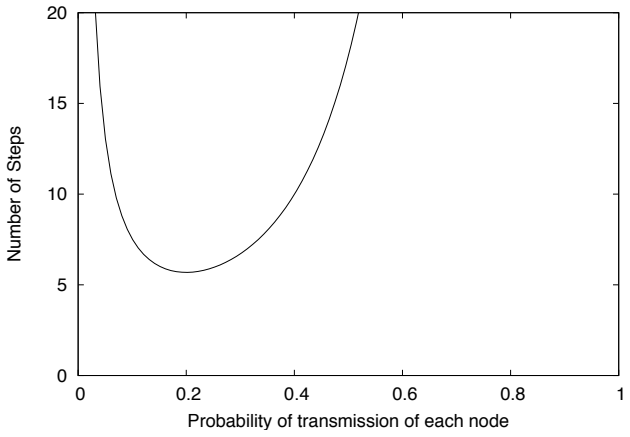

Figure 6.1: Number of required steps S, in order to the probability of transmission of each node p_t

represented in Figure 6.1 has a minimum around step 6, for $p_t = 0.2$. If we increase the total number of nodes, there will still be a minimum around value 6, but for a value of p_t much lower. Intuitively, when we increase the number of nodes, each node needs to reduce the number of transmission (i.e. reduce p_t), to avoid an excessive number of collisions and consequently, an increase in the total number of steps needed. Additionally, if we require a higher probability p_{sp}, of having a contribution from a correct node in the nonce set, the minimum of steps increases slightly (for example, for $p_{sp} = 0.99$, the minimum number of steps required increases to 9 steps).

This way, the total number of steps ($\mathcal{MS}$) the nonce generation with collision detection has to make, to achieve a nonce with the required strength against a guessing attack, for this particular example, is given by: $\mathcal{S} = c_i \cdot f + 6$. Note that this number os steps is probabilistic, and the number of steps really used may be either larger, or smaller.

Without collision detection In the algorithm without collision detection, the only source of randomness are the n bits given by the occupation, or idleness, of the communication medium. As before, we want to guarantee that the difficulty of guessing the node's contributions is at least as high as guessing the nonce. In this algorithm, we are using the hash SHA-1 to output the final nonce. Assuming SHA-1 to be perfect, the nonce will be constituted by 160 random bits. Consequently, if we wish to have the same level of difficulty in the contributions from the nodes, we need at least 160 steps (remember that one step can be seen as one random bit).

```
// Run at every node i
algorithm NSQ-NonceGeneration (n) is
    nonceSet ⟵ ∅
    collisions ⟵ f · c_i
    while (|nonceSet| < (n + collisions)) do
        if rand() < p_t then
            broadcast.send ([rand()]);
            nonce ⟵ nonce ∪ 1;
        else
            msg ⟵ broadcast.receive();
            if msg = null then
                nonce ⟵ nonce ∪ 0;
            else
                nonce ⟵ nonce ∪ 1;
    return hash(nonceSet);
```

Figure 6.2: NSQ-NonceGeneration algorithm.

Adding the number of steps in which malicious nodes are able to generate intentional collisions, we have a total number of steps given by: $\mathcal{S} = c_i \cdot f + 160$. Note that, while this number of steps is higher than the one presented for the previous algorithm, this number is fixed, and does not vary with the total number of network nodes.

6.4 The NSQ-NonceGeneration Algorithm

This section presents and describes in more detail the NSQ-Nonce Generation, which will later be used as a building block for our NSQ Protocol. Both nonce generation algorithms could be used as a building block, given that the same abstractions still apply, however, we choose the nonce generation without collision detection due to the fact that it possesses a limited number of steps, and does not require the existence of a collision detector.

The NSQ-NonceGeneration has the following interface: a node has to invoke NSQ-NonceGeneration(n) to obtain the correct nonce, taking as a parameter the number of correct bits (not influenced by byzantine nodes), the nonce should have (n). The algorithm for the NSQ-NonceGeneration is presented in Figure 6.2.

The nonce set (nonceSet) starts empty. In each step of communication, and while the desired number of contributions is less than n plus the maximum number of collisions malicious

nodes may generate $(f \cdot c_i)$, a node randomly decides whether to remain idle in that step, or to transmit a message. If the node decides to transmit (with a probability p_t), then it adds the value 1 to the nonceSet. Note that adding this value is independent of any action other nodes may perform (including byzantine nodes), since one or more transmissions will always result in all the network nodes adding a 1 to the nonceSet in that slot. If, on the other hand, the node chooses to remain idle, it listens the medium to verify the occurrence of a transmission (or collision), and adds 0 or 1, accordingly. Finally, the algorithm returns the 160-bit digest output by the application of the hash SHA-1 to the nonceSet.

6.5 Properties

6.5.1 Safety

Lemma 7: The nonce algorithm without collision detection outputs a correct nonce, with a probability arbitrarily close to 1.

Proof sketch: Since having contributions of n bits from correct nodes is enough to provide correctness to the output of the algorithm, and taking into account the maximum number of intentional collisions malicious nodes are able to generate (c_i), in a given number of steps $\mathcal{P}$, we have that, according to Equation 6.2, we only need to use a number of steps greater than $n + f \cdot c_i < \mathcal{P}$. Since $f \cdot c_i < \mathcal{P}$, $\mathcal{MS}'$ is finite and, therefore, it will always be possible to exceed it. □

6.5.2 Output Consistency

Lemma 8: At the end of the nonce algorithm without collision detection, every correct node is aware of the same nonce value.

Proof sketch: This property comes from the fact that in each step every node is aware of: i) the existence of a transmission (or collision); ii) the existence of radio silence. Every correct node is thus capable of following the protocol and outputs the same sequence of events and, therefore, the same final nonce. □

6.5.3 Termination

Lemma 9: Assuming the existence of a bound on the number of intentional collisions byzantine nodes may generate, the algorithm ends.

Proof sketch: Taking into consideration the fact that there is a maximum number of contributions from the network nodes, and that the algorithm progression does not depend on external conditions, the nonce generation algorithms return after a finite number of steps. □

Summary

In this chapter we further detailed the Nonce Generation algorithm, introducing two new algorithms, and discussing some relevant aspects. Furthermore, we detailed the implementation of the NSQ-NonceGeneration building block, providing proof of some relevant properties.

7 Non-Sybil Quorum Algorithm Construction

In this chapter, we propose an algorithm to create a quorum constituted by a maximum of q non-sybil identities (NSQ)[1], in a one-hop neighborhood of an ad hoc wireless network. We rely in the techniques and tests presented in previous chapters. At the end of the algorithm execution, every correct node possesses a quorum constituted by a majority of correct identities (e.g. identities from correct nodes). The final size of the quorum is one of the parameters of the algorithm. The existence of a correct quorum allows the assignment of critical network decisions to the quorum identities. This algorithm uses a combination of different resource tests, in order to achieve high efficiency and robustness. Furthermore, we show that none of the resource tests used in isolation are able to guarantee both the robustness and efficiency of the proposed combination.

7.1 Non-Sybil Quorum Construction

In previous chapters, we presented two distinct resource testing approaches to mitigate the sybil attack, namely, the radio and computational resource tests. Each of them has their own advantages and disadvantages. Radio Resource Tests (RRTs) were shown to be capable of eliminating all the sybil identities in a one-hop neighborhood with a probability arbitrarily close to 1. However, this technique requires a large number of tests, being therefore impractical for a network with a great number of nodes. Furthermore, when used to test the whole neighborhood, this method becomes an easy prey to a simple denial-of-Service attack, in which an attacker simply generates a large number of sybil identities and requires the network to make unnecessary tests. Computational Resource Tests (CRTs) were shown to be scalable and able to handle a large number of network nodes, at the cost of being unable to completely eliminate sybil identities.

[1] For simplicity, we will, from now on, designate by NSQ the quorum build at each node, and not the intersection of the several individual build quorums.

```
// executed at every node
algorithm NSQ (n, q) is
    C ⟵ ∅; //set of candidate identities to the quorum
    nonce ⟵ null; // string, nonce for the CRT
    // Phase one (Nonce generation)
    nonce ⟵ NSQ-NonceGeneration(n);
    // Phase two (Candidate Selection)
    C ⟵ NSQ-CRT (nonce,q);
    // Phase three (Quorum Validation)
    C ⟵ NSQ-RRT (C);
    return truncate_q(C);
```

Figure 7.1: Skeleton of the NSQ construction algorithm.

In light of these issues, we devised a protocol that allows us to take advantage of both methods and achieve a higher efficiency and robustness than any of the solutions applied individually would possess. We use this protocol to create a quorum of non-sybil identities, to which we may delegate the responsibility of critical network decisions.

The non-sybil quorum (NSQ) construction consists of three distinct phases: the nonce generation phase; candidate selection phase; and the quorum validation phase. The algorithm is depicted on Figure 7.1.

At the core of the algorithm, there is a candidate selection phase (phase 2). The purpose of this step is to find a suitable set of identities, of size σ (derived from the q input parameter to the NSQ algorithm) to form the non-sybil quorum. This set is obtained by using a computational resource test. As describe in Chapter 5, for each identity that a node wishes to propose, it has to solve a cryptographic puzzle that will consume some processing time. Taking into consideration the limited resources that each node possesses, this test limits the number of identities that each node is able to propose. To avoid the usage of previously calculated solutions, the puzzle has a parameter (nonce) which is only known at that time. This nonce is obtained by combining the input of n distinct identities (n is another input parameter of the NSQ algorithm, which was described in Chapter 6). As seen earlier, it is extremely difficult, using solely a CRT, to guarantee that there are no sybil identities in the candidate list. For this reason, the algorithm includes a third phase, that uses a Radio Resource Test (NSQ-RRT), with the objective of eliminating sybil identities from the candidate set, and prove to all remaining network elements (even though not being quorum candidates, they want to be aware of its existence) that the

identities of the quorum are non-sybil.

7.1.1 Nonce Generation

In our algorithm, the nonce generation phase has the sole purpose of providing a correct nonce for the second phase. We can use either one of the nonce generation algorithms presented in Chapter 6. If we choose the algorithm with collision detection, the parameter n given to the NSQ will be used as the number of distinct contributions from nodes that we require. If we choose the algorithm without collision detection, the parameter given to the NSQ should be the total number of random bits n we want the nonce to possess.

7.1.2 Candidate Selection

As stated earlier, the purpose of the candidate selection phase is to obtain a set of σ identities that are good candidates to form the quorum. In order to avoid byzantine nodes from proposing multiple identities for the candidate set, we use a CRT. More specifically, we use the Trusted Hashcash described in Chapter 5. The idea is to require that all nodes that wish to propose their identity for the candidate set, solve a cryptographic puzzle. This puzzle receives as parameters the correct nonce created in the previous phase, and the identity to be proposed.

This phase consists on receiving and validating the answers that are sent as soon as a node solves the puzzle, to try to be incorporated in the candidate set. It is assumed that the average time to solve the crypto-puzzle is orders of magnitude greater than the time necessary to disseminate the solutions. This way, when a byzantine node finishes the calculation of the puzzle for a second identity, all the correct nodes have already broadcasted their legitimate proposals, with a high (and configurable) probability. Note that a byzantine node that knows the nonce *a priori*, would be able to pre-compute as many answers as necessary, and propose an unlimited number of sybil identities. This motivates the existence of the first phase, the nonce generation algorithm.

Ideally, a perfect CRT would not allow a participant to solve more than one crypto-puzzle in time to be able to propose more than one identity and, thus, would not allow sybil identities in the candidate set. However, the nature of these tests makes the resolution time of the puzzle a random variable. Therefore, a non-zero probability exists, that a byzantine node is capable of

proposing more than one identity. However, and since the CRT provides us with a probabilistic limit on the number of sybil identities that the byzantine nodes may present $(s - f)$, we can obtain the number of required answers σ, in order to guarantee that at least $q - f$ correct identities are chosen to the candidate set, as was discussed in Chapter 5.

7.1.3 Quorum Validation

The quorum validation phase purges the remaining sybil identities in the set of identities resulting from the previous phases, by using a RRT. In this phase, the σ identities that advertised their valid crypto-puzzle answers will be tested against each other with the Optimized Simultaneous Sender Test (oSST). Note that, the use of a RRT to test the identities output by the CRT has two important advantages over testing the whole population: the number of executed tests decreases significantly, since for large values of N, $\sigma < N$; the attacker becomes unable to create a denial-of-service attack by proposing many sybil identities, since the CRT limits the total number of sybil identities in the output set.

There is however, a disadvantage in the use of the RRT: the output in all the correct nodes may not be the same. While in Chapter 4 we addressed the problem of consistency in detecting sybil identities, we only considered the case where the malicious nodes are consistent in defending their identities. A malicious node can, however, defend different identities at different times, or simply not defend them at all. This inconsistent behavior creates the possibility of some of the correct nodes eliminating different sybil identities (since the choice of the channels is random by nature), for example.

Note that, while different nodes may finish the NSQ protocol with different sets of identities, the only difference between each set output lies in the identities owned by malicious nodes. This is true since the correct nodes always behave consistently, and will never be excluded by other correct nodes. Moreover, and since sybil identities are detected with a configurable high probability, this means that every correct node will output all the identities of other correct nodes and a number between 0 and f of identities owned by malicious nodes.

Therefore, instead of trying to guarantee a consistent output in every node, and in order to have a correct quorum, we just have to guarantee that the intersection between the quorums of all correct nodes has the minimum of required correct identities.

7.2 Analysis

In order to assess the performance of the NSQ protocol, we will compare it with the use of each individual kind of resource test used in an entire population of nodes.

If we intend to use a RRT test to test a population of N nodes, as seen earlier, we need the number of messages given by Equation 4.4

From this, we have that the message complexity of applying a RRT to a whole population is in the order of $O(N^h)$ messages. Notice that this complexity applies to all the RRTs described in Chapter 4, and that the lowest possible complexity we are able to achieve is $O(N^2)$, when $h = 2$.

The complexity of CRTs has been discussed in Chapter 5. Both the Democracy and the Hashcash CRTs have a $O(N^2)$ complexity, while our Trusted Hashcash has a $O(N)$.

Finally, and regarding the message complexity of our protocol, we have: i) In the first phase, the nonce generation period, we have an upper-bound on the number of node contributions, given by variable n; ii) In the second phase, the number of CRT answers received is also bounded, but by variable σ. iii) Finally we have the radio resource test, which has a total number of sent messages of $\binom{\sigma}{h} \cdot \mathcal{R} \cdot h$.

From this, we have that the total number of messages transmitted by the protocol is $n + \sigma + \binom{\sigma}{h} \cdot \mathcal{R} \cdot h$. Since n is a fixed variable, we have a $O(\sigma^h)$ message complexity. We can easily see that the message complexity of the application of a resource test is much higher than in our protocol, since our message complexity does not depend directly on N, but depends on σ, instead. It is of particular interest to note that, an increase in the total number of nodes in the network, actually decreases the size of σ, since it also decreases the total number of identities the byzantine nodes are able to propose (s). However, σ also depends on the maximum number of byzantine nodes in the system f, increasing when we wish to tolerate a higher number of byzantine nodes.

7.3 Correcteness

We now provide a sketch of the proof for the two properties defined in Chapter 3, the "Sybil-Free Partial Consistent Quorum Set" and the "Probabilistic Termination".

7.3.1 Sybil-Free Partial Consistent Quorum Set

In order to prove the "Sybil-Free Partial Consistent Quorum Set" property, we first prove the following two intermediate Lemmas.

Lemma 10 (Sybil-Free): With a probability arbitrarily close to 1, there are no sybil identities in NSQ_i. More precisely, let: $U_i = \bigcup_{id \in NSQ_i} uses(id)$. We have that $|U_i| \geq |NSQ_i|$.

Proof sketch: This Lemma follows from Lemma 1. Since the last phase of the NSQ is the NSQ-RRT with the σ identities resulting from the CRT, having a sybil identity in NSQ_i would imply that the RRT did not eliminate that identity from the candidate set. However, according to Lemma 1, the output of the RRT, should not contain sybil identities, with the desired probability. □

Lemma 11 (Do no harm): Let Q_i be the output of the NSQ-RRT, for a given node n_i. If there is a correct identity A in Q_i, then, identity A is also present on all the outputs of the NSQ-RRT of every other correct node. More precisely, if $A \in Q_i$ for node n_i, then $A \in Q_i \ \forall_{n_i}$.

Proof sketch: This Lemma follows from Lemma 2, and the fact that the NSQ-CRT outputs the same candidate set $\mathcal{C}$ in every node. Assume the opposite, that there is a identity A, from a correct node that is present in the output Q_i of a correct node n_i, but not in the output Q_j of a correct node n_j. However, given that the input the NSQ-RRT in both nodes is the same ($\mathcal{C}$), and that the NSQ-RRT does not eliminate identities from correct nodes (Lemma 2), if $A \in Q_i$ then necessarily, $A \in Q_j$, which contradicts the assumption. □

With these two lemmas we are now able to prove the "Sybil-Free Partial Consistent Quorum Set" property of NSQ.

Theorem 1 (Sybil-Free Partial Consistent Quorum Set): There is a set of identities from correct nodes NSQ, common to every NSQ_i, such that $|NSQ| \geq q - f$, with a probability arbitrarily close to 1. Let $\Gamma()$ be defined as:

$$\Gamma(n_i) = \begin{cases} 1, \text{ if } n_i \text{ is a correct node} \\ 0, \text{ otherwise} \end{cases} .$$

Let $NSQ = NSQ_1 \cap NSQ_2 \cap ... \cap NSQ_{N-f}$ be the set of identities common to all NSQ_i of the correct nodes. Then,

$$\sum_{n_i}^{uses(NSQ)} \Gamma(n_i) \geq q - f.$$

Proof sketch:

This Theorem follows from Lemmas, 2, 5, 10 and 11. According to Lemma 5, the output set of the NSQ-CRT $\mathcal{C}$, contains at least $q - f$ correct identities. Furthermore, consider that: i) the set $\mathcal{C}$ is the same for every node; ii) the NSQ-RRT does not eliminate identities from correct nodes (Lemma 2), and all the correct identities are present in the output of every correct node (Lemma 11); iii) there are at most f identities from byzantine nodes in each correct node's NSQ_i output (Lemma 10). From this, and if we order lexicographically the output of the NSQ algorithm and truncate the first q identities, we are able to guarantee that: the remaining set is composed by, at most f identities from byzantine nodes and, at least by the first $q - f$ correct identities in the ordered set of NSQ_i. □

7.3.2 Probabilistic Termination

We can prove the "Probabilistic Termination" property, by leveraging on the previously proved lemmas, for each of the presented building blocks.

Lemma 12: Every node returns the quorum with a probability arbitrarily close to 1, given a finite number of steps.

Proof sketch: This Lemma follows from Lemma 3, Lemma 6 and Lemma 11. Since in each phase of the NSQ protocol there is a finite number of steps (Lemmas 3,6 and 11), the total number of steps to end the NSQ algorithm is the sum of all the steps of each phase, and is, therefore, finite. □

Summary

In this chapter we presented the Non-Sybil Quorum (NSQ) algorithm, describing succinctly the three building blocks that constitute it: NSQ-NonceGeneration, NSQ-CRT and the NSQ-

RRT. We discussed some of the properties these building blocks bring to the solution, finishing with proof of the problem statement introduced in Chapter 3

8 Conclusions and Future Work

In this thesis we focused on the development of security mechanisms to thwart the sybil attack in wireless ad hoc networks. The cornerstone of our work are resource tests, a promising technique that allows the mitigation of sybil identities, without requiring any pre-configuration of the nodes, being thus able of improving the scalability of the network. We presented two particular types of resource tests: Radio Resource Tests (RRTs) and Computational Resource Tests (CRTs).

Radio resource tests were analyzed both in isolation and when used to test an one-hop population. Furthermore, a comparison was made between them, using a framework that fully captures their characteristics. We also proposed two novel RRTs and an optimization to a RRT previously described in the literature, showing that each of them is best adapted to a different specific scenario, which has been described. Computational resource tests were also analyzed when used to test an one-hop population, and compared, using the same framework applied to the CRTs. We concluded that, while both techniques are viable mechanisms for detecting sybil identities, RRTs possess the capability of eliminating all the sybil identities, with a probability arbitrarily close to 1, and most CRTs do not. Conversely, CRTs are much more scalable when used to test a population with a large number of nodes then RRTs.

To be able to enhance the performance of our CRT optimization, we also proposed two new nonce generation algorithms, that allowed us to create a correct nonce (random and previously unknown) in a completely distributed fashion.

These resource tests and the nonce generation method, were used together in a new algorithm, the NSQ, to construct a quorum of non-sybil nodes in a one-hop wireless network. The algorithm is based on the use of resource tests not only to detect (and exclude) sybil identities, but also to speed the quorum construction. We concluded that our approach is efficient, and able to reduce the potential of denial-of-service attacks, still providing a full spectrum of desired probabilistic guarantees. This algorithm enables the buildup of a sybil free quorum, in a

scalable and robust way, by building upon the complementarity of the different algorithms that constitute it.

As future work, we would like to extend the Non-Sybil Quorum protocol to work in multi-hop networks. Moreover, we would also like to weaken our model assumptions, such as the existence of a collision detector, and the lack of network omissions. Finally, it would be an interesting the analysis of Radio Resource tests from a more theoretical point of view, and the proposal of a new type of *universe detector*, based on this kind of resource tests.

Bibliography

A. Juels, J. B. (1999). Proceedings of ndss '99 (networks and distributed security systems). pp. 151–165.

Ahn, L., M. Blum, N. Hopper, & J. Langford (2003). Captcha: Using hard ai problems for security. In *In Proceedings of Eurocrypt*, pp. 294–311. Springer-Verlag.

Akyildiz, I., W. Su, Y. Sankarasubramaniam, & E. Cayirci (2002). Wireless sensor networks: a survey. *Comput. Netw. 38*(4), 393–422.

Anthony, D. & A. John (1992). Data networks. In *Upper Saddle River*. Prentice âHall, Inc.

Aspnes, J., C. Jackson, & A. Krishnamurthy (2005). Exposing computationally-challenged Byzantine impostors. Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer Science.

Aura, T., P. Nikander, & J. Leiwo (2001). Dos-resistant authentication with client puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, London, UK, pp. 170–177. Springer-Verlag.

Back, A. (2002). Hashcash - a denial of service counter-measure. Technical report.

Bar-Yehuda, R., O. Goldreich, & A. Itai (1991). Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distrib. Comput. 5*(2), 67–71.

Bazzi, R. & G. Konjevod (2005). On the establishment of distinct identities in overlay networks. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, New York, NY, USA, pp. 312–320. ACM.

Becker, K. & U. Wille (1998). Communication complexity of group key distribution. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, New York, NY, USA, pp. 1–6. ACM.

Bellare, M., R. Canetti, & H. Krawczyk (1996). Keying hash functions for message authentication. pp. 1–15. Springer-Verlag.

bo Guo, Y. & J. feng Ma (2003). An efficient and robust conference key distribution protocol. *Computer Networks and Mobile Computing, International Conference on 0*, 338.

Boris, N. M. & B. Levine (2008). Quantifying resistance to the sybil attack. In *Proc. Financial Cryptography (FC)*.

Cheng, A. & E. Friedman (2005). Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, New York, NY, USA, pp. 128–132. ACM.

Cynthia Dwork, a. M. N. (1993). Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, London, UK, pp. 139–147. Springer-Verlag.

Dahill, B., K. Sanzgiri, B. Levine, E. Belding-Royer, & C. Shields (2002). A secure routing protocol for ad hoc networks. Technical report, Amherst, MA, USA.

Demirbas, M. & Y. Song (2006). An rssi-based scheme for sybil attack detection in wireless sensor networks. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, Washington, DC, USA, pp. 564–570. IEEE Computer Society.

Douceur, J. (2002). The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK, pp. 251–260. Springer-Verlag.

Eastlake, D. & P. Jones. Us secure hash algorithm 1 (sha1). http://www.ietf.org/rfc/rfc3174.txt?number=3174.

Ford, W. (1994). *Computer communications security: principles, standard protocols and techniques*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Fullmer, C. & J. Garcia-Luna-Aceves (1995). Fama-pj: A channel access protocol for wireless lans. In *Proc. ACM Mobile Computing and Networking '95*, pp. 76–85. ACM.

Gupta, P. & P. Kumar (2000). The capacity of wireless networks. *Information Theory, IEEE Transactions on 46*(2), 388–404.

Haas, Z. & b. Liang (1999). Ad hoc mobility management with uniform quorum systems. *Networking, IEEE/ACM Transactions on 7*(2), 228–240.

Hu, Y., D. Johnson, & A. Perrig (2002). Sead: secure efficient distance vector routing for mobile wireless ad hoc networks. In *Mobile Computing Systems and Applications, 2002.*

Proceedings Fourth IEEE Workshop on, pp. 3–13.

Hu, Y., A. Perrig, & D. Johnson (2002). Ariadne: a secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, New York, NY, USA, pp. 12–23. ACM Press.

Hubaux, J., J. Boudec, S. Giordano, M. Hamdi, L. Blazevic, L. Buttyan, & M. Vojnovic (2000). Towards mobile ad-hoc wans: terminodes. *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE 3*, 1052–1059 vol.3.

Hubaux, J.-P., L. Buttyán, & S. Capkun (2001). The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, New York, NY, USA, pp. 146–155. ACM.

Ibe, O. (2005). *Fundamentals of applied probability and random processes*. Academic Press.

Ilyas, M. & R. Dorf (Eds.) (2003). *The handbook of ad hoc wireless networks*. Boca Raton, FL, USA: CRC Press, Inc.

Jiang, J.-R., Y.-C. Tseng, C.-S. Hsu, & T.-H. Lai (2005). Quorum-based asynchronous power-saving protocols for ieee 802.11 ad hoc networks. *Mobile Networks and Applications 10*(1), 169–181.

Karlof, C. & D. Wagner (2003). Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks 1*(2-3), 293 – 315. Sensor Network Protocols and Applications.

Khattab, S., D. Mosse, & R. Melhem (2008). Jamming mitigation in multi-radio wireless networks: reactive or proactive? In *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, New York, NY, USA, pp. 1–10. ACM.

Kim, Y., A. Perrig, & G. Tsudik (2000). Simple and fault-tolerant key agreement for dynamic collaborative groups. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, New York, NY, USA, pp. 235–244. ACM.

Koo, C.-Y., V. Bhandari, J. Katz, & N. Vaidya (2006). Reliable broadcast in radio networks: the bounded collision case. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, New York, NY, USA, pp. 258–264. ACM.

Krishnamurthy, P. M. S. (2004). *Ad Hoc Networks Technologies and Protocols*. Springer.

Lamport, L., R. Shostak, & M. Pease (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems 4*, 382–401.

Lau, F., S. Rubin, M. Smith, & L. Trajkovic (2000). Distributed denial of service attacks. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, Volume 3, pp. 2275–2280 vol.3.

Levine, B., C. Shields, & N. Margolin (2006). A Survey of Solutions to the Sybil Attack. Tech report 2006-052, University of Massachusetts Amherst.

Li, J., C. Blake, D. De Couto, H. Lee, & R. Morris (2001). Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, Rome, Italy, pp. 61–69.

Luo, J., J. pierre Hubaux, & P. Eugster (2003). Pan: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In *In Proc. of MobiHoc*, pp. 1–12.

Malkhi, D. & M. Reiter (1998). Byzantine quorum systems. *Distributed Computing 11*, 569–578.

Maniatis, P., D. Rosenthal, M. Roussopoulos, M. Baker, T. Giuli, & Y. Muliadi (2003). Preserving peer replicas by rate-limited sampled voting. *SIGOPS Oper. Syst. Rev. 37*(5), 44–59.

Maniatis, P., M. Roussopoulos, T. Giuli, D. Rosenthal, & M. Baker (2005). The lockss peer-to-peer digital preservation system. *ACM Trans. Comput. Syst. 23*(1), 2–50.

Martucci, L., M. Kohlweiss, C. Andersson, & A. Panchenko (2008). Self-certified sybil-free pseudonyms. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, New York, NY, USA, pp. 154–159. ACM.

Merkle, R. (1978). Secure communications over insecure channels. *Commun. ACM 21*(4), 294–299.

Mishra, A. (2008). *Security and Quality of Service in Ad Hoc Wireless Networks*. Cambridge University Press.

Mónica, D., J. Leitão, L. Rodrigues, & C. Ribeiro (2009a). Contrucão observável de um sistema de quorum não sybil na vizinhanca rádio de uma rede ad-hoc sem fios. In *Actas do primeiro Simpósio de Informática*, Lisboa, Portugal.

Mónica, D., J. Leitão, L. Rodrigues, & C. Ribeiro (2009b). 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 (WRAITS)*, Estoril, Portugal, pp. F21–F26.

Nadkarni, K. & A. Mishra (2004). A novel intrusion detection approach for wireless ad hoc networks. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE 2*, 831–836 Vol.2.

Newsome, J., E. Shi, D. Song, & A. Perrig (2004). The sybil attack in sensor networks: analysis & defenses. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pp. 259–268.

Noubir, G. (2004). On connectivity in ad hoc network under jamming using directional antennas and mobility. In *In International Conference on Wired /Wireless Internet Communications, Lecture Notes in Computer Science*, pp. 186–200. Springer-Verlag.

Papadimitratos, P. & Z. Haas (2002). Secure routing for mobile ad hoc networks. In *in SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002*, pp. 27–31.

Perrig, A. Efficient collaborative key management protocols for secure autonomous group communication. In *International Workshop on Cryptographic Techniques and E-Commerce CrypTEC '99*, pp. 192–202.

Piro, C., C. Shields, & B. Levine (2006). Detecting the sybil attack in mobile ad hoc networks. *Securecomm and Workshops, 2006*, 1–11.

Pointcheval, D. & J. Stern (2000). Security arguments for digital signatures and blind signatures. *Journal of Cryptology 13*, 361–396.

Rafaeli, S. & D. Hutchison (2003). A survey of key management for secure group communication. *ACM Comput. Surv. 35*(3), 309–329.

Rivest, R., A. Shamir, & D. Wagner (1996). Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684.

Stajano, F. & R. Anderson (2002). The resurrecting duckling: security issues for ubiquitous computing. *Computer 35*(4), 22–26.

Steiner, M., G. Tsudik, & M. Waidner (1996). Diffie-hellman key distribution extended to group communication. In *CCS '96: Proceedings of the 3rd ACM conference on Computer*

and communications security, New York, NY, USA, pp. 31–37. ACM.

Veríssimo, P., N. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, & I. Welch (2006). Intrusion-tolerant middleware: the road to automatic security. *Security Privacy, IEEE 4*(4), 54–62.

Veríssimo, P. & L. Rodrigues (2001). *Distributed Systems for System Architects*. Norwell, MA, USA: Kluwer Academic Publishers.

Vesa, K. (2000). Security in ad hoc networks, in proceedings of the helsinki university of technology, seminars on network security, helsinki, finland,.

Wang, X. & M. Reiter (2003). Defending against denial-of-service attacks with puzzle auctions. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, pp. 78. IEEE Computer Society.

Xiao, B., B. Yo, & C. Gao (2006). Detection and localization of sybil nodes in vanets. In *DIWANS '06: Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, New York, NY, USA, pp. 1–8. ACM.

Xu, T. & J. Wu (2007). Quorum based ip address autoconfiguration in mobile ad hoc networks. *Distributed Computing Systems Workshops, International Conference on 0*, 1.

Yu, H., M. Kaminsky, P. Gibbons, & A. Flaxman (2008). Sybilguard: Defending against sybil attacks via social networks. *Networking, IEEE/ACM Transactions on 16*(3), 576–589.

Zhang, Y. & W. Lee (2000). Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, New York, NY, USA, pp. 275–283. ACM.

Zhou, L. & Z. Haas (1999). Securing ad hoc networks. *Network, IEEE 13*(6), 24–30.