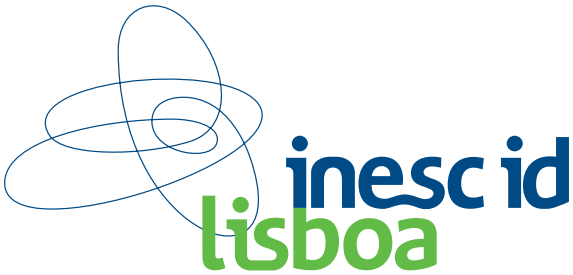

Technical Report RT/55/2009

Observable Non-Sybil Quorums Construction in One-Hop Wireless Ad Hoc Networks

Diogo Mónica
INESC-ID/IST
diogo.monica@gsd.inesc-id.pt

João Leitão
INESC-ID/IST
jleitao@gsd.inesc-id.pt

Luis Rodrigues
INESC-ID/IST
ler@ist.utl.pt

Carlos Ribeiro
INESC-ID/IST
carlos.ribeiro@ist.utl.pt

Dec 2009

Abstract

The Sybil Attack is a serious threat to the secure and dependable operation of wireless ad hoc networks. This paper proposes an algorithm to provide each correct node in an one-hop wireless network with a quorum of non-Sybil identities from the neighbourhood. The quorums provided to different correct nodes may differ, but their intersection is composed by a majority of correct identities, with an arbitrarily close to 1 probability. Therefore, the quorums may be used for different purposes, such as voting. The algorithm is based on the combination of different resource tests, to efficiently detect (and exclude) Sybil identities.

Keywords: Distributed Protocols; Intrusion Detection and Tolerance, Survivability; Mobile, Wireless, or Ad-Hoc Systems; Networking and Networked Systems; Security

This work was partially supported by FCT under grants PTDC/EIA/65588/2006 (Privato) and PTDC/EIA/71752/2006 (Redico).

Portions of this technical report have been published in The 40^{th} Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010).

Observable Non-Sybil Quorums Construction in One-Hop Wireless Ad Hoc Networks *

Diogo Mónica João Leitão Luís Rodrigues Carlos Ribeiro
INESC-ID/IST
{diogo.monica, jleitao}@gsd.inesc-id.pt {ler, carlos.ribeiro}@ist.utl.pt

Abstract

The Sybil Attack is a serious threat to the secure and dependable operation of wireless ad hoc networks. This paper proposes an algorithm to provide each correct node in an one-hop wireless network with a quorum of non-Sybil identities from the neighbourhood. The quorums provided to different correct nodes may differ, but their intersection is composed by a majority of correct identities, with an arbitrarily close to 1 probability. Therefore, the quorums may be used for different purposes, such as voting. The algorithm is based on the combination of different resource tests, to efficiently detect (and exclude) Sybil identities.

1 Introduction

Quorums may be used as a mechanism to increase the dependability of critical services in a distributed system. Typically, one has a set of replicated servers that provide a service to a group of clients; to ensure the correct operation of the replicated service, any operation submitted by a client must be executed on a quorum of distinct replicas [9]. Unfortunately, such a quorum cannot be enforced if a malicious node may assume simultaneously multiple identities in the system, and impersonate several replicas. Such an attack, known as the Sybil attack [3], can easily disrupt the correct operation of the system.

Wireless and ad hoc networks are naturally prone to Sybil attacks. A malicious node that can simultaneously assume several identities can easily disrupt the operation of several distributed protocols, such as storage, routing, data aggregation, voting, intrusion detection, and resource sharing [12]. The nature of the wireless communication, and the ad hoc deployment of nodes, make it easier for a malicious node to launch such an attack. The Sybil attack is, therefore, a serious threat to the secure and dependable operation of wireless ad hoc networks.

Although several solutions to tackle the Sybil attack have been previously proposed [14, 6, 12], most of them require a central trusted authority or some pre-shared secret, which makes them unsuitable for ad hoc networks. A promising approach, with the potential to thwart the Sybil attack without the aforementioned unpractical requirements, consists in using resource tests. This approach assumes that it is possible to establish bounds on the amount of resources available to a single node. By testing the bounds on the resources owned by a set of identities, one can detect when such a set participating in the system does not own the aggregate amount of resources expected if all of the participating identities belonged to distinct participants, thus allowing the detection and exclusion of Sybil identities.

One such family of resource tests are the Radio Resource Tests (RRT), where the nodes are tested for the ability to receive or transmit messages simultaneously in different radio channels. These tests assume that each node may

*This work was partially supported by FCT under grants PTDC/EIA/65588/2006 (Privato) and PTDC/EIA/71752/2006 (Redico).

only send or receive a message simultaneously. However, in previous work [10], we have shown that even the most efficient RRT requires a prohibitively high number of messages, even when the number of identities to test is small. Since a malicious node may propose an unbounded number of identities, the test is not viable even for a network with a small number of participants.

In this paper we propose a novel algorithm, based on two distinct types of resource tests, that provides each participant in a one-hop wireless ad hoc network, with a probabilistic Sybil-free quorum-system [9]. This algorithm is called the Non-Sybil Quorum (NSQ). The quorums provided to different nodes may differ, but our algorithm ensures that their intersection has a majority of identities owned by non-malicious (*i.e.* non-Byzantine) nodes, with an arbitrarily high probability. Thanks to the combination of more than one resource test, our algorithm is both efficient and robust. The resulting quorums offer the possibility for delegation of critical network operations to a trusted group of identities, without exposing the system to the Sybil attack. For example, if the network nodes wish to vote on the outcome of a certain operation (e.g. ignoring a misbehaving node), each node can trust the votes from the identities present in the quorum output from our algorithm (since this quorum is guaranteed to be Sybil-free, and to possess a configurable majority of correct nodes).

The size q of the quorums is a parameter of the algorithm, which allows its output to be used in a large number of scenarios. For instance, if f Byzantine nodes are assumed to exist in a one-hop neighbourhood, one may configure the algorithm to return quorums of size $q \geq 3f + 1$, in order to generate Byzantine fault-tolerant quorums.

The rest of this paper is organised as follows. Section 2 introduces the system model and the problem statement. In Section 3 we provide a detailed description of our algorithm. Section 4 proves the correctness of the presented solution, followed by Section 5, in which we show a performance evaluation of our algorithm. Finally, Section 6 concludes the paper, establishing directions for future work.

2 Model and Problem Statement

In this section we first enumerate the set of assumptions that model the ad hoc networks for which our algorithm is designed. Then we provide a precise description of the problem addressed in this paper.

2.1 System Model

We assume a system composed of N nodes that communicate through a shared wireless medium, constituting a one-hop neighbourhood (i.e. all nodes can communicate with each other). All nodes know an estimate of N. Note that this estimate is very easy to obtain in a one-hop wireless network, since there is a bound on the number of nodes that are able to communicate successfully in the same vicinity. Network nodes can be either malicious (Byzantine) or correct (we will use the terms malicious and Byzantine interchangeably). Byzantine nodes may exhibit arbitrary behaviour, such as sending arbitrary messages or failing communication steps. We consider that there exist at most f Byzantine nodes, which may collude between themselves.

We assume that the system is synchronous and 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. All nodes are assumed to have a single radio device (including the Byzantine). Radio devices can operate in K different channels, being, however, incapable of simultaneous operation in more than one channel. Furthermore, we assume that $f < K$, which is a requirement for the use of radio resource tests, as will be described further ahead in the text[1].

We assume that all nodes have limited resources, namely bounded communication and computational power, and that the resources are similar for every node in the system. Consistently with this assumption, we limit the number of transmissions c_i that all nodes (including Byzantine nodes) are able to perform, in a given number of consecutive steps P; 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,

[1]Note that K is limited by the communication technology used, e.g., for 802.11, $K = 13$.

in that period, to $c_i \cdot f < P$. Note that this assumption does not prevent correct nodes from generating (a limited number of) non-intentional collisions as a result of the algorithm operation. This model includes, but it is not limited to, nodes with battery constraints.

Also, we assume that whenever a (forced or involuntary) collision occurs, it can be detected by correct nodes, including the transmitters. This can be achieved by using the following simple technique [5]: upon detecting a collision, correct participants transmit a dedicated message in the same radio channel where the collision happened, for a period long enough to ensure that all nodes that transmitted the colliding messages are able to detect the collision. Finally, we assume that, when no collisions occur, all correct nodes successfully receive the transmitted messages (techniques to mask omissions are described in [7]).

2.2 Identities

All messages exchanged in the system are cryptographically associated with a given *identity*, assigned by the sending node. Correct nodes are assumed to use a single identity, that is never used by any other node, whenever they send a message (we dub such an identity as a *correct identity*). 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 to sign outgoing messages [11]. 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 the remaining nodes. The public key can then be used as an identity, and the message signed with the corresponding private key.

Malicious nodes may launch a Sybil attack by using more than one identity for communication, in an attempt to be perceived by correct nodes as more nodes than they actually are. These 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 associated to 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 $1 \leq |uses(id_b)| \leq f$.

2.3 Problem Statement

The objective of our algorithm is to deliver to each and every correct node n_i, in a one-hop radio neighbourhood, a quorum of identities NSQ_i of size q. The identities in each NSQ_i can belong to either correct or malicious nodes. However, if f malicious nodes exist in the system, they cannot impose more than f identities in any given NSQ_i.

The quorums provided to different correct nodes may differ. For instance, let i and j be two correct nodes: malicious nodes may impose identity id_a to NSQ_i and id_b to NSQ_j. Also, malicious nodes may stop operating at any time. However, the intersection of all quorums returned to each correct node contains at least $q - f$ identities belonging to correct nodes.

More formally, the problem addressed in this paper is to design a algorithm that provides each correct node i with a quorum NSQ_i with the following properties:

P1: Q-Size. Any quorum of a correct node NSQ_i has exactly q identities. More precisely $|NSQ_i| = q \; \forall_i \; s.t. \; n_i$ is correct.

P2: Probabilistic Sybil-free. With a probability arbitrarily close to 1, in any quorum NSQ_i delivered to a correct node n_i, the number of identities that have been proposed by malicious nodes is no larger than the number of malicious nodes in the system, f. More precisely, $|NSQ_i| \leq |\; uses(NSQ_i)|$, where $uses(NSQ_i) = \bigcup_{id_j \in NSQ_i} uses(id_j)$.

```
// Executed at every node i
algorithm NSQ-algorithm is
    C_i ⟵ ∅; //set of candidate identities to the quorum
    nonce ⟵ null; // string, nonce for the CRT

    nonce ⟵ nonceGeneration(); // Phase one
    C_i ⟵ candidateSelection (nonce); // Phase two
    NSQ_i ⟵ quorumValidation (C_i); // Phase three
    return NSQ_i;
```

Figure 1. Skeleton of the NSQ algorithm.

P3: Probabilistic Partial Consistency. The intersection of the quorums delivered to all correct nodes has, at least, $q - f$ identities used by correct nodes, with a probability arbitrarily close to 1.

P4: Termination. Every correct node's execution of NSQ returns a quorum set in a finite number of steps.

No protocol can prevent Byzantine nodes from joining the quorum, given that malicious nodes may respect the protocol and use a single identity. In fact, if a Byzantine node behaves correctly, it is indistinguishable from a correct node. Consequently, the size of q must be selected according to the type of application that uses the quorum returned by the NSQ algorithm. Typically, q must be set high enough to ensure that a majority of correct nodes exists in the quorums intersection, for instance, by setting $q = 3f + 1$ as in [8].

3 Non-Sybil Quorum Construction

We first provide an overview of the NSQ algorithm. The algorithm has 3 phases, as depicted in Figure 1, namely: a nonce generation phase, a candidate selection phase, and a quorum validation phase.

The rationale for the 3 phases is easier to explain if we start by describing *Phase 2*, which is at the core of the algorithm. The purpose of this phase is to find a set of candidate identities to form the non-Sybil quorums. To prevent malicious nodes from proposing an unbounded number of Sybil identities, a computational resource test (CRT) is used. More precisely, before proposing a candidate identity, a node is required to solve a cryptographic-puzzle that will consume processing time. Given that nodes have limited resources, the number of crypto-puzzles that a malicious node is able to solve in a given time is limited. We will describe the crypto-puzzle in detail in Section 3.2. As will be seen, there are two characteristics of the computational resource test that justify the other two phases of the algorithm, namely:

i) The crypto-puzzle requires, as input, a random nonce that cannot be pre-computed or guessed by malicious nodes. The online computation of such a nonce is the purpose of *Phase 1* of the algorithm.

ii) The nature of the crypto-puzzle prevents malicious nodes from proposing an unbounded number of Sybil candidate identities but cannot completely exclude the possibility of a malicious node proposing multiple Sybil identities (and this may happen with a non-negligible probability). The purpose of *Phase 3* of the algorithm is to filter any remaining Sybil identity that is not excluded by the CRT. *Phase 3* is based on a radio resource test (RRT).

It is worth noting that, although the RRT used in *Phase 3* is more effective in detecting (and excluding) Sybil identities, it is not as efficient as the CRT used in *Phase 2*. Therefore, our algorithm uses the most efficient method to exclude the vast majority of Sybil identities that could, otherwise, be proposed by malicious nodes, and the more expensive method to eliminate the remaining Sybil identities. We now describe each phase of the NSQ algorithm in detail.

```
// Executed at every node i
algorithm nonceGeneration () is
  contrib_i ⟵ rand();
  nonce ⟵ ∅;
  currentStep ⟵ 0;
  done ⟵ 0;
  while (currentStep < TS) do
    if done = 0 and rand() < p_t then
      if bcast.send([contrib_i]) ≠ collision then
        nonce ⟵ nonce ∪ [contrib_i];
        done ⟵ 1;
    else
      msg ⟵ bcast.receive();
      if msg ≠ null and msg ≠collision then
        nonce ⟵ nonce ∪ msg;
    currentStep ⟵ currentStep + 1;
  return hash(nonce);
```

Figure 2. Nonce generation.

3.1 Nonce Generation

The goal of the nonce generation phase is to create, online and in a distributed manner, a random string of bits that depends on the input of, at least, one correct node. By ensuring that at least one correct node contributes to the string, and that the contribution of correct nodes cannot be predicted in advance, we create a nonce that can safely be used as input to the crypto-puzzle that implements the computational resource test of *Phase 2*.

The nonce generation phase consist of a sequence of communication steps. In each step, at most one node provides a contribution to the nonce but, as will become clear, not every step contributes to the nonce. The number of steps is fixed *a priori* and is a function of the system's size, and the number and communication power of malicious nodes that need to be tolerated. Intuitively, the idea is to execute enough communication steps such that, with limited resources, malicious nodes cannot systematically prevent correct nodes from transmitting. Therefore, when all steps have been executed, the nonce includes the contribution of at least one correct node, with a configurable high probability.

The nonce generation algorithm is depicted in Figure 2. Initially, the nonce has no contributions. In each step, a correct node that has not yet contributed to the nonce tries to transmit. The contribution of a correct node to the nonce is a string of random bits. To avoid unnecessary collisions, nodes only transmit with a given probability $p_t > 0$. If a node transmits and no collisions occurs, the random value is added to the nonce. If a collision occurs, or no node opts to transmit it's contribution, no value is added to the nonce. Note that the properties of the communication channel, listed in Section 2, provide, by construction, the necessary guaranties to have an implicit agreement among correct nodes on when to add or not a value to the nonce. In fact, since collisions can be detected by all correct nodes and, when no collisions occur, omissions are masked, either all correct nodes accept or discard a contribution to the nonce in each step.

Correct nodes never contribute with more than one value to the nonce. I.e., when a correct node decides to participate in one step and a collision is not detected, it adds its own value to its copy of the nonce and never participates actively again in the subsequent steps of the nonce generation. Conversely, malicious nodes may propose multiple values, or generate collisions to prevent correct nodes from proposing theirs. However, since the resources of all nodes are limited, they cannot do this at all steps of the nonce generation.

When all steps have been executed, every node returns the same nonce value, by applying a deterministic function over the resulting nonce set, for instance, by using a one-way hash function such as SHA-1.

The total number of communication steps of the nonce generation phase (TS) must be set to ensure that,

with a probability arbitrarily close to 1, at least one correct node transmits in a collision-free step. Successful transmissions may not occur in all communication steps due to:

i) Collisions intentionally caused by malicious nodes. The total number of collisions that can occur due to this effect is a function of the number of malicious nodes f and of the power of each node (i.e., the number of transmissions they are able to perform). We denote the total number of steps with collisions caused by Byzantine nodes BS. The remaining steps in TS, i.e., the steps in which there will be no intentional collisions caused by Byzantine nodes, will be denoted by correct steps (CS).

ii) Involuntary collisions caused by two correct nodes attempting to transmit on the same step. The total number of collisions that can occur due to this effect is a function of the total number of nodes in the system, the transmission probability p_t, and the number of transmissions they are able to perform. We denote the total number of steps with collisions caused by correct nodes CoS.

iii) Silent steps, in which no node chooses to transmit. The total number of silent steps is also a function of the number of nodes in the system and the transmission probability p_t. We denote the total number of silent steps as SS.

We also denote the number of steps in which there is a valid contribution to the nonce as VS. Hence, the number of correct steps is given by:

$$CS = CoS + SS + VS \tag{1}$$

and the number of total steps is given by:

$$TS = BS + CS \tag{2}$$

Recall from Section 2 that each Byzantine node is able to send at most c_i messages in P steps. Therefore, BS satisfies the following inequality:

$$BS \leq \lceil TS \cdot \frac{c_i}{P} \cdot f \rceil \tag{3}$$

On the other hand, the number of valid contributions VS is a random variable that depends on the number of correct steps CS. The distribution probability of having at least one valid transmission from one of the $N - f$ correct nodes is:

$$P(VS \geq 1) = 1 - (1 - p_s)^{CS}, \tag{4}$$

where p_s is the probability that only one of the $N - f$ correct nodes tries to transmit on a given step:

$$p_s = p_{et} \cdot (1 - p_{et})^{N-f-1} \cdot (N - f),$$

and $p_{et} = p_t \cdot (1 - p_{ex})$ is the effective probability of transmission in each step of each correct node, which is given by the probability of transmission p_t times the probability of the node not having exhausted its transmission resources ($1 - p_{ex}$). By choosing $p_t \ll \frac{c_i}{P}$, the probability that a correct node exhausts its transmission capabilities in the TS steps can be brought arbitrarily close to zero for any reasonable sized TS, therefore $p_{et} \simeq p_t$.[2]

Lemma 1. *There is a probability $p_t < \frac{c_i}{P}$, such that the probability of resource exhaustion $p_{ex} = f(p_t)$ is arbitrarily close to 0, for every large number of steps S. More precisely, for a given probability δ there is a $p_t < \frac{c_i}{P} : f(p_t) < \delta$.*

[2]For a probability of transmission as high as $p_t = \frac{c_i}{4P}$ and a capacity of transmission of each correct node $ct = TS \cdot \frac{c_i}{P}$ as low as 8 messages, the probability p_{ex} presents values below 10^{-5}. For lower values of p_t and higher capacities of transmissions (ct), p_{ex} values becomes even lower.

Proof. Given that $1 - p_{ex} = p(X < TS \cdot \frac{c_i}{P})$, where X is a binomial distribution variable $B(S, p_t)$, and that the binomial distribution may be approximated to the Normal distribution $X \sim N(\mu, \sigma^2)$ for large S, then $p(X < TS \cdot \frac{c_i}{P})$ is given by the Normal cumulative distribution function

$$\Phi(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right],$$

which for the specific values of $\mu = S \cdot p_t$, $\sigma^2 = S \cdot p_t(1-p_t))$ and $x = TS \cdot \frac{c_i}{P}$, gives

$$p_{ex} = f(p_t) = \frac{1}{2}\left[1 - \text{erf}\left(\frac{TS \cdot \frac{c_i}{P} - S \cdot p_t}{\sqrt{2S \cdot p_t(1-p_t)}}\right)\right].$$

By choosing $\delta = f(1.1p_t)$ it is trivial to verify that $f(p_t) < \delta$ □

According to (3), the amount of Byzantine steps BS can be upper bounded by:

$$BS < TS \cdot \frac{c_i}{P} \cdot f + 1,$$

and, using (2), we get a lower bound on CS

$$CS > TS \cdot (1 - \frac{c_i}{P} \cdot f) - 1. \tag{5}$$

Replacing CS in (4) we get the lower bound on the probability of success $P(VS \geq 1)$ as a function of TS

$$P(VS \geq 1) > 1 - (1 - p_s)^{TS \cdot (1 - \frac{c_i}{P} \cdot f) - 1.} \tag{6}$$

Lemma 2. *There is a number of steps TS, such that we have a probability $P(VS \geq 1)$, arbitrarily close to 1, of a correct nonce output.*

Proof. Given that by definition $c_i \cdot f < P \Leftrightarrow (1 - \frac{c_i}{P} \cdot f) > 0$ and $(1 - p_s) < 1$, then, when $TS \to \infty$, $(1 - p_s)^{TS \cdot (1 - \frac{c_i}{P} \cdot f) - 1.}$ will tend to 0. This then implies that $P(VS \geq 1) \to 1$ (6). □

3.2 Candidate Selection

The goal of the candidate selection phase is to obtain a set $\mathcal{C}$ of identities that are good candidates to participate in the non-Sybil quorums. $\mathcal{C}$ must be such that: *i)* at least $q - f$ identities proposed by correct nodes are included in $\mathcal{C}$; *ii)* $\mathcal{C}$ includes as few Sybil identities as possible.

To limit the number of Sybil identities proposed by malicious nodes, this phase resorts to a computational resource test. The fundamental idea is that nodes, before being able to propose an identity, are required to solve a cryptographic puzzle, in a given period of time (T). This puzzle takes the previously generated nonce and the identity to be proposed as input parameters. Therefore, in the candidate selection phase each correct node performs the following tasks:

i) It attempts to solve the crypto-puzzle once, to propose a single identity to the candidate set. A correct node sends its proposal as soon as it solves the crypto-puzzle.

ii) It listens to proposals sent by other nodes until the end of period T.

Malicious nodes may attempt to solve the puzzle for multiple identities, in order to propose Sybil identities to the candidate set. However, in order for a malicious node to propose s Sybil identities, it has to solve s distinct crypto-puzzles. Note that, if a malicious node has knowledge about the nonce *a priori*, it may pre-compute the

```
// Executed at every node i
algorithm candidateSelection (nonce, q) is
  C ⟵ ∅;
  done ⟵ 0;
  t ⟵ 0;
  start ⟵ time();
  Task 1: // Crypto-puzzle resolution
    a_i ⟵ crt.solve(nonce, id_i);
    t ⟵ 1;

  Task 2: // Answer verification
    while time() < start + T do
      if t = 1 and rand() < p_t and done = 0 do
        if bcast.send(id_i,a_i) ≠ collision do
          C ⟵ C ∪ {id_i};
          done ⟵ 1;
        else
          m ⟵ bcast.receive();
          if m = (PROPOSAL) do
            if crt.verify(nonce,m_id,m_a) do
              C ⟵ C ∪ {id};
    return C;
```

Figure 3. Candidate Selection Phase.

solution to the crypto-puzzle for an arbitrary number of Sybil identities, being thus able to propose an unlimited number of identities. This motivates the existence of *Phase 1*.

The operation of the candidate selection phase is captured in Figure 3. The algorithm uses two tasks. One task to solve the crypto-puzzle and put the response to the puzzle in a variable t; and a second task that, in each communication step, either attempts to collect and validate proposals from other participants, or, if the solution to the puzzle is already available, to propose the node's identity. A PROPOSAL message includes the identity being proposed and the solution of the crypto-puzzle for that identity. When nodes receive a proposal, they must confirm that the solution to the crypto-puzzle is valid. In the affirmative case, nodes add the corresponding identity to the candidate set. If the solution is not valid, the received proposal is ignored. The algorithm terminates after the pre-defined period T. Proposals are transmitted using the medium access protocol used for the nonce generation. Note that T is several orders of magnitude higher than TS, the time it takes to send at least one PROPOSAL message from a correct node. Therefore, the Byzantine nodes will be unable to hinder the proposal of identities, by resorting to denial-of-service attacks.

We assume that the computational resource test has the following interface: A node invokes crt.solve(nonce,id) to solve the crypto-puzzle (taking the nonce and the node identity as input parameters); the invocation returns the response to the crypto-puzzle a_{id}. Also, the method crt.verify(nonce,id, a_{id}) allows any node to validate a response to the crypto-puzzle, given a nonce, an identity id and the response itself (a_{id}). The challenges in selecting and configuring the computational resource test are:

i) Setting the right value for the period T, such that, with high probability, at least $q - f$ correct nodes are able to propose their correct (non-Sybil) identity.

ii) Selecting a suitable cryptographic puzzle, such that it is unlikely that malicious nodes succeed in proposing many Sybil identities in the time it takes the $N - f$ correct nodes to propose their correct (non-Sybil) identities.

The crypto-puzzle should ensure that there is a high probability that at least $q - f$ nodes have been able to disseminate their (legitimate) proposals, before a Byzantine node is able to finish the crypto-puzzle for a second (Sybil) identity. Although our algorithm is not tied to a specific crypto-puzzle, for self containment we now discuss

an example of a crypto-puzzle that meets our requirements.

3.2.1 Example Crypto-Puzzle

Several kinds of computational resource tests have been proposed in the literature. The use of the technique was firstly proposed in [4], as a method for mitigating junk email ("spam"), and later used as a defence against denial-of-service attacks [2]. In [1], the authors introduced a computational resource test (CRT) to verify if participants own an expected amount of computational power. The test, simply entitled crypto-puzzle, consists of having nodes solve a cryptographic problem, which takes a nonce as an input and is only solvable by brute force calculation, in a limited amount of time. This way, a node with constrained computationally power has a limit to the number of crypto-puzzles it can solve in that time period, which sets an upper bound (eventually probabilistic) to the number of Sybil identities it can present to the network.

Given a one-way hash function $\mathcal{H}$, and an operator $\mathbf{left}_b(str)$ that returns a sub string composed by the b leftmost bits of a bit string str, a 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 a static configuration parameter, that controls the puzzle *hardness*.

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 a is a bit string where the b leftmost bits are zeros.

The fastest known algorithm for computing such a *partial collision* for a one-way hash function is brute force [13]. Another key property for this puzzle is that it must be verifiable by any node, without requiring any trust relation between the nodes. This is true in our crypto-puzzle, as all input parameters for the test are locally available to all nodes at the start of this phase: the nonce becomes globally known in the previous phase of our algorithm; and id is the identity of the node solving the puzzle. Verification can be easily performed by computing a hash, using the answer provided by a node, and verifying if the b leftmost bits are zeros.

3.2.2 Sybil Mitigation

As stated earlier, the candidate selection phase has a limited time for the crypto-puzzle resolution (i.e. period T) in which nodes have to answer to the crypto-puzzle. Ideally, a CRT would not allow nodes to be able to solve more than one cryptographic puzzle in T, preventing the proposal of more than one identity per node for the candidate set. However, the probabilistic nature of these tests makes the puzzle resolution time a random variable, and thus, there exists a non-zero probability that a Byzantine node is able to propose multiple identities to the candidate set. Note that, while in the following examples, T is represented in number of hashes, we can convert it into steps by taking into consideration the computational capabilities of each node.

The choice of the value for T implies a trade-off between the need to accept at least $q - f$ replies from correct nodes, with a probability arbitrarily close to 1, and the avoidance of Sybil identities proposed by Byzantine nodes. Due to the randomness of the time taken by a node to solve one puzzle through brute-force, these two objectives, as will become clear in the next paragraph, are typically incompatible.

To solve the cryptographic puzzle described above, 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. Assuming a perfect hash function, the probability of a node being able to solve ν crypto-puzzles in exactly T tries, is then given by a binomial distribution:

$$p(\nu) = \binom{T}{\nu} (p_{te})^{\nu} (1 - p_{te})^{(T-\nu)}, \tag{7}$$

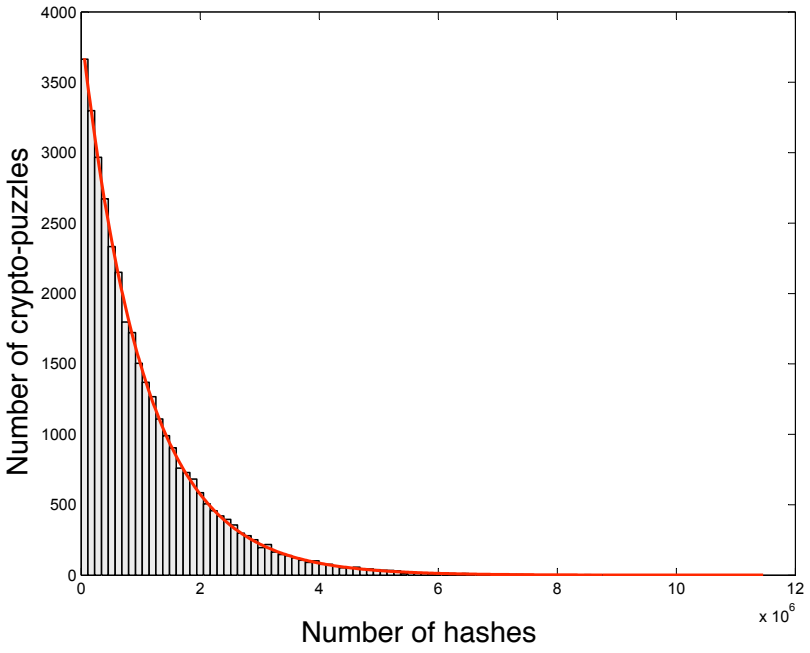

Figure 4. Distribution function and histogram, with 100 classes, of the simulated crypto-puzzle answers.

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

To validate this conclusion using a standard hash function, we created a program in Python, 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 4, where we can see both the resulting histogram, with 100 classes, and the analytic solution (Equation 7).

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 $q - f$ correct nodes to be able to reply to the crypto-puzzle. The probability p_c that the $N - f$ correct nodes are able to propose at least $q - f$ identities is given by:

$$p_c = \sum_{i=q-f}^{N-f} \binom{N-f}{i} (p(\nu \neq 0))^i (1 - p(\nu \neq 0))^{(N-f-i)}. \quad (8)$$

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, for N=50, f=4, q=13, and b=20. As shown, to achieve a probability near 1 that at least $q - f$ identities belonging to correct nodes are proposed, we allow a probability of approximately 0.2 of Sybil identities being proposed.

This clearly shows that, to increase the capability of collecting an adequate number of responses from correct nodes, it is unavoidable to also increase the probability of Sybil identities being proposed, therefore justifying the existence of *Phase 3*.

In the following Lemmas, we show the correctness of the candidate selection phase.

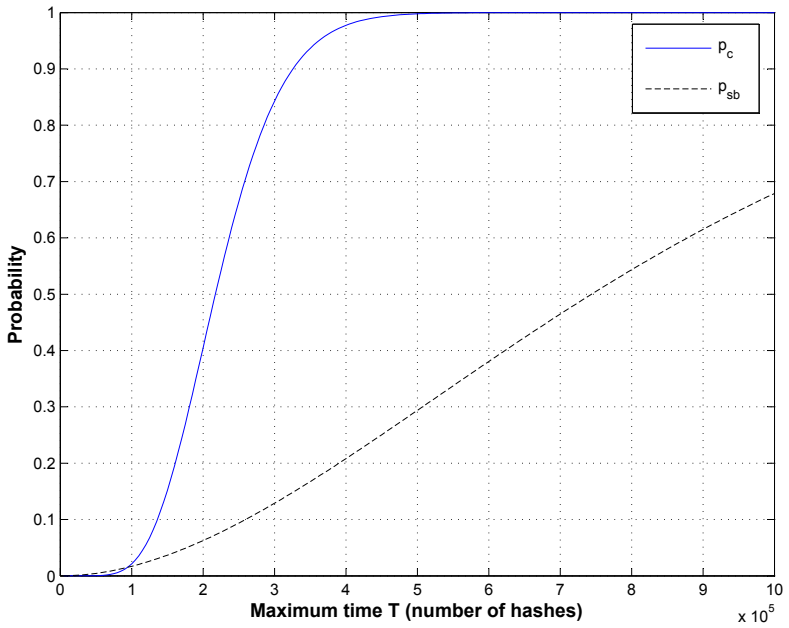

Figure 5. Probability of correct (p_c) and Sybil identities (p_{sb}) entering the network, for $N = 50$, $f = 4$, $q = 13$, $b = 20$.

Lemma 3. *Assuming the existence of a correct nonce, it is always possible to find a period T such that, with a probability arbitrarily close to 1, there will be at least $q - f$ correct identities in the output of* Phase 2. *More precisely, for any $0 < \delta < 1$, there is a $T : p_c(T) > 1 - \delta$.*

Proof. Note that $p(\nu \neq 0) = 1 - (1 - p_{te})^T$ is a strictly increasing monotonic function of T. Also, since the skewness of a binomial distribution becomes increasingly negative for increasing probability of success, the sum in (8) will monotonically increase for increasing values of $p(\nu \neq 0)$. This concludes the proof. □

Phase 3 requires a bounded number of identities to be tested. Therefore, the output of *Phase 2* must be finite.

Lemma 4. *The number of identities returned by* Phase 2 *is finite.*

Proof. Since there is a time limit on the acceptance of crypto-puzzle answers, the candidate set output is composed by the maximum number of valid identities the Byzantine nodes were able to propose, and the maximum number of identities proposed by correct nodes in the system. It follows that the total number of identities returned by *Phase 2* is finite. □

Lemma 5. *Let C_i be the output of the candidate selection phase, for a given node n_i. If there is a proposal from a correct node, P, in C_i, then, P is also present on all the outputs of every other correct node. More precisely, if $P \in C_i$ for correct node n_i, then $P \in C_j \; \forall_{n_j} \; s.t. \; n_j$ is correct.*

Proof. Assume the opposite, that there is a proposal from a correct node P, that is present in the output C_i of a correct node n_i, but not in the output C_j of a correct node n_j. However, given that every proposal is sent using the wireless medium; that, in our model, every node in the network receives the same set of messages; and that every node is able to locally check their validity, if $P \in C_i$ then necessarily, $P \in C_j$, which contradicts the assumption. □

3.3 Quorum Validation

```
// Executed at every node i
algorithm quorumValidation (C) is
   excluded ⟵ ∅;
   valid ⟵ ∅;
   I ⟵  rrt.schedule(C,K);
   for (j = 0 to rrt.length(C,K)) do
      if id_i ∈ I[j] then
         broadcast.send (I[j].channelFor(id_i), VALIDATE);
      else
         channel ⟵ rand(1,K);
         if broadcast.receive (channel) = silence then
            excluded ⟵ excluded ∪ I[j][channel];
   valid ⟵ truncate(q, sort(C \ excluded));
   if |valid| < q then
      valid ⟵ valid ∪ padding; // with void identities
   return valid;
```

Figure 6. Quorum Validation Procedure.

The quorum validation phase detects Sybil identities in a set of identities $\mathcal{C}$, through the use of a radio resource test. As noted in Section 2, the radio resource test (RRT) is another particular case of resource testing. The test, in our case, consists in requiring different identities to transmit simultaneously in different channels. Since a node cannot transmit on more than one channel simultaneously, when two or more of its identities are tested and, therefore, required to transmit simultaneously on different channels, the node cannot comply. The Sybil identities involved in that particular test will, thus, not transmit, and will eventually be detected. However, since the nodes that wish to verify a specific test are also unable to listen simultaneously in more than one channel, the test is repeated several times, ensuring that, with a configurable high probability, an existing Sybil identity is detected. However, if the number of identities being tested is higher than the number of radio channels available, we need to verify the existence of Sybil identities in every combination of available identities, further increasing the total numbers of tests required. Like most of the resource testing solutions, RRTs have the potential to support protocols that do not require pre-configuration or pre-shared secrets. Furthermore, RRTs have an additional advantage over other resource tests: to the best of our knowledge, RRTs are the only proposed type of resource test that allows identities, not participating in a specific test, to witness the outcome and validate the results without requiring trust on any participant. For further details see [12, 10].

In previous work we analysed the complexity of different RRT [10]. We shown that the cost of systematically running RRTs to check for Sybil identities among a large population of identities is prohibitively expensive, limiting the scalability of the solution. This observation further motivates the need of *Phase 2*, which allows us to greatly reduce the number of identities tested by the RRT.

The RRT is characterised by the following interface. The method "rrt.length ($\mathcal{C}$,K)" returns the number of steps required to test $|\mathcal{C}|$ identities with K radio channels. The method "rrt.schedule($\mathcal{C}$,K)" returns a list of identities, that are required to transmit on each step of the test. Channels are assigned to the participating identities, from channel 1 to channel K.

This third phase of the NSQ algorithm is depicted in Figure 6. The phase runs for a fixed number of steps defined by "rrt.length($\mathcal{C}$,K)". In each step, the node checks if it is scheduled to transmit on that step. In the affirmative case, it transmits a VALIDATE message in the channel specified by the radio resource test schedule. Otherwise, it simply randomly listens to one of the radio channels involved in the test. Note that nodes that do not belong to the candidate set always listen in all steps. If silence is detected in the channel, the identity that was

scheduled to transmit on that channel is added to an exclude list.

This phase ends by lexicographically sorting every non-excluded identity, and returning the first q identities as the final NSQ_i quorum.

3.3.1 RRT Used

Since Byzantine nodes may act in a way that causes their identities to be excluded, it is possible that at the end of the CRT less than q identities remain (i.e., $q - f \leq |\text{valid}| \leq q$). In this case, NSQ_i is just padded with void identities. This is a purely formal step, which only ensures that an upper level layer using our algorithm, sees a constant interface output (quorum with q members).

Finally, note that each node may return a different set, since the Byzantine nodes can defend different Sybil identities in different tests of the RRT. The properties of the RRT, and the set ordering, guarantee that the quorums returned in each correct node, possess the properties described in Section 2. We will further discuss these properties in Section 4.

The algorithm presented in Figure 6 assumes that all combination of all the $\mathcal{C}$ identities are tested, K at a time. As previously described, each combination should be tested several times to increase the probability of detection of Sybil identities. Due to lack of space, we are unable to reproduce here the calculations that allow us to show the value of "rrt.length($\mathcal{C}$,K)" in order to achieve the desired detection probability. These results, and a detailed description of the tests scheduling, can be found in [10].

In theory, the application of an RRT alone (without combining with other resource tests), would be sufficient to eliminate the Sybil identities of a network, with an arbitrarily high probability. In practice, this approach is unfeasible. First, the absence of obstructions to the proposal of Sybil identities (such as is the case of *Phase 2* of the NSQ algorithm), allows for two different attacks, in the case where the whole population of identities is being tested: *i*) since Byzantine nodes may propose an unlimited number of identities, they are able to increase significantly the number of tests the whole network has to execute, thereby increasing the time taken for the tests to conclude; *ii*) if every new identity is required to be validated in order to take part in the network, Byzantine nodes may continuously propose new identities, triggering new rounds of RRT tests, and thus, postponing the convergence of the algorithm.

Moreover, even without Sybil identities the overhead of testing a whole population of identities becomes unfeasible, even for a small number of identities. For example, the number of messages required to test 20 identities, in a network with $f = 5$ and a probability of Sybil detection 0.95, with the best RRT described in [10](in terms of performance) is approximately 2000 messages. This number grows quickly with the total number of identities: for 40 identities, approximately 8000 messages are required. Further details regarding the performance of RRTs will be given in Section 5.

However, RRTs have the advantage of allowing the identification of Sybil identities with a probability arbitrarily close to 1, something that makes them a fundamental building block of our algorithm. Namely, it is the RRT that ultimately guarantees the Sybil-Free property of each quorum.

Lemmas 6 and 7 provide proof of the correctness of the quorum validation phase.

Lemma 6. *No correct identity is excluded on the RRT by a correct node.*

Proof. 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 our model, which contradicts the assumption. □

Lemma 7. *There is a number of rounds r, such that there exists a probability p_d, arbitrarily close to 1, of removing any Sybil identities from the output of the RRT.*

Proof. This result was proven in [10]. □

4 Algorithm Correctness

As described in Section 2, we require the quorums returned by the NSQ algorithm to possess four distinct properties, namely: Q-Size, Probabilistic Sybil-free, Probabilistic Partial Consistency and Termination. We now present proofs for each of these four properties.

Theorem 1. *(Q-Size) Any quorum of a correct node NSQ_i has exactly q identities. More precisely $|NSQ_i| = q \; \forall_i \; s.t. \; n_i$ is correct.*

Proof. This proof follows directly from the fact that the quorum validation algorithm trims by q (padding with void identities, if necessary) the final set returned in each node. □

Theorem 2. *(Probabilistic Sybil-free) With a probability arbitrarily close to 1, in any quorum NSQ_i delivered by a correct node n_i, the number of identities that have been proposed by malicious nodes is no larger than the number f of malicious nodes in the system. More precisely, $|NSQ_i| \leq |uses(NSQ_i)|$, where $uses(NSQ_i) = \bigcup_{id_j \in NSQ_i} uses(id_j)$.*

Proof. This Lemma follows from Lemma 7. Since the last phase of the NSQ algorithm is *Phase 3*, 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 7, the output of the RRT, should not contain Sybil identities, with a probability arbitrarily close to 1. □

Theorem 3. *(Probabilistic Partial Consistency) The intersection of the quorums delivered to all correct nodes has, at least, $q - f$ identities used by correct nodes, with a probability arbitrarily close to* 1.

Proof. Let the output of the quorum validation phase of a correct node be $Q_i = Q_i^+ \cup Q_i^-$, where Q_i^+ is the set of identities proposed by correct nodes and Q_i^- is the set of identities proposed by Byzantine nodes. Let the lexicographical poset of Q_i be $\{Q_i, \prec_L\}$. Then, $\{Q_i, \prec_L\} = \{Q_i^+, \prec_L\} \circ \{Q_i^-, \prec_L\}$, where $\circ$ is the ordered composition of two posets. Given that by Lemma 6, $Q_i^+ = Q^+ \forall_i$ and, by Theorem 2, $|Q_i^-| \leq f$, then the subset of the first q elements of $\{Q_i, \prec_L\}$ contains at most f elements of $\{Q_i^-, \prec_L\}$ and at least the first $q - f$ elements of $\{Q^+, \prec_L\}$. □

Theorem 4. *Every correct node's execution of NSQ returns a quorum set in a finite number of steps.*

Proof. For each of the three phases of the NSQ algorithm, there is a finite number of steps that ensures their properties, with a arbitrarily high probability. Since the total number of steps to end the NSQ algorithm is the sum of all the steps of each phase, the algorithm is, therefore, finite. □

5 Performance evaluation and Optimisations

To evaluate our algorithm in terms of efficiency, we compare the total number of messages sent in an execution of the NSQ algorithm, with the total number of messages that sent in a direct application of the most efficient RRT, to the whole population of nodes, as proposed in [10]. Only messages sent from correct nodes are taken into consideration in this comparison, and the probabilities of success are set for 0.9999 in both cases, with $q = 16$, and $f = 5$.

In the NSQ algorithm, correct nodes transmit messages in all three phases. In *Phase 1*, correct nodes send, on average, $p_t \cdot (N - f) \cdot TS$ contributions to the nonce (this is a pessimistic approach, since it assumes that no node is able to terminate before TS). In *Phase 2*, the number of messages sent is equivalent to the number of

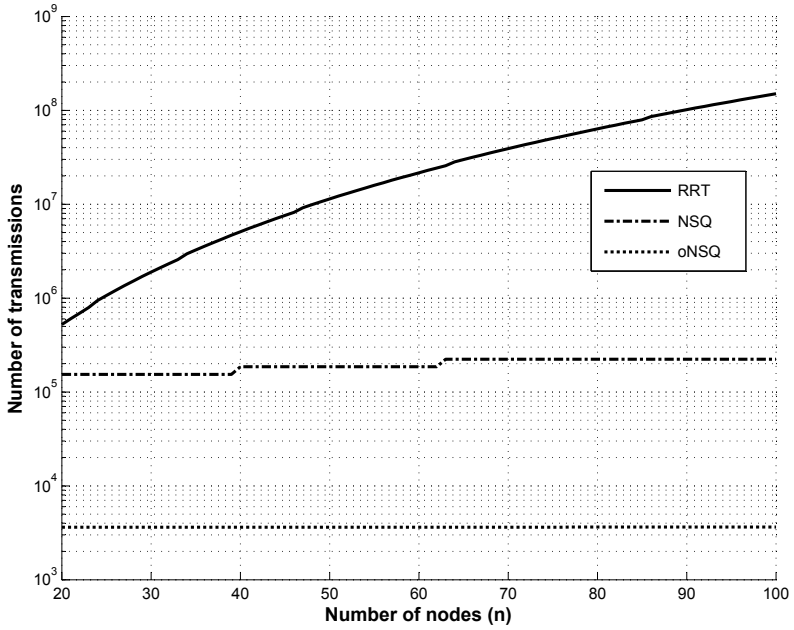

Figure 7. Nessages sent by the direct use of an RRT, the NSQ algorithm NSQ optimised version ($q = 3 \cdot f + 1, f = 5$).

crypto-puzzles solved by the $N - f$ correct nodes during T. This value is given by: $1 - (1 - p_{te})^T \cdot (N - f)$. Finally, in *Phase 3*, the number of messages sent is given by the specific schedule of the resource test [10]. This number is dependent on the total number of identities being tested. To obtain the number of identities that reaches *Phase 3*, we have to take into account the total number of identities proposed in *Phase 2*, which is given by $1 - (1 - p_{te})^T \cdot (N - f) + p_{te} \cdot T \cdot f$.

To evaluate the direct use of the RRT test, we assume that each of the f Byzantine nodes creates three Sybil identities and, thus, the total number of identities tested is given by: $N + 3 \cdot f$. Note that this is a very conservative value: in the single use of RRT there is, in fact, no bound on the number of Sybil identities that Byzantine nodes may propose; the same is not true, of course, in the present solution (NSQ algorithm), since the existence of *Phase 2* allows us to impose a bound on this number of Sybil identities.

Figure 7 depicts the total number of messages sent in each case (curves RRT and NSQ). As one can see, for reasonably sized networks, the NSQ algorithm sends a number of messages orders of magnitude lower than the direct application of the RRT.

There is still an optimisation that can be applied to further reduce the number of identities that reaches *Phase 3*, and, consequently, the total number of messages. Since T is configured to ensure that at least $q - f$ correct nodes are able to propose their identities, with an arbitrarily large probability, this number might be much larger than required. In order to reduce it, we set another stopping condition for *Phase 2*, besides T. *Phase 2* also stops whenever $\sigma = (q - f) + s$ identities are received, where s is the maximum number of identities that Byzantine nodes are able to propose in T, with an arbitrarily high probability p_{il}.

Note that, the probability of having up to s total identities proposed by the f Byzantine nodes ($p_{il}(s)$), is the sum of the number of identities proposed by each Byzantine node. It is, thus, a random variable resulting from the sum of independent binomial random variables:

$$p_{il}(s) = \sum_{i=0}^{s} \binom{f \cdot T}{i} p_{te}^{i} \cdot (1 - p_{te})^{(f \cdot T - i)}. \quad (9)$$

Therefore, by choosing s such that p_{il} is set to a large enough probability, we ensure that when σ identities are received, a minimum of $q - f$ of them are correct. In Figure 7 (curve oNSQ), we can see that this optimisation further reduces the total number of required transmissions.

Lemma 8. *There is an upper bound, $s \geq 0$ with $s \in \mathbb{N}$, on the number of Sybil identities that Byzantine nodes are able to propose in a time period T, with a probability arbitrarily close to 1 (p_{il}). More precisely, given any $0 < \delta < 1$, there is an s such that: $p_{il}(s) > 1 - \delta$.*

Proof. By choosing $\delta = p_{il}(s - 1)$ it is trivial to verify that $p_{il}(s) > \delta$, given that p_{il} is a cumulative binomial distribution on s. Also, by the same reason, when $s \to f \cdot T$, $p_{il}(s) \to 1$. □

6 Conclusions and Future Work

In this paper we proposed an algorithm to construct a partially consistent quorum of non-Sybil nodes in a one-hop wireless network. The algorithm is based on the use of a combination of different resource tests, not only due to the need of detecting and excluding Sybil identities, but also to the need of doing it efficiently. As future work, we plan on expanding this algorithm to a multi-hop network.

References

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

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

[3] J. Douceur. The sybil attack. In *IPTPS '01*, pages 251–260, London, UK, 2002. Springer-Verlag.

[4] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92*, pages 139–147, London, UK, 1993. Springer-Verlag.

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

[6] Y. Haifeng, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *S&P 2008*, pages 3–17, 2008.

[7] C.-Y. Koo, V. Bhandari, J. Katz, and N. Vaidya. Reliable broadcast in radio networks: the bounded collision case. In *PODC '06*, pages 258–264, NY, USA, 2006. ACM.

[8] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[9] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11:569–578, 1998.

[10] D. Mónica, J. Leitão, L. Rodrigues, and C. Ribeiro. On the use of radio resource tests in wireless ad hoc networks. In *Proceedings of the 3rd WRAITS*, pages F21–F26, Estoril, Portugal, jun 2009.

[11] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) identifiers and addresses. In *NDSS*. Internet Society, February 2002.

[12] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *IPSN 2004*, pages 259–268, 2004.

[13] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

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