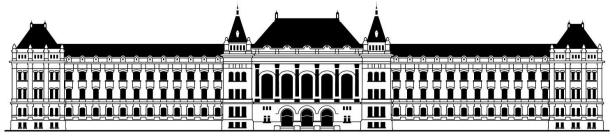


Budapest University of Technology and Economics
Department of Networked Systems and Services

New security mechanisms for wireless ad hoc and sensor networks

Collection of Habilitation Theses
by
Levente Buttyán, Ph.D.

Budapest, Hungary
2013

Contents

Introduction

This document contains new research results in the field of security and privacy in wireless ad hoc and sensor networks. Wireless ad hoc networks are self-organizing wireless networks of end-user devices, where all networking services are provided by the devices themselves without the help of any fixed infrastructure. Such networks will never replace the existing infrastructure based Internet, but they can provide a new form of wireless access, which has some advantages over traditional wireless access solutions. Wireless sensor networks represent a special application area of ad hoc networking, where the devices are tiny sensors that also have computing and wireless communication capabilities. The sensors collect measurement data from the environment, and send their data over multiple wireless hops to a set of few sink nodes, or base stations, for further processing. From the networking point of view, sensor networks are often considered to be self-organizing ad hoc networks.

While these new types of wireless networks have potentially useful applications, they also represent an interesting challenge in terms of security. The most important challenges include the lack of physical protection and the scarcity of resources. In many applications, such networks are deployed in an environment where the devices simply cannot be protected by physical means. In addition, providing tamper resistance for devices is expensive, and therefore, it is not a viable option in applications where devices must be deployed in large quantities (e.g., sensors) and hence unit cost must be kept very low. For this reason, we must assume that devices can be compromised, and we must design our security mechanisms in such a way that they do not fail in the presence of such compromised devices. For the same reason of economic viability, devices in wireless ad hoc and sensor networks are usually constrained in terms of CPU power, memory, communication range and speed, and available energy. Hence, our security mechanisms should be designed with these resource limitations in mind.

The new security mechanisms that we propose in this document satisfy the above requirements: they can tolerate compromised nodes and they also respect the resource constraints of the environment. We grouped our results into 5 thesis groups as follows:

In the first thesis group (Section 1), we study the problem of securing routing protocols in wireless ad hoc networks. First, we present new attacks on existing routing protocols. Then, we propose a mathematical framework in which security of routing can be precisely defined, and routing protocols for wireless ad hoc networks can be proved to be secure in a rigorous manner. Our framework is tailored for on-demand source routing protocols, but the general principles are applicable to other types of protocols too. We also propose a new on-demand source routing protocol, called endairA, and we demonstrate the usage of our framework by proving that it is secure in our model.

In the second thesis group (Section 2), we study another aspect of routing in wireless ad hoc networks, namely, the function of packet forwarding. As mentioned before, wireless ad hoc networks are often assumed to be fully self-organizing, where the nodes have to forward packets for each other in order to enable multi-hop communication. This requires the nodes to cooperate, but nodes may behave selfishly and jeopardize the operation of the network. Here, we study if cooperation can emerge spontaneously in static wireless ad hoc networks, without any explicit incentive mechanism. We propose a model based on game theory to investigate equilibrium conditions of packet forwarding strategies. We give the conditions under which cooperation can exist sponatneously, and we perform simulations to estimate the probability that the conditions for a cooperative equilibrium hold. We conclude that in static ad hoc networks – where the relationships between the nodes are likely to be stable – cooperation is unlikely to emerge spontaneously and it needs to be encouraged.

In the third thesis group (Section 3), we address the problem of wormhole attacks in wireless networks. A wormhole is a fast out-of-band connection between two distant physical locations,

which is established by the attacker for the purpose of tunneling traffic between those two locations. Wormholes can mislead neighbor discovery protocols, and they can have serious negative effects on routing in ad hoc networks. To address this problem, we propose three new wormhole detection mechanisms. Two of our mechanisms use a centralized approach applicable in wireless sensor networks, and they are both based on statistical hypothesis testing. Both mechanisms assume that the sensors send their neighbor list to the base station, and it is the base station that runs the wormhole detection algorithm on the network graph that is reconstructed from the received neighborhood information. Our third wormhole detection mechanism follows a decentralized approach applicable in any ad hoc network, where pairs of nodes can detect locally if they are connected via a wormhole by using our proposed authenticated distance bounding protocol.

In the fourth thesis group (Section 4), we address the problem of pollution attacks in coding based distributed storage systems proposed for wireless sensor networks. In a pollution attack, the adversary maliciously alters some of the stored encoded packets, which results in the incorrect decoding of a large part of the original data upon retrieval. We propose algorithms to detect and recover from such attacks and we study the performance of the proposed algorithms in terms of communication and computing overhead, and in terms of success rate. In contrast to existing approaches to solve this problem, our approach is not based on adding cryptographic checksums or signatures to the encoded packets; rather, we take advantage of the inherent redundancy in such distributed storage systems.

Finally, in the fifth thesis group (Section 5), we study the problem of efficient privacy preserving authentication in resource constrained environments, such as sensor networks or RFID systems. More specifically, we improve an approach that was proposed earlier by others. This approach uses key-trees, and its basic problem is that the level of privacy provided by the system to its members decreases considerably if some members are compromised. We analyze this problem, and show that careful design of the key-tree can help to minimize this loss of privacy. First, we introduce a benchmark metric for measuring the resistance of the system to a single compromised member. This metric is based on the well-known concept of anonymity sets. Then, we show how the parameters of the key-tree should be chosen in order to maximize the system's resistance to single member compromise under some constraints on the authentication delay. In the general case, when any member can be compromised, we give a lower bound on the level of privacy provided by the system. We also present some simulation results that show that this lower bound is sharp.

1 Securing on-demand source routing in wireless ad hoc networks

THESIS GROUP 1. *I propose new, previously unknown attacks on existing ad hoc network routing protocols. I propose a novel modeling framework that allows for a precise definition of routing security, and a corresponding proof technique that can be used to argue about the security of rotuing protocols. I propose endairA, a new on-demand source routing protocol for ad hoc networks and prove formally that it is secure in the proposed model. [C4, J1]*

Routing is one of the most basic networking functions in wireless ad hoc networks. Hence, an adversary can easily paralyze the operation of the network by attacking the routing protocol. This has been realized by many researchers, and several "secure" routing protocols have been proposed for ad hoc networks (see [23] for a survey). However, the security of those protocols have been analyzed either by informal means only, or with formal methods that have never been intended for the analysis of this kind of protocols (e.g., BAN logic [9]).

In this thesis group, we present new attacks on exisiting "secure" routing protocols, which clearly demonstrate that flaws can be very subtle, and therefore, hard to discover by informal reasoning. Hence, we advocate a more systematic approach to analyzing ad hoc routing protocols, which is based on a rigorous mathematical model, in which precise definitions of security can be given, and sound proof techniques can be developed.

Routing has two main functions: route discovery and packet forwarding. The former is concerned with discovering routes between nodes, whereas the latter is about sending data packets through the previously discovered routes. There are different types of ad hoc routing protocols. One can distinguish proactive (e.g., OLSR [14]) and reactive (e.g., AODV [34] and DSR [26]) protocols. Protocols of the latter category are also called on-demand protocols. Another type of classification distinguishes routing table based protocols (e.g., AODV) and source routing protocols (e.g., DSR). In this work, *we focus on the route discovery part of on-demand source routing protocols.* However, in [1], we show that the general principles of our approach are applicable to the route discovery part of other types of protocols too.

At a very informal level, security of a routing protocol means that it can perform its functions even in the presence of an adversary whose objective is to prevent the correct functioning of the protocol. Since we are focusing on the route discovery part of on-demand source routing protocols, in our case, attacks are aiming at achieving that honest nodes receive "incorrect" routes as a result of the route discovery procedure. We will make it more precise later what we mean by an "incorrect" route.

Regarding the capabilities of the adversary, we assume that it can mount active attacks (i.e., it can eavesdrop, modify, delete, insert, and replay messages). However, we make the realistic assumption that the adversary is not all powerful, by which we mean that it cannot eavesdrop, modify, or control all communications of the honest participants. Instead, the adversary launches its attacks from a few adversarial nodes that have similar communication capabilities to the nodes of the honest participants in the network. This means that the adversary can receive only those messages that were transmitted by one of its neighbors, and its transmissions can be heard only by its neighbors. The adversarial nodes may be connected through proprietary, out-of-band channels and share information. We further assume that the adversary has compromised some identifiers, by which we mean that it has compromised the cryptographic keys that are used to authenticate those identifiers. Thus, the adversary can appear as an honest participant under any of these compromised identities.

The mathematical framework that we introduce is based on the so called *simulation paradigm* [5, 36], which has already been used extensively for the analysis of key establishment protocols, but we are the first who apply it in the context of ad hoc routing. We also propose a new on-

demand source routing protocol, called endairA, and we demonstrate the usage of our framework by proving that it is secure in our model.

1.1 New attacks on existing protocols

THESIS 1.1. *I analysed two previously proposed secure ad hoc network routing protocols SRP [33] and Ariadne [24]. As a result of this analysis, I discovered new, previously unknown attacks against both protocols. More specifically, I discovered an attack on SRP, an attack on Ariadne when used with MACs, an attack on Ariadne when used with digital signatures, and an attack on an optimized version of Ariadne. In all of these attacks, the attacker is able to force the acceptance of a non-existent route with the initiator of the route discovery procedure of the routing protocol. [C4, J1]*

Due to space limits, here we present only one of the discovered attacks. The interested reader can find the description of the other attacks in [12] and [2].

Operation of the Ariadne protocol

Ariadne has been proposed in [24] as a secure on-demand source routing protocol for ad hoc networks. Ariadne comes in three different flavors corresponding to three different techniques for data authentication. More specifically, authentication of routing messages in Ariadne can be based on TESLA [35], on digital signatures, or on MACs. We discuss Ariadne with digital signatures.

The initiator of the route discovery generates a route request message and broadcasts it to its neighbors. The route discovery message contains the identifiers of the initiator and the target, a randomly generated request identifier, and a MAC computed over these elements with a key shared by the initiator and the target. This MAC is hashed iteratively by each intermediate node together with its own identifier using a publicly known one-way hash function. The hash values computed in this way are called per-hop hash values. Each intermediate node that receives the request for the first time re-computes the per-hop hash value, appends its identifier to the list of identifiers accumulated in the request, and generates a digital signature on the updated request. Finally, the signature is appended to a signature list in the request, and the request is re-broadcast.

When the target receives the request, it verifies the per-hop hash by re-computing the initiator's MAC and the per-hop hash value of each intermediate node. Then it verifies all the digital signatures in the request. If all these verifications are successful, then the target generates a route reply and sends it back to the initiator via the reverse of the route obtained from the route request. The route reply contains the identifiers of the target and the initiator, the route and the list of digital signatures obtained from the request, and the digital signature of the target on all these elements. Each intermediate node passes the reply to the next node on the route (towards the initiator) without any modifications. When the initiator receives the reply, it verifies the digital signature of the target and the digital signatures of the intermediate nodes (for this it needs to reconstruct the requests that the intermediate nodes signed). If the verifications are successful, then it accepts the route returned in the reply.

An attack on Ariadne

Let us consider Figure 1, which illustrates part of a configuration where the discovered attack is possible. The attacker is denoted by A. Let us assume that S sends a route request towards D. The request reaches V that re-broadcasts it. Thus, A receives the following route request

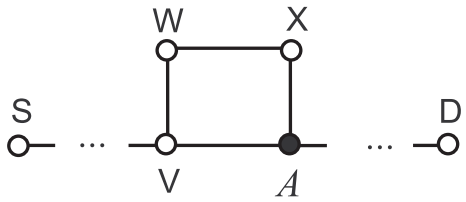


Figure 1: Part of a configuration where an attack against Ariadne is possible

message:

$$msg_1 = (\mathsf{rreq},\ \mathsf{S},\ \mathsf{D},\ id,\ h_{\mathsf{V}},\ (\ldots,\mathsf{V}),(\ldots,sig_{\mathsf{V}}))$$

where id is the random request identifier, h_{V} is the per-hop hash value generated by V, and sig_{V} is the signature of V. A does *not* re-broadcast msg_1. Later, A receives another route request from X:

$$\begin{aligned} msg_2 \ &= \ (\mathsf{rreq},\ \mathsf{S},\ \mathsf{D},\ id,\ h_{\mathsf{X}},\ (\ldots,\mathsf{V},\mathsf{W},\mathsf{X}),\\ &\quad\ (\ldots,sig_{\mathsf{V}},sig_{\mathsf{W}},sig_{\mathsf{X}})) \end{aligned}$$

From msg_2, A knows that W is a neighbor of V. A computes $h_A = H(A, H(\mathsf{W}, h_{\mathsf{V}}))$, where h_{V} is obtained from msg_1, and H is the publicly known hash function used in the protocol. A obtains the signatures $\ldots, sig_{\mathsf{V}}, sig_{\mathsf{W}}$ from msg_2. Then, A generates and broadcasts the following request:

$$\begin{aligned} msg_3 \ &= \ (\mathsf{rreq},\ \mathsf{S},\ \mathsf{D},\ id,\ h_A,\ (\ldots,\mathsf{V},\mathsf{W},A),\\ &\quad\ (\ldots,sig_{\mathsf{V}},sig_{\mathsf{W}},sig_A)) \end{aligned}$$

Later, D generates the following route reply and sends it back towards S:

$$\begin{aligned} msg_4 \ &= \ (\mathsf{rrep},\ \mathsf{D},\ \mathsf{S},\ (\ldots,\mathsf{V},\mathsf{W},A,\ldots),\\ &\quad\ (\ldots,sig_V,sig_W,sig_A,\ldots),\ sig_{\mathsf{D}}) \end{aligned}$$

When A receives this route reply, it forwards it to V in the name of W. Finally, S will output the route $(\mathsf{S},\ldots,\mathsf{V},\mathsf{W},A,\ldots,\mathsf{D})$, which is a non-existent route.

1.2 The proposed analysis framework

THESIS 1.2. *I propose a novel modeling framework that allows for a precise definition of routing security and rigorous proofs about the security of routing protocols. This model is based on the simulation paradigm known from the cryptographic literature, but I am the first to apply it for the analysis of ad hoc network routing protocols. I define the elements of the model and a corresponding proof technique that can be used in practice. Using the framework, I formally define what security of the route discovery part of on-demand source routing protocols mean. [J1]*

The attacks we discovered clearly show that security flaws in ad hoc routing protocols can be very subtle. Consequently, making claims about the security of a routing protocol based on informal arguments only is dangerous. Hence, we propose a mathematical framework, which allows us to define the notion of routing security precisely and to prove that a protocol satisfies our definition of security. It is important to emphasize that the proposed framework is best suited for proving that a protocol is secure (if it really is), but it is not directly usable to

discover attacks against routing protocols that are flawed. We note, however, that such attacks may be discovered indirectly by attempting to prove that the protocol is secure, and examining where the proof fails.

Our framework is based on the simulation paradigm [5, 36]. In this approach, two models are constructed for the protocol under investigation: a *real-world model*, which describes the operation of the protocol with all its details in a particular computational model, and an *ideal-world model*, which describes the protocol in an abstract way mainly focusing on the services that the protocol should provide. One can think of the ideal-world model as a description of a specification, and the real-world model as a description of an implementation. Both models contain adversaries. The real-world adversary is an arbitrary process, while the abilities of the ideal-world adversary are usually constrained. The ideal-world adversary models the *tolerable imperfections* of the system; these are attacks that are unavoidable or very costly to defend against, and hence, they should be tolerated instead of being completely eliminated. The protocol is said to be secure if the real-world and the ideal-world models are equivalent, where the equivalence is defined as some form of indistinguishability (e.g., statistical or computational) from the point of view of the honest protocol participants. Technically, security of the protocol is proven by showing that the effects of any real-world adversary on the execution of the real protocol can be *simulated* by an appropriately chosen ideal-world adversary in the ideal-world model.

Configurations and plausible routes

We model the ad hoc network (in a given instance of time) as an undirected graph $G(V, E)$, where V is the set of vertices, and E is the set of edges. Each vertex represents either a single non-adversarial node, or a set of adversarial nodes that can share information among themselves by communicating via direct wireless links or via out-of-band channels. The former is called a non-adversarial vertex, while the latter is called an adversarial vertex. The set of adversarial vertices is denoted by V^*, and $V^* \subset V$.

There is an edge between two non-adversarial vertices if the corresponding non-adversarial nodes established a wireless link between themselves by successfully running the neighbor discovery protocol. Furthermore, there is an edge between a non-adversarial vertex u and an adversarial vertex v^* if the non-adversarial node that corresponds to u established a wireless link with at least one of the adversarial nodes that correspond to v^*. Finally, there is no edge between two adversarial vertices in G. The rationale is that edges represent direct wireless links, and if two adversarial vertices u^* and v^* were connected, then there would be at least two adversarial nodes, one corresponding to u^* and the other corresponding to v^*, that could communicate with each other directly. That would mean that the adversarial nodes in u^* and v^* could share information via those two connected nodes, and thus, they should belong to a single vertex in G.

We assume that the adversary has compromised some identifiers, by which we mean that the adversary has compromised the cryptographic keys that are necessary to authenticate those identifiers. We assume that all the compromised identifiers are distributed to all the adversarial nodes, and they are used in the neighbor discovery protocol and in the routing protocol. On the other hand, we assume that each non-adversarial node uses a single and unique identifier, which is not compromised. We denote the set of all identifiers by L, and the set of the compromised identifiers by L^*.

Let $\mathcal{L} : V \to 2^L$ be a labelling function, which assigns to each vertex in G a set of identifiers in such a way that for every vertex $v \in V \setminus V^*$, $\mathcal{L}(v)$ is a singleton, and it contains the non-compromised identifier $\ell \in L \setminus L^*$ that is used by the non-adversarial node represented by vertex v; and for every vertex $v \in V^*$, $\mathcal{L}(v)$ contains *all* the compromised identifiers in L^*.

A *configuration* is a triplet $(G(V,E), V^*, \mathcal{L})$. Figure 2 illustrates a configuration, where the solid black vertices are the vertices in V^*, and each vertex is labelled with the set of identifiers that $\mathcal{L}$ assigns to it. Note that the vertices in V^* are not neighboring.

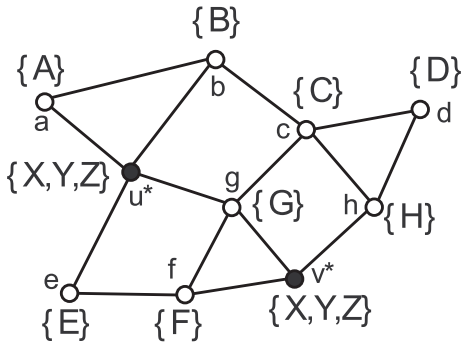


Figure 2: Illustration of a configuration. Adversarial vertices u^* and v^* are represented by solid black dots. Labels on the vertices are identifiers used by the corresponding nodes. Note that adversarial vertices are not neighboring.

We make the assumption that the configuration is static (at least during the time interval that is considered in the analysis). Thus, we view the route discovery part of the routing protocol as a distributed algorithm that operates on this static configuration.

Now, we make it more precise what we mean by an existing route. If there was no adversary, then a sequence $\ell_1, \ell_2, \ldots, \ell_n$ $(n \geq 2)$ of identifiers would be an existing route given that each of the identifiers $\ell_1, \ell_2, \ldots, \ell_n$ are different, and there exists a sequence $v_1, v_2, \ldots, v_n$ of vertices in V such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < n$ and $\mathcal{L}(v_i) = \{\ell_i\}$ for all $1 \leq i \leq n$. However, the situation is more complex due to the adversary that can use all the compromised identifiers in L^*. Essentially, we must take into account that the adversary can always extend any route that passes through an adversarial vertex with any sequence of compromised identifiers. This is a fact that our definition of security must tolerate, since otherwise we cannot hope that any routing protocol will satisfy it. This observation leads to the following definition:

Definition 1.1 (Plausible route)**.** Let $(G(V,E), V^*, \mathcal{L})$ be a configuration. A sequence $\ell_1, \ell_2, \ldots, \ell_n$ of identifiers is a plausible route with respect to $(G(E,V), V^*, \mathcal{L})$ if each of the identifiers $\ell_1, \ell_2, \ldots, \ell_n$ is different, and there exists a sequence $v_1, v_2, \ldots, v_k$ $(2 \leq k \leq n)$ of vertices in V and a sequence $j_1, j_2, \ldots, j_k$ of positive integers such that

1. $j_1 + j_2 + \ldots + j_k = n$,
2. $\{\ell_{J_i+1}, \ell_{J_i+2}, \ldots, \ell_{J_i+j_i}\} \subseteq \mathcal{L}(v_i)$ $(1 \leq i \leq k)$, where $J_i = j_1 + j_2 + \ldots + j_{i-1}$ if $i > 1$ and $J_i = 0$ if $i = 1$,
3. $(v_i, v_{i+1}) \in E$ $(1 \leq i < k)$.

Intuitively, the definition above requires that the sequence $\ell_1, \ell_2, \ldots, \ell_n$ of identifiers can be partitioned into k sub-sequences of length j_i (condition 1) in such a way that each of the resulting partitions is a subset of the identifiers assigned to a vertex in V (condition 2), and in addition, these vertices form a path in G (condition 3).

Real-world model

Next, we need to define a computational model that can be used to represent the possible executions of the route discovery part of the routing protocol. The real-world model that corresponds to a configuration $conf = (G(V, E), V^*, \mathcal{L})$ and adversary $\mathcal{A}$ is denoted by $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$, and it is illustrated on the left side of Figure 3. $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ consists of a set $\{M_1, \ldots, M_n, A_1, \ldots, A_m, H, C\}$ of interacting Turing machines, where the interaction is realized via common tapes. Each M_i represents a non-adversarial vertex in $V \setminus V^*$ (more precisely the corresponding non-adversarial node), and each A_j represents an adversarial vertex in V^* (more precisely the corresponding adversarial nodes). H is an abstraction of higher-layer protocols run by the honest parties, and C models the radio links represented by the edges in E. All machines apart from H are probabilistic.

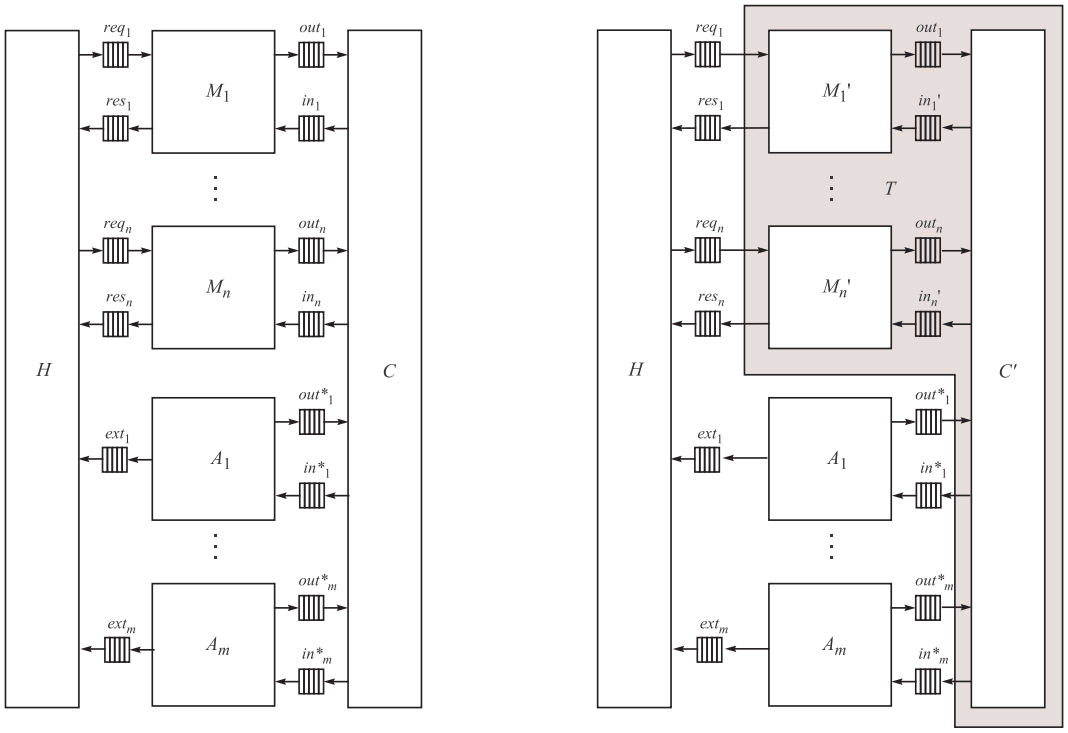


Figure 3: Interconnection of the machines in $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ (on the left side) and in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ (on the right side)

Each machine is initialized with some input data, which determines its initial state. In addition, the probabilistic machines also receive some random input (the coin flips to be used during the operation). Once the machines have been initialized, the computation begins. The machines operate in a reactive manner, which means that they need to be activated in order to perform some computation. When a machine is activated, it reads the content of its input tapes, processes the received data, updates its internal state, writes some output on its output tapes, and goes back to sleep (i.e., starts to wait for the next activation). Reading a message from an input tape removes the message from the tape, while writing a message on an output tape means that the message is appended to the current content of the tape. Note that each tape is considered as an output tape for one machine and an input tape for another machine. The machines are activated in *rounds* by a hypothetic *scheduler* (not illustrated in Figure 3). In each round, the scheduler activates the machines in the following order: $A_1, \ldots, A_m, H, M_1, \ldots, M_n, C$. In fact, the order of activation is not important, apart from the requirement that C must be activated at the end of the round. Thus, the round ends when C goes back to sleep.

Machine C is intended to model the broadcast nature of radio communications. Its task is to read the content of the output tape of each machine M_i and A_j and copy it on the input tapes of *all* the neighboring machines, where the neighbor relationship is determined by the configuration *conf*. Clearly, in order for C to be able to work, it needs to be initialized with some random input, denoted by r_C, and configuration *conf*.

Machine H models higher-layer protocols (i.e., protocols above the routing protocol) and ultimately the end-users of the non-adversarial devices. H can initiate a route discovery process at any machine M_i by placing a request (c_i, ℓ_{tar}) on tape req_i, where c_i is a sequence number used to distinguish between different requests sent to M_i, and $\ell_{tar} \in L$ is the identifier of the target of the discovery. A response to this request is eventually returned via tape res_i. The response has the form $(c_i, routes)$, where c_i is the sequence number of the corresponding request, and *routes* is the set of routes found. In some protocols, *routes* is always a singleton, in others it may contain several routes. If no route is found, then $routes = \emptyset$.

In addition to req_i and res_i, H can access the tapes ext_j. These tapes model an out-of-band channel through which the adversary can instruct the honest parties to initiate route discovery processes. The messages read from ext_j have the form (ℓ_{ini}, ℓ_{tar}), where $\ell_{ini}, \ell_{tar} \in L$ are the identifiers of the initiator and the target, respectively, of the route discovery requested by the adversary. When H reads (ℓ_{ini}, ℓ_{tar}) from ext_j, it places a request (c_i, ℓ_{tar}) in req_i where i is the index of the machine M_i that has identifier ℓ_{ini} assigned to it (see also the description of how the machines M_i are initialized). In order for this to work, H needs to know which identifier is assigned to which machine M_i; it receives this information as an input in the initialization phase.

The set of machines M_i $(1 \leq i \leq n)$ represent the non-adversarial vertices in $V \setminus V^*$. The operation of M_i is essentially defined by the routing algorithm. M_i communicates with H via its input tape req_i and its output tape res_i. Through these tapes, it receives requests from H for initiating route discoveries and sends the results of the discoveries to H, as described above.

M_i communicates with the other protocol machines via its output tape out_i and its input tape in_i. Both tapes can contain messages of the form $(sndr, rcvr, msg)$, where $sndr \in L$ is the identifier of the sender, $rcvr \in L \cup \{*\}$ is the identifier of the intended receiver ($*$ meaning a broadcast message), and $msg \in \mathcal{M}$ is the actual protocol message. Here, $\mathcal{M}$ denotes the set of all possible protocol messages, which is determined by the routing protocol under investigation.

When M_i is activated, it first reads the content of req_i. For each request (c_i, ℓ_{tar}) received from H, it generates a route request msg, updates its internal state according to the routing protocol, and then, it places the message $(\mathcal{L}(M_i), *, msg)$ on out_i, where $\mathcal{L}(M_i)$ denotes the identifier assigned to machine M_i.

When all the requests found on req_i have been processed, M_i reads the content of in_i. For each message $(sndr, rcvr, msg)$ found on in_i, M_i checks if $sndr$ is its neighbor and $rcvr \in \{\mathcal{L}(M_i), *\}$. If these verifications fail, then M_i ignores msg. Otherwise, M_i processes msg and updates its internal state. The way this is done depends on the particular routing protocol in question.

The set of machines A_j $(1 \leq j \leq m)$ represent the adversarial vertices in V^*. Regarding its communication capabilities, A_j is identical to any machine M_i, which means that it can read from in_j^* and write on out_j^* much in the same way as M_i can read from and write on in_i and out_i, respectively.

While its communication capabilities are similar to that of the non-adversarial machines, A_j may not follow the routing protocol faithfully. In fact, we place no restrictions on the operation of A_j apart from being polynomial-time in the security parameter (e.g., the key size of the cryptographic primitives used in the protocol) and in the size of the network (i.e., the number of vertices). This allows us to consider arbitrary attacks during the analysis. In particular, A_j may

delay or delete messages that it would send if it followed the protocol faithfully. In addition, it can modify messages and generate fake ones.

In addition, A_j may send out-of-band requests to H by writing on ext_j as described above. This gives the power to the adversary to specify who starts a route discovery process and towards which target. Here, we make the restriction that the adversary initiates a route discovery only between non-adversarial machines, or in other words, for each request (ℓ_{ini}, ℓ_{tar}) that A_j places on ext_j, $\ell_{ini}, \ell_{tar} \in L \setminus L^*$ holds.

Note that each A_j can write several requests on ext_j, which means that we allow several parallel runs of the routing protocol. On the other hand, we restrict each A_j to write on ext_j only once, at the very beginning of the computation (i.e., before receiving any messages from other machines). This essentially means that we assume that the adversary is *non-adaptive*; it cannot initiate new route discoveries as a function of previously observed messages.

As it can be seen from the description above, each M_i should know its own assigned identifier, and those of its neighbors in G. M_i receives these identifiers in the initialization phase. Similarly, each A_j receives the identifiers of its neighbors and the set L^* of compromised identifiers.

In addition, the machines may need some cryptographic material (e.g., public and private keys) depending on the routing protocol under investigation. We model the distribution of this material as follows. We assume a function I, which takes only random input r_I, and it produces a vector $I(r_I) = (\kappa_{pub}, \kappa_1, \ldots, \kappa_n, \kappa^*)$. The component κ_{pub} is some public information that becomes known to all A_j and all M_i. κ_i becomes known only to M_i $(1 \leq i \leq n)$, and κ^* becomes known to all A_j $(1 \leq j \leq m)$. Note that the initialization function can model the out-of-band exchange of initial cryptographic material of both asymmetric and symmetric cryptosystems. In the former case, κ_{pub} contains the public keys of all machines, while κ_i contains the private key that corresponds to the non-compromised identifier $\mathcal{L}(M_i)$, and κ^* contains the private keys corresponding to the compromised identifiers in L^*. In the latter case, κ_{pub} is empty, κ_i contains the symmetric keys known to M_i, and κ^* contains the symmetric keys known to the adversary (i.e., all A_j).

Finally, all M_i and all A_j receive some random input in the initialization phase. The random input of M_i is denoted by r_i, and that of A_j is denoted by r_j^*.

The computation ends when H reaches one of its final states. This happens when H receives a response to each of the requests that it placed on the tapes req_i $(1 \leq i \leq n)$. The output of $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ is the sets of routes found in these responses. We will denote the output by $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r)$, where $r = (r_I, r_1, \ldots, r_n, r_1^*, \ldots, r_m^*, r_C)$. In addition, $Out^{\mathsf{real}}_{conf,\mathcal{A}}$ will denote the random variable describing $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r)$ when r is chosen uniformly at random.

Ideal-world model

The ideal-world model that corresponds to a configuration $conf = (G(V,E), V^*, \mathcal{L})$ and adversary $\mathcal{A}$ is denoted by $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$, and it is illustrated on the right side of Figure 3. One can see that the ideal-world model is very similar to the real-world one. Just like in the real-world model, here as well, the machines are interactive Turing machines that operate in a reactive manner, and they are activated by a hypothetic scheduler in rounds. The tapes work in the same way as they do in the real-world model. There is only a small (but important) difference between the operation of M_i' and M_i, and that of C' and C. Below, we will focus on this difference.

Our notion of security is related to the requirement that the routing protocol should return only plausible routes. The differences between the operation of M_i' and M_i, and C' and C, will ensure that this requirement is always satisfied in the ideal-world model. In fact, the ideal-world model is meant to be ideal exactly in this sense.

The main idea is the following: Since C' is initialized with $conf$, it can easily identify and

mark those route reply messages that contain non-plausible routes. A marked route reply is processed by each machine M_i' in the same way as a non-marked one (i.e., the machines ignore the marker) except for the machine that initiated the route discovery process to which the marked route reply belongs. The initiator first performs all the verifications on the route reply that the routing protocol requires, and if the message passes all these verifications, then it also checks if the message is marked as non-plausible. If so, then it drops the message, otherwise it continues processing (e.g., returns the received route to H). This ensures that in the ideal-world model, every route reply that contains a non-plausible route is caught and filtered out by the initiator of the route discovery[1].

Before the computation begins, each machine is initialized with some input data. This is done in the same way as in the real-world model. The computation ends when H reaches one of its final states. This happens when H receives a response to each of the requests that it placed on the tapes req_i $1 \leq i \leq n$. The output of $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ is the sets of routes returned in these responses. We will denote the output by $Out^{\mathsf{ideal}}_{conf,\mathcal{A}}(r)$, where $r = (r_I, r_1, \ldots, r_n, r_1^*, \ldots, r_m^*, r_C)$. $Out^{\mathsf{ideal}}_{conf,\mathcal{A}}$ will denote the random variable describing $Out^{\mathsf{ideal}}_{conf,\mathcal{A}}(r)$ when r is chosen uniformly at random.

1.3 Definition of routing security

Now, we are ready to introduce our definition of secure routing:

Definition 1.2 (Statistical security)**.** A routing protocol is said to be statistically secure if, for any configuration *conf* and any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{A}'$, such that $Out^{\mathsf{real}}_{conf,\mathcal{A}} \stackrel{\mathsf{s}}{=} Out^{\mathsf{ideal}}_{conf,\mathcal{A}'}$, where $\stackrel{\mathsf{s}}{=}$ means "statistically indistinguishable" [2].

Intuitively, statistical security of a routing protocol means that the effect of any real-world adversary in the real-world model can be *simulated* "almost perfectly" by an ideal-world adversary in the ideal-world model. Since, by definition, no ideal-world adversary can achieve that a non-plausible route is accepted in the ideal-world model, it follows that no real-world adversary can exist that can achieve that a non-plausible route is accepted with non-negligible probability in the real-world model, because if such a real-world adversary existed, then no ideal-world adversary could simulate it "almost perfectly". In other words, if a routing protocol is statistically secure, then it can return non-plausible routes only with negligible probability in the real-world model. This negligible probability is related to the fact that the adversary can always forge the cryptographic primitives (e.g., generate a valid digital signature) with a very small probability.

1.4 Proof technique

In order to prove the security of a given routing protocol, one has to find the appropriate ideal-world adversary $\mathcal{A}'$ for any real-world adversary $\mathcal{A}$ such that Definition 1.2 is satisfied. Due to the constructions of our models, a natural candidate is $\mathcal{A}' = \mathcal{A}$. This is because for any configuration *conf*, the operation of $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ can easily be *simulated* by the operation of $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ assuming that the two systems were initialized with the same random input r. In order

[1] Of course, marked route reply messages can also be dropped earlier during the execution of the protocol for other reasons. What we mean is that if they are not caught earlier, then they are surely removed at latest by the initiator of the route discovery to which they belong.

[2] Two random variables are statistically indistinguishable if the L_1 distance of their distributions is negligibly small. In fact, it is possible to give a weaker definition of security, where instead of statistical indistinguishability, we require computational indistinguishability. Two random variables are computationally indistinguishable if no feasible algorithm can distinguish their samples (although their distribution may be completely different). Clearly, statistical indistinguishability implies computational indistinguishability, but not vice versa, therefore, computational security is a weaker notion. Here, we will only use the concept of statistical security.

to see this, let us assume for a moment that no message is dropped due to its plausibility flag being false in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$. In this case, $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ and $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ are essentially identical, meaning that in each step, the state of the corresponding machines and the content of the corresponding tapes are the same (apart from the plausibility flags attached to the messages in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$). Since the two systems are identical, $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r) = Out^{\mathsf{ideal}}_{conf,\mathcal{A}}(r)$ holds for every r, and thus, we have $Out^{\mathsf{real}}_{conf,\mathcal{A}} \stackrel{\mathsf{s}}{=} Out^{\mathsf{ideal}}_{conf,\mathcal{A}}$.

However, if some route reply messages are dropped in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ due to their plausibility flags being set to false, then $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ and $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ may end up in different states and their further steps may not match each other, since those messages are not dropped in $Sys^{\mathsf{real}}_{conf,\mathcal{A}}$ (by definition, they have already successfully passed all verifications required by the routing protocol). We call this situation a *simulation failure*. In case of a simulation failure, it might be that $Out^{\mathsf{real}}_{conf,\mathcal{A}}(r) \neq Out^{\mathsf{ideal}}_{conf,\mathcal{A}}(r)$. Nevertheless, the definition of statistical security can still be satisfied, if simulation failures occur only with negligible probability. Hence, when trying to prove statistical security, one tries to prove that for any configuration *conf* and adversary $\mathcal{A}$, the event of dropping a route reply in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ due to its plausibility flag being set to false can occur only with negligible probability.

Note that if the above statement cannot be proven, then the protocol can still be secure, because it might be possible to prove the statement for another ideal-world adversary $\mathcal{A}' \neq \mathcal{A}$. In practice, however, failure of a proof in the case of $\mathcal{A}' = \mathcal{A}$ usually indicates a problem with the protocol, and often, one can construct an attack by looking at where the proof failed.

1.5 endairA: a provably secure on-demand source routing protocol

THESIS 1.3. *I propose a new on-demand source routing protocol for ad hoc networks, called endairA, and I prove (Theorem 1.1), using the above defined mathematical framework, that it is secure. [J1]*

Inspired by Ariadne with digital signatures, we designed a routing protocol that can be proven to be statistically secure according to the definition above. We call the protocol endairA (which is the reverse of Ariadne), because instead of signing the route request, we propose that intermediate nodes should sign the route reply. Here, we describe the operation of the basic endairA protocol, and we prove it to be statistically secure.

The operation and the messages of endairA are illustrated in Figure 4. In endairA, the initiator of the route discovery process generates a route request, which contains the identifiers of the initiator and the target, and a randomly generated request identifier. Each intermediate node that receives the request for the first time appends its identifier to the route accumulated so far in the request, and re-broadcasts the request. When the request arrives to the target, it generates a route reply. The route reply contains the identifiers of the initiator and the target, the accumulated route obtained from the request, and a digital signature of the target on these elements. The reply is sent back to the initiator on the reverse of the route found in the request. Each intermediate node that receives the reply verifies that its identifier is in the node list carried by the reply, and that the preceding identifier (or that of the initiator if there is no preceding identifier in the node list) and the following identifier (or that of the target if there is no following identifier in the node list) belong to neighboring nodes. Each intermediate node also verifies that the digital signatures in the reply are valid and that they correspond to the following identifiers in the node list and to the target. If these verifications fail, then the reply is dropped. Otherwise, it is signed by the intermediate node, and passed to the next node on the route (towards the initiator). When the initiator receives the route reply, it verifies if the first identifier in the route carried by the reply belongs to a neighbor. If so, then it verifies all

the signatures in the reply. If all these verifications are successful, then the initiator accepts the route.

$S \to *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ ())$
$A \to *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ (A))$
$B \to *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ (A, B))$
$T \to B$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T))$
$B \to A$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T, sig_B))$
$A \to S$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T, sig_B, sig_A))$

Figure 4: An example for the operation and messages of endairA. The initiator of the route discovery is S, the target is T, and the intermediate nodes are A and B. id is a randomly generated request identifier. sig_A, sig_B, and sig_T are digital signatures of A, B, and T, respectively. Each signature is computed over the message fields (including the signatures) that precede the signature.

The proof of the following theorem illustrates how the framework introduced in Section 1.2 can be used in practice.

Theorem 1.1. *endairA is statistically secure if the signature scheme is secure against chosen message attacks.*

Proof. We provide only a sketch of the proof. We want to show that for any configuration $conf = (G(V, E), V^*, \mathcal{L})$ and any adversary $\mathcal{A}$, a route reply message in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ is dropped due to its plausibility flag set to false with negligible probability.

In what follows, we will refer to non-adversarial machines with their identifiers. Let us suppose that the following route reply is received by a non-adversarial machine ℓ_{ini} in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$:

$$msg = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_1}))$$

Let us suppose that msg passes all the verifications required by endairA at ℓ_{ini}, which means that all signatures in msg are correct, and ℓ_{ini} has a neighbor that uses the identifier ℓ_1. Let us further suppose that msg has been received with a plausibility flag set to false, which means that $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$ is a non-plausible route in $conf$. Hence, msg is dropped due to its plausibility flag being false.

Recall that, by definition, adversarial vertices cannot be neighbors. In addition, each non-adversarial vertex has a single and unique non-compromised identifier assigned to it. It follows that every route, including $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$, has a unique *meaningful* partitioning, which is the following: each non-compromised identifier, as well as each sequence of consecutive compromised identifiers should form a partition.

Let $P_1, P_2, \ldots, P_k$ be the unique meaningful partitioning of the route $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$. The fact that this route is non-plausible implies that at least one of the following two statements holds:

- *Case 1:* There exist two partitions $P_i = \{\ell_j\}$ and $P_{i+1} = \{\ell_{j+1}\}$ such that both ℓ_j and ℓ_{j+1} are non-compromised identifiers, and the corresponding non-adversarial vertices are not neighbors.
- *Case 2:* There exist three partitions $P_i = \{\ell_j\}$, $P_{i+1} = \{\ell_{j+1}, \ldots, \ell_{j+q}\}$, and $P_{i+2} = \{\ell_{j+q+1}\}$ such that ℓ_j and ℓ_{j+q+1} are non-compromised and $\ell_{j+1}, \ldots, \ell_{j+q}$ are compromised identifiers, and the non-adversarial vertices that correspond to ℓ_j and ℓ_{j+q+1}, respectively, have no common adversarial neighbor.

We show that in both cases, the adversary must have forged the digital signature of a non-adversarial machine.

In Case 1, machine ℓ_{j+1} does not sign the route reply, since it is non-adversarial and it detects that the identifier that precedes its own identifer in the route does not belong to a neighboring machine. Hence, the adversary must have forged $sig_{\ell_{j+1}}$ in msg.

In Case 2, the situation is more complicated. Let us assume that the adversary has not forged the signature of any of the non-adversarial machines. Machine ℓ_j must have received

$$msg' = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_{j+1}}))$$

from an adversarial neighbor, say A, since ℓ_{j+1} is compromised, and thus, a non-adversarial machine would not send out a route reply message with $sig_{\ell_{j+1}}$. In order to generate msg', machine A must have received

$$msg'' = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_{j+q+1}}))$$

because by assumption, the adversary has not forged the signature of ℓ_{j+q+1}, which is non-compromised. Since A has no adversarial neighbor, it could have received msg'' only from a non-adversarial machine. However, the only non-adversarial machine that would send out msg'' is ℓ_{j+q+1}. This would mean that A is a common adversarial neighbor of ℓ_j and ℓ_{j+q+1}, which contradicts the assumption of Case 2. This means that our original assumption cannot be true, and hence, the adversary must have forged the signature of a non-adversarial machine.

It should be intuitively clear that if the signature scheme is secure, then the adversary can forge a signature only with negligible probability, and thus, a route reply message in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ is dropped due to its plausibility flag set to false only with negligible probability. Nevertheless, we sketch how this could be proven formally. The proof is indirect. We assume that there exist a configuration $conf$ and an adversary $\mathcal{A}$ such that a route reply message in $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ is dropped due to its plausibility flag set to false with probability ϵ, and then, based on that, we construct a forger F that can break the signature scheme with probability ϵ/n. If ϵ is non-negligible, then so is ϵ/n, and thus, the existence of F contradicts with the assumption about the security of the signature scheme.

The construction of F is the following. Let puk be an arbitrary public key of the signature scheme. Let us assume that the corresponding private key prk is not known to F, but F has access to a signing oracle that produces signatures on submitted messages using prk. F runs a simulation of $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ where all machines are initialized as described in the model, except that the public key of a randomly selected non-adversarial machine ℓ_i is replaced with puk. During the simulation, whenever ℓ_i signs a message m, F submits m to the oracle, and replaces the signature of ℓ_i on m with the one produced by the oracle. This signature verifies correctly on other machines later, since the public verification key of ℓ_i is replaced with puk. By assumption, with probability ϵ, the simulation of $Sys^{\mathsf{ideal}}_{conf,\mathcal{A}}$ will result in a route reply message msg such that all signatures in msg are correct and msg contains a non-plausible route. As we saw above, this means that there exists a non-adversarial machine ℓ_j such that msg contains the signature sig_{ℓ_j} of ℓ_j, but ℓ_j has never signed (the corresponding part of) msg. Let us assume that $i = j$. In this case, sig_{ℓ_j} is a signature that verifies correctly with the public key puk. Since ℓ_j did not sign (the corresponding part of) msg, F did not call the oracle to generate sig_{ℓ_j}. This means that F managed to produce a signature on a message that verifies correctly with puk. Since F selected ℓ_i randomly, the probability of $i = j$ is $\frac{1}{n}$, and hence, the success probability of F is ϵ/n. □

Besides being provably secure, endairA has another significant advantage over Ariadne (and similar protocols): it is more efficient, because it requires less cryptographic computation overall from the nodes. This is because in endairA, only the processing of the route reply messages

involves cryptographic operations, and a route reply message is processed only by those nodes that are in the node list carried in the route reply. In contrast to this, in Ariadne, the route request messages need to be digitally signed by all intermediate nodes; however, due to the way a route request is propagated, this means that each node in the network must sign each and every route request.

1.6 Summary

Attacks against ad hoc routing protocols can be subtle and difficult to discover by informal reasoning about the properties of the protocol. We demonstrated this by presenting novel attacks on exisiting rotuing protocols. We also show that it is possible to adopt rigorous techniques developed for the security analysis of cryptographic algorithms and protocols, and apply them in the context of ad hoc routing protocols in order to gain more assurances about their security. We demonstrated this by proposing a simulation based framework for on-demand source routing protocols that allows us to give a precise definition of routing security, to model the operation of a given routing protocol in the presence of an adversary, and to prove (or fail to prove) that the protocol is secure. We also proposed a new on-demand source routing protocol, endairA, and we demonstrated the usage of the proposed framework by proving that it is secure in our model. Originally, we developed endairA for purely illustrative purposes, however, it has some noteworthy features that may inspire designers of future protocols. We focused on on-demand source routing protocols, but similar principles can be applied to other types of protocols too.

2 Cooperative packet forwading in wireless ad hoc networks

THESIS GROUP 2. *I propose a model based on game theory to investigate equilibrium conditions of packet forwarding strategies in static ad hoc networks. I prove theorems about the equilibrium conditions for both cooperative and non-cooperative strategies. I perform simulations to estimate the probability that the conditions for a cooperative equilibrium hold in randomly generated network scenarios. By means of these simulations, I show that in static ad hoc networks cooperation does not emerge by itself, but it needs to be encouraged. This result formally justifies the value of a huge body of research on mechanisms that aim at stimulating cooperation among the nodes of ad hoc networks. [C6, J3]*

In multi-hop wireless ad hoc networks, networking services are provided by the nodes themselves. As a fundamental example, the nodes must make a mutual contribution to packet forwarding in order to ensure an operable network. If the network is under the control of a single authority, as is the case for military networks and rescue operations, the nodes cooperate for the critical purpose of the network. However, if each node is its own authority, cooperation between the nodes cannot be taken for granted; on the contrary, it is reasonable to assume that each node has the goal to maximize its own benefits by enjoying network services and at the same time minimizing its contribution. This selfish behavior can significantly damage network performance [10, 30].

Researchers have identified the problem of stimulating cooperation in ad hoc networks and proposed several solutions to give nodes incentive to contribute to common network services. These solutions are based on a reputation system [8, 31] or on a virtual currency [11, 40]. All of these solutions are heuristics to provide a reliable cooperation enforcement scheme, assuming that there is indeed a need for such mechnisms to stimulate cooperation. Other researchers, on the other hand, have claimed that under specific conditions, cooperation may emerge without incentive techniques [38, 39]. However, they have assumed a random connection setup, thus abstracting away the topology of the network.

We aim at determining under which conditions cooperation without incentives can exist, while taking the network topology into account. Indeed, in reality, the interactions between nodes are not random, as they are determined by the network topology and the communication pattern in the network. We focus on the most basic networking mechanism, namely packet forwarding. We define a model in a game theoretic framework and identify the conditions under which an equilibrium based on cooperation exists. As the problem is involved, we deliberately restrict ourselves to a static configuration.

2.1 Game theoretic model of packet forwarding

THESIS 2.1. *I define a model and a meta-model that allow for the study of strategic interactions between the nodes in an ad hoc network. The model is based on game theory, and it essentially consists in the definition of a forwarding game played by the source and the forwarders of a data flow. The meta-model is based on automata theory, and it is used to study the properties of the forwarding game. I introduce the important notions of dependecy graph and dependency loop. [C6, J3]*

System model

Connectivity graph: Let us consider an ad hoc network of n nodes. Let us denote the set of all nodes by N. Each node has a given power range and two nodes are said to be neighbors if they reside within the power range of each other. We represent the neighbor relationship

between the nodes with an undirected graph, which we call the *connectivity graph*. Each vertex of the connectivity graph corresponds to a node in the network, and two vertices are connected with an edge if the corresponding nodes are neighbors.

Routes: Communication between two non-neighboring nodes is based on multi-hop relaying. This means that packets from the source to the destination are forwarded by intermediate nodes. For a given source and destination, the intermediate nodes are those that form the shortest path[3] between the source and the destination in the connectivity graph. We call such a chain of nodes (including the source and the destination) a *route*. We call the topology of the network with a given set of communicating nodes a *scenario*.

Time: We use a discrete model of time where time is divided into slots. We assume that both the connectivity graph and the set of existing routes remain unchanged during a time slot, whereas changes may happen at the end of each time slot. We assume that the duration of the time slot is much longer than the time needed to relay a packet from the source to the destination. This means that a node is able to send several packets within one time slot. This allows us to abstract away individual packets and to represent the data traffic in the network with *flows*. We assume CBR flows, which means that a source node sends the same amount of traffic in each time slot. Note, however, that this amount may be different for every source node and every route.

Forwarding game

We model the operation of the network as a game, which we call the *forwarding game*. The players of the forwarding game are the nodes. In each time slot t, each node i chooses a cooperation level $p_i(t) \in [0, 1]$, where 0 and 1 represent full defection and full cooperation, respectively. Here, defection means that the node does not forward traffic for the benefit of other nodes, whereas cooperation means that it does. Thus, $p_i(t)$ represents the fraction of the traffic routed through i in t that i actually forwards. Note that i has a single cooperation level $p_i(t)$, which it applies to every route in which it is involved as a forwarder. We prefer to not require the nodes to be able to distinguish the flows that belong to different routes, because this would require identifying the source-destination pairs and applying a different cooperation level to each of them; this would probably increase the computation at the nodes significantly.

Let us assume that in time slot t there exists a route r with source node s and ℓ intermediate nodes $f_1, f_2, \ldots, f_\ell$. Let us denote by $T_s(r)$ the constant amount of traffic that s wants to send on r in each time slot. The throughput $\tau(r, t)$ experienced by the source s on r in t is defined as the fraction of the traffic sent by s on r in t that is delivered to the destination. Since we are studying cooperation in packet forwarding, we assume that the main reason for packet losses in the network is the non-cooperative behavior of the nodes. In other words, we assume that the network is not congested and that the number of packets dropped because of the limited capacity of the nodes and the links is negligible. Hence, $\tau(r, t)$ can be computed as the product of $T_s(r)$ and the cooperation levels of all intermediate nodes:

$$\tau(r, t) = T_s(r) \cdot \prod_{k=1}^{\ell} p_{f_k}(t) \tag{1}$$

In addition, we define the normalized throughput $\hat{\tau}(r, t)$ as follows:

$$\hat{\tau}(r, t) = \frac{\tau(r, t)}{T_s(r)} = \prod_{k=1}^{\ell} p_{f_k}(t) \tag{2}$$

[3]In other words, here, we abstract away the details of the routing protocol, and we model it as a function that returns the shortest path between the source and the destination. If there are multiple shortest paths, then one of them is selected at random.

We will use the normalized throughput later as an input of the strategy function of s.

The payoff $\xi_s(r,t)$ of s on r in t depends on the experienced throughput $\tau(r,t)$. In general, $\xi_s(r,t) = u_s(\tau(r,t))$, where the utility u_s is some non-decreasing function. We further assume that u_s is concave, derivable at $T_s(r)$, and $u_s(0) = 0$. We place no other restrictions on u_s. Note that the utility function of different nodes may be different.

The payoff $\eta_{f_j}(r,t)$ of the j-th intermediate node f_j on r in t is non-positive and represents the cost for node f_j to forward packets on route r during time slot t. It is defined as follows:

$$\eta_{f_j}(r,t) = -T_s(r) \cdot c \cdot \hat{\tau}_j(r,t) \tag{3}$$

where c is the cost of forwarding one unit of traffic, and $\hat{\tau}_j(r,t)$ is the normalized throughput on r in t leaving node j. For simplicity, we assume that the nodes have the same, fixed transmission power, and therefore c is the same for every node in the network, and it is independent from r and t. $\hat{\tau}_j(r,t)$ is computed as the product of the cooperation levels of the intermediate nodes from f_1 up to and including f_j:

$$\hat{\tau}_j(r,t) = \prod_{k=1}^{j} p_{f_k}(t) \tag{4}$$

In our model, the payoff of the destination is 0. In other words, we assume that only the source benefits if the traffic reaches the destination (information push). However, our model can be applied in the reverse case: all our results also hold when only the destination benefits from receiving traffic. An example of this case is a file download (information pull).

The total payoff $\pi_i(t)$ of node i in time slot t is then computed as

$$\pi_i(t) = \sum_{q \in S_i(t)} \xi_i(q,t) + \sum_{r \in F_i(t)} \eta_i(r,t) \tag{5}$$

where $S_i(t)$ is the set of routes in t where i is the source, and $F_i(t)$ is the set of routes in t where i is an intermediate node.

Strategy space

In every time slot, each node i updates its cooperation level using a strategy function σ_i. In general, i could choose a cooperation level to be used in time slot t, based on the information it obtained in *all* preceding time slots. In order to make the analysis feasible, we assume that i uses only information that it obtained in the previous time slot. More specifically, we assume that i chooses its cooperation level $p_i(t)$ in time slot t based on the normalized throughput it experienced in time slot $t-1$ on the routes where it was a source:

$$p_i(t) = \sigma_i([\hat{\tau}(r,t-1)]_{r \in S_i(t-1)}) \tag{6}$$

where $[\hat{\tau}(r,t-1)]_{r \in S_i(t-1)}$ represents the normalized throughput vector for node i in time slot $t-1$, each element of which is the normalized throughput experienced by i on a route where it was source in $t-1$. The strategy of a node i is then defined by its strategy function σ_i and its initial cooperation level $p_i(0)$.

Note that σ_i takes as input the normalized throughput and not the total payoff received by i in the previous time slot. The rationale is that i should react to the behavior of the rest of the network, which is represented by the normalized throughput in our model.

There is an infinite number of possible strategies; here we highlight only a few of them for illustrative purposes. In these examples, we assume that the input of the strategy function is a scalar (i.e., a vector of length 1) denoted by *in* below.

- *Always Defect (AllD):* A node playing this strategy defects in the first time slot, and then uses the strategy function $\sigma_i(in) = 0$.
- *Always Cooperate (AllC):* A node playing this strategy starts with cooperation, and then uses the strategy function $\sigma_i(in) = 1$.
- *Tit-For-Tat (TFT):* A node playing this strategy starts with cooperation, and then mimics the behavior of its opponent in the previous time slot. The strategy function that corresponds to the TFT strategy is $\sigma_i(in) = in$.
- *Suspicious Tit-For-Tat (S-TFT):* A node playing this strategy defects in the first time slot, and then applies the strategy function $\sigma_i(in) = in$.
- *Anti Tit-For-Tat (Anti-TFT):* A node playing this strategy does exactly the opposite of what its opponent does. In other words, after cooperating in the first time slot, it applies the strategy function $\sigma_i(in) = 1 - in$.

If the output of the strategy function is independent of its input, then the strategy is called a *non-reactive strategy* (e.g., AllD or AllC). If the output depends on the input, then the strategy is *reactive* (e.g., TFT or Anti-TFT).

Our model requires that each source be able to observe the throughput in a given time slot on each of its routes. We assume that this is made possible with high enough precision by using some higher level control protocol above the network layer.

2.2 Meta-model

We introduce a meta-model in order to formalize the properties of the packet forwarding game. In the meta-model, we focus on the evolution of the cooperation levels of the nodes; all other details of the model defined earlier (e.g., amounts of traffic, forwarding costs, and utilities) are abstracted away. Unlike in the model, in the meta-model, we will assume that routes remain unchanged during the lifetime of the network. In addition, we assume for the moment that each node is the source of only one route (we will relax this assumption later). We emphasize that all of our analytical results hold in the extended case as well.

Dependency graph

Let us consider a route r. The payoff received by the source on r depends on the cooperation levels of the intermediate nodes on r. We represent this dependency relationship between the nodes with a directed graph, which we call the *dependency graph*. Each vertex of the dependency graph corresponds to a network node. There is a directed edge from vertex i to vertex j, denoted by the ordered pair (i, j), if there exists a route where i is an intermediate node and j is the source. Intuitively, an edge (i, j) means that the behavior (cooperation level) of i has an effect on j. The concept of dependency graph is illustrated in Figure 5.

Game automaton

Now we define the automaton Θ that will model the unfolding of the forwarding game in the meta-model. The automaton is built on the dependency graph. We assign a machine M_i to every vertex i of the dependency graph and interpret the edges of the dependency graph as links that connect the machines assigned to the vertices. Each machine M_i thus has some input and some (possibly 0) output links.

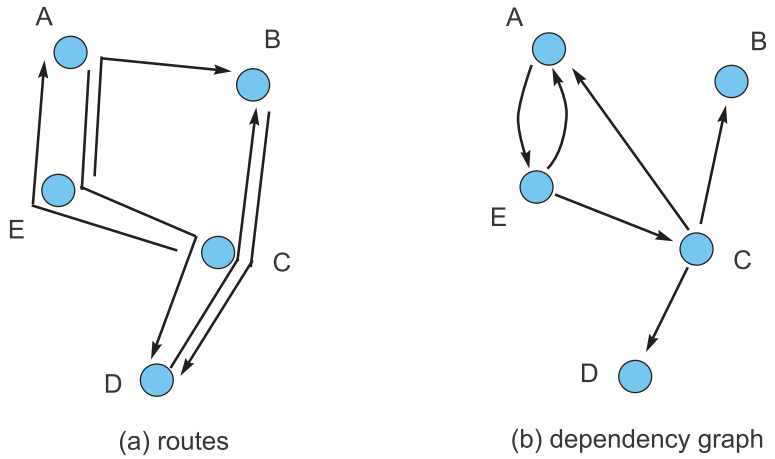


Figure 5: Representation of a network: (a) a graph showing 5 routes and (b) the corresponding dependency graph.

The internal structure of the machine is illustrated in Figure 6. Each machine M_i consists of a multiplication[4] gate $\prod$ followed by a gate that implements the strategy function σ_i of node i. The multiplication gate $\prod$ takes the values on the input links and passes their product to the strategy function gate[5]. Finally, the output of the strategy function gate is passed to each output link of M_i.

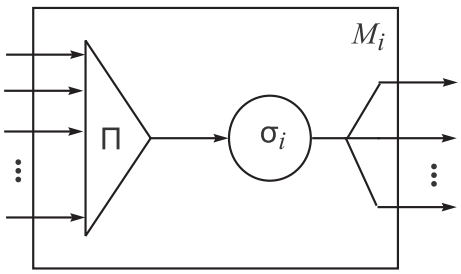


Figure 6: Internal structure of machine M_i.

The automaton Θ works in discrete steps. Initially, in step 0, each machine M_i outputs some initial value $x_i(0)$. Then, in step $t > 0$, each machine computes its output $x_i(t)$ by taking the values that appear on its input links in step $t - 1$.

Note that if $x_i(0) = p_i(0)$ for all i, then in step t, each machine M_i will output the cooperation level of node i in time slot t (i.e., $x_i(t) = p_i(t)$), as we assumed that the set of routes (and hence the dependency graph) remains unchanged in every time slot. Therefore, the evolution of the values (which, in fact, represent the state of the automaton) on the output links of the machines models the evolution of the cooperation levels of the nodes in the network.

In order to study the interaction of node i with the rest of the network, we extract the gate that implements the strategy function σ_i from the automaton Θ. What remains is the automaton without σ_i, which we denote by Θ_{-i}. Θ_{-i} has an input and an output link; if we connect these

[4]The multiplication comes from the fact that the experienced normalized throughput for the source (which is the input of the strategy function of the source) is the product of the cooperation levels of the forwarders on its route.

[5]Note that here σ_i takes a single real number as input, instead of a vector of real numbers as we defined earlier, because we assume that each node is source of only one route.

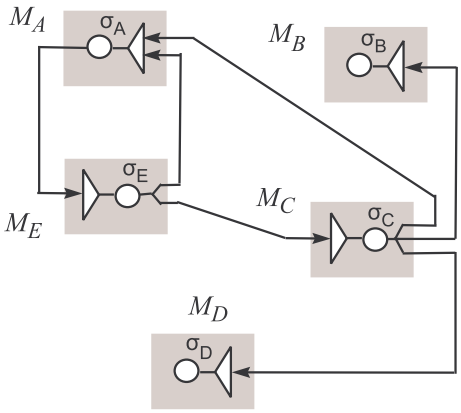


Figure 7: The automaton that corresponds to the dependency graph of Figure 5.

to the output and the input, respectively, of σ_i (as illustrated in Figure 8), then we get back the original automaton Θ. In other words, the automaton in Figure 8 is another representation of the automaton in Figure 7, which captures the fact that from the viewpoint of node i, the rest of the network behaves like an automaton: The input of Θ_{-i} is the sequence $\overline{x}_i = x_i(0), x_i(1), \ldots$ of the cooperation levels of i, and its output is the sequence $\overline{y}_i = y_i(0), y_i(1), \ldots$ of the normalized throughput values for i.

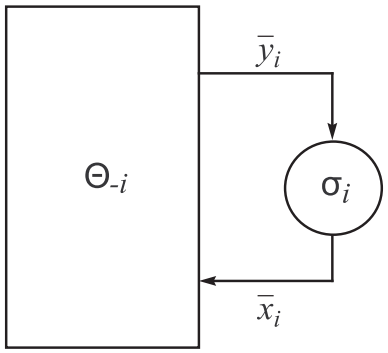


Figure 8: Model of interaction between node i and the rest of the network represented by the automaton Θ_{-i}.

By using the system of equations that describe the operation of Θ, one can easily express any element $y_i(t)$ of sequence $\overline{y}_i$ as some function of the preceding elements $x_i(t-1), x_i(t-2), \ldots, x_i(0)$ of sequence $\overline{x}_i$ and the initial values $x_j(0)$ $(j \neq i)$ of the machines within Θ_{-i}. We call such an expression of $y_i(t)$ the t-th *input/output formula* or the t-th *i/o formula* of Θ_{-i}, for short. It is important to note that the i/o formulae of Θ_{-i} may involve any strategy function σ_j where $j \neq i$, but they never involve σ_i. Considering again the automaton in Figure 7, and extracting, for instance, σ_A, we can determine the first few i/o formulae of Θ_{-A} as follows:

$$\begin{aligned}
y_A(0) &= x_C(0) \cdot x_E(0) \\
y_A(1) &= \sigma_C(x_E(0)) \cdot \sigma_E(x_A(0)) \\
y_A(2) &= \sigma_C(\sigma_E(x_A(0))) \cdot \sigma_E(x_A(1)) \\
y_A(3) &= \sigma_C(\sigma_E(x_A(1))) \cdot \sigma_E(x_A(2)) \\
\ldots &\quad \ldots
\end{aligned}$$

Dependency loops

A *dependency loop* L of node i is a sequence $(i, v_1), (v_1, v_2), \ldots, (v_{\ell-1}, v_\ell), (v_\ell, i)$ of edges in the dependency graph. The length of a dependency loop L is defined as the number of edges in L, and it is denoted by $|L|$. The existence of dependency loops is important: if node i has no dependency loops, then the cooperation level chosen by i in a given time slot has no effect on the normalized throughput experienced by i in future time slots. In the example, nodes B and D have no dependency loops.

Every node i has two types of dependency loops; these types depend on the strategies played by the other nodes in the loop. If L is a dependency loop of i, and all other nodes $j \neq i$ in L play reactive strategies, then L is said to be a *reactive dependency loop* of i. If, on the contrary, there exists at least one node $j \neq i$ in L that plays a non-reactive strategy, then L is called a *non-reactive dependency loop* of i.

2.3 Analytical results

THESIS 2.2. *I determine the conditions for the existence of cooperative and non-cooperative equilibria in the forwarding game. In particular, I prove the following [J3]:*

- *If a node is a forwarder on some route, but it has no dependency loop, then its best strategy is to defect, i.e., to deny packet forwarding. (Theorem 2.1)*
- *If a node is a forwarder on some route, and it has only non-reactive dependency loops, then its best strategy is to defect, i.e., to deny packet forwarding. (Theorem 2.2)*
- *If every node j ($j \neq i$) defects, then node i cannot have any reactive dependency loop, and hence its best response is to defect. Consequently, every node defecting is a Nash equilibrium of the forwarding game. (Corollary 2.1)*
- *Assuming that node i is a forwarder on at least one route, its best strategy is to cooperate, i.e., to forward packets if (a) node i has a dependency loop with all of the sources for which it forwards packets; and (b) all these dependency loops are reactive; and (c) the maximum forwarding cost for node i on every route where it is a forwarder is smaller than its possible future benefit averaged over all those routes. If all three conditions are satisfied, then node i has an incentive to cooperate, since otherwise its defective behavior will negatively affect its own future payoff. (Theorem 2.3)*
- *If conditions (a) and (c) described above hold for all nodes which act as a forwarder on some route, then all nodes playing the Tit-for-Tat reactive strategy is a Nash equilibrium in the forwarding game. (Corollary 2.2)*

Our goal is to find *possible* Nash equilibria of packet forwarding strategies. In the next section, we will investigate the *probability of fulfillment* of the conditions for possible Nash equilibria in randomly generated scenarios. The existence of a Nash equilibrium based on cooperation would mean that there are cases in which cooperation is "naturally" encouraged, i.e. without using incentive mechanisms. In the following, we use the model and the meta-model that we introduced earlier.

The goal of the nodes is to maximize the payoff that they accumulate over time. However, the end of the game is unpredictable. Thus, we apply the standard technique used in the theory of iterative games [4]. We model the *finite* forwarding game with an unpredictable end as an *infinite* game where future payoffs are *discounted*. The cumulative payoff $\overline{\pi}_i$ of a node i is

computed as the weighted sum of the payoffs $\pi_i(t)$ that i obtains in each time slot t:

$$\overline{\pi}_i = \sum_{t=0}^{\infty} [\pi_i(t) \cdot \omega^t] \tag{7}$$

where $0 < \omega < 1$, and hence, the weights exponentially decrease with t. The *discounting factor* ω represents the degree to which the payoff of each time slot is discounted relative to the previous time slot.

Recall that $S_i(t)$ denotes the set of routes for which i is the source, and that $F_i(t)$ denotes the set of routes for which i is an intermediate node. As we assume that the routes remain static, meaning that $S_i(t)$ and $F_i(t)$ do not change over time, we will simply write S_i and F_i instead of $S_i(t)$ and $F_i(t)$. In addition, since we assume that each node is a source on exactly one route, S_i is a singleton. We denote the single route in S_i by r_i, and the amount of traffic sent by i on r_i in every time slot by T_i. The cardinality of F_i will be denoted by $|F_i|$. For any route $r \in F_i$, we denote the set of intermediate nodes on r upstream from node i (including node i) by $\Phi(r, i)$. Moreover, $\Phi(r)$ denotes the set of all forwarder nodes on route r, and $src(r)$ denotes the source of route r. Finally, the set of nodes that are forwarders on at least one route is denoted by Φ (i.e., $\Phi = \{i \in N : F_i \neq \emptyset\}$).

Theorem 2.1. *If a node i is in Φ, and it has no dependency loops, then its best strategy is AllD (i.e., to choose cooperation level 0 in every time slot).*

Proof. Node i wants to maximize its cumulative payoff $\overline{\pi}_i$ defined in (7). In our case, $\pi_i(t)$ can be written as:

$$\begin{aligned} \pi_i(t) &= \xi_i(r_i, t) + \sum_{r \in F_i} \eta_i(r, t) \\ &= u_i(T_i \cdot y_i(t)) - \sum_{r \in F_i} T_{src(r)} \cdot c \cdot \prod_{k \in \Phi(r,i)} x_k(t) \end{aligned}$$

Given that i has no dependency loops, $y_i(t)$ is independent of all the previous cooperation levels $x_i(t')$ $(t' < t)$ of node i. Thus, $\overline{\pi}_i$ is maximized if $x_i(t') = 0$ for all $t' \geq 0$. □

Theorem 2.2. *If a node i is in Φ, and it has only non-reactive dependency loops, then its best strategy is AllD.*

Proof. The proof is similar to the proof of Theorem 2.1. Since all dependency loops of i are non-reactive, its experienced normalized throughput $\overline{y}_i$ is independent of its own behavior $\overline{x}_i$. This implies that its best strategy is full defection. □

From this theorem, we can easily derive the following corollary.

Corollary 2.1. *If every node j $(j \neq i)$ plays AllD, then the best response of i to this is AllD. Hence, every node playing AllD is a Nash equilibrium.*

If the conditions of Theorems 2.1 and 2.2 do not hold, then we cannot determine the best strategy of a node i in general, because it very much depends on the particular scenario (dependency graph) in question and the strategies played by the other nodes.

Now, we will show that, under certain conditions, cooperative equilibria do exist in the network. In order to do so, we first prove the following lemma:

Lemma 2.1. *Let us assume that node i is in Φ, and let us consider a route $r \in F_i$. In addition, let us assume that there exists a dependency loop L of i that contains the edge $(i, src(r))$. If all nodes in L (other than i) play the TFT strategy, then the following holds:*

$$y_i(t+\delta) \leq \prod_{k \in \Phi(r,i)} x_k(t) \tag{8}$$

where $\delta = |L| - 1$.

Proof. Let L be the following sequence of edges in the dependency graph: $(v_0, v_1), (v_1, v_2), \ldots, (v_\delta, v_{\delta+1})$, where $v_{\delta+1} = v_0 = i$ and $v_1 = src(r)$. We know that each node is the source of a single route; let us denote by r_{v_j} $(0 < j \leq \delta + 1)$ the route, on which v_j is the source. It follows that $r_{v_1} = r$. In addition, we know that the existence of edge (v_j, v_{j+1}) $(0 \leq j \leq \delta)$ in the dependency graph means that v_j is a forwarder on $r_{v_{j+1}}$. The following holds for every node v_j $(0 \leq j \leq \delta)$:

$$x_{v_j}(t) \geq \prod_{k \in \Phi(r_{v_{j+1}}, v_j)} x_k(t) \geq \prod_{k \in \Phi(r_{v_{j+1}})} x_k(t) = y_{v_{j+1}}(t) \tag{9}$$

Furthermore, since every node except for $v_0 = v_{\delta+1} = i$ plays TFT, we have the following for every $0 < j \leq \delta$:

$$x_{v_j}(t+1) = y_{v_j}(t) \tag{10}$$

Using (9) and (10) in an alternating order, we get the following:

$$x_{v_0}(t) \geq \prod_{k \in \Phi(r_{v_1}, v_0)} x_k(t) \geq y_{v_1}(t) = x_{v_1}(t+1) \geq y_{v_2}(t+1) = x_{v_2}(t+2) \geq \ldots \geq y_{v_{\delta+1}}(t+\delta) \tag{11}$$

By substituting i for v_0 and $v_{\delta+1}$, and r for r_{v_1}, we get the statement of the lemma:

$$x_i(t) \geq \prod_{k \in \Phi(r,i)} x_k(t) \geq \ldots \geq y_i(t+\delta) \tag{12}$$

□

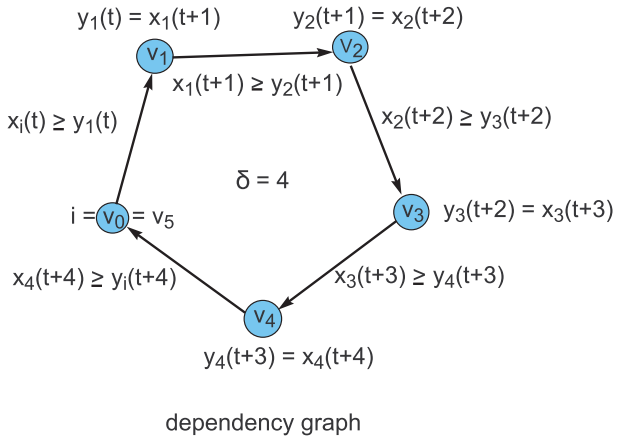


Figure 9: Example to illustrate the propagation of behavior as expressed formally in Lemma 2.1.

As an example, let us consider Figure 9, which illustrates a dependency loop of length 5 (i.e., $\delta = 4$). According to Lemma 2.1, if nodes v_1, v_2, v_3, and v_4 play TFT, then the normalized

throughput enjoyed by node i in time slot $t+4$ is upper bounded by its own cooperation level in time slot t. Intuitively, this means that if node i does not cooperate, then this defection "propagates back" to it on the dependency loop. The delay of this effect is given by the length of the dependency loop.

Theorem 2.3. *Assuming that node i is in Φ, the best strategy for i is full cooperation in each time slot, if the following set of conditions holds:*

1. *for every $r \in F_i$, there exists a dependency loop $L_{i,src(r)}$ that contains the edge $(i, src(r))$;*
2. *for every $r \in F_i$,*
$$\frac{u_i'(T_i) \cdot T_i \cdot \omega^{\delta_{i,src(r)}}}{|F_i|} > T_{src(r)} \cdot c \tag{13}$$
where $u_i'(T_i)$ is the value of the derivative[6] of $u_i(\tau)$ at $\tau = T_i$, and $\delta_{i,src(r)} = |L_{i,src(r)}| - 1$; and
3. *every node in Φ (other than i) plays the TFT strategy.*

Proof. In this proof we will express the maximum possible value of the total payoff for node i in general. Then we will show that the maximum corresponds to the case in which node i fully cooperates. First, we introduce the linear function $f(\tau) = u_i'(T_i) \cdot \tau + u_i(T_i) - u_i'(T_i) \cdot T_i$. Function f is the tangent of function u_i at $\tau = T_i$. Note that due to the fact that u_i is non-decreasing and concave, we have that $f(\tau) \geq u_i(\tau)$ for all τ; in addition, we have equality at $\tau = T_i$ (i.e., $f(T_i) = u_i(T_i)$).

By definition, the total payoff $\overline{\pi}_i$ of node i is the following:

$$\begin{aligned} \overline{\pi}_i &= \sum_{t=0}^{\infty} \left[\xi_i(r_i, t) + \sum_{r \in F_i} \eta_i(r, t) \right] \omega^t \\ &= \sum_{t=0}^{\infty} \left[u_i(T_i \cdot y_i(t)) - \sum_{r \in F_i} T_{src(r)} \cdot c \cdot \prod_{k \in \Phi(r,i)} x_k(t) \right] \omega^t \end{aligned} \tag{14}$$

Because of *Condition 1* and *Condition 3*, we can use Lemma 2.1 to obtain the following inequality for every $r \in F_i$:

$$\prod_{k \in \Phi(r,i)} x_k(t) \;\geq\; y_i(t + \delta_{i,src(r)}) \tag{15}$$

which leads to the following upper bound on $\overline{\pi}_i$:

$$\overline{\pi}_i \;\leq\; \sum_{t=0}^{\infty} \left[u_i(T_i \cdot y_i(t)) - \sum_{r \in F_i} T_{src(r)} \cdot c \cdot y_i(t + \delta_{i,src(r)}) \right] \omega^t \tag{16}$$

Since the first term of the right side of (16), $u_i(T_i \cdot y_i(t))$, is independent of r, the following holds:

$$u_i(T_i \cdot y_i(t)) \;=\; \sum_{r \in F_i} \frac{u_i(T_i \cdot y_i(t))}{|F_i|} \tag{17}$$

[6] Recall the assumption that u_i is derivable at T_i.

By substituting the right side of (17) into (16), we get the following:

$$\begin{aligned}\overline{\pi}_i &\leq \sum_{t=0}^{\infty}\left[\sum_{r\in F_i}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}-\sum_{r\in F_i}T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\right]\omega^t \\ &= \sum_{r\in F_i}\left[\sum_{t=0}^{\infty}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t-\sum_{t=0}^{\infty}T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\cdot\omega^t\right] \qquad (18)\end{aligned}$$

Let us consider the first term of (18). We will now split up the summation that goes from $t=0$ to ∞ into two summations such that one goes from $t=0$ to $\delta_{i,src(r)}-1$, and the other goes from $t=\delta_{i,src(r)}$ to ∞. Then, we shift the index in the second sum in such a way that the summation goes from $t=0$ to ∞ again:

$$\begin{aligned}&\sum_{t=0}^{\infty}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t \\ &= \sum_{t=0}^{\delta_{i,src(r)}-1}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t+\sum_{t=\delta_{i,src(r)}}^{\infty}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t \\ &= \sum_{t=0}^{\delta_{i,src(r)}-1}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t+\sum_{t=0}^{\infty}\frac{u_i(T_i\cdot y_i(t+\delta_{i,src(r)}))}{|F_i|}\cdot\omega^{t+\delta_{i,src(r)}} \qquad (19)\end{aligned}$$

By writing (19) back into (18), we get the following:

$$\begin{aligned}\overline{\pi}_i \leq \sum_{r\in F_i}\Bigg[&\sum_{t=0}^{\delta_{i,src(r)}-1}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t \\ &+\sum_{t=0}^{\infty}\left[\frac{u_i(T_i\cdot y_i(t+\delta_{i,src(r)}))}{|F_i|}\cdot\omega^{\delta_{i,src(r)}}-T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\right]\cdot\omega^t\Bigg] \qquad (20)\end{aligned}$$

Let us consider the first term of (20). Since the utility function u_i is non-decreasing and $y_i(t)\leq 1$, we get the following:

$$\begin{aligned}\sum_{t=0}^{\delta_{i,src(r)}-1}\frac{u_i(T_i\cdot y_i(t))}{|F_i|}\cdot\omega^t &\leq \sum_{t=0}^{\delta_{i,src(r)}-1}\frac{u_i(T_i)}{|F_i|}\cdot\omega^t \\ &= \frac{u_i(T_i)}{|F_i|}\cdot\frac{1-\omega^{\delta_{i,src(r)}}}{1-\omega} \qquad (21)\end{aligned}$$

Now let us consider the second term of (20). By using the fact that $f(\tau)\geq u_i(\tau)$ for all τ, we get the following:

$$
\begin{aligned}
&\sum_{t=0}^{\infty}\left[\frac{u_i(T_i\cdot y_i(t+\delta_{i,src(r)}))}{|F_i|}\cdot\omega^{\delta_{i,src(r)}}-T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\right]\cdot\omega^t\\
&\leq \sum_{t=0}^{\infty}\left[\frac{f(T_i\cdot y_i(t+\delta_{i,src(r)}))}{|F_i|}\cdot\omega^{\delta_{i,src(r)}}-T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\right]\cdot\omega^t\\
&= \sum_{t=0}^{\infty}\left[\frac{u_i'(T_i)\cdot T_i\cdot y_i(t+\delta_{i,src(r)})+u_i(T_i)-u_i'(T_i)\cdot T_i}{|F_i|}\cdot\omega^{\delta_{i,src(r)}}-T_{src(r)}\cdot c\cdot y_i(t+\delta_{i,src(r)})\right]\cdot\omega^t\\
&= \sum_{t=0}^{\infty}\left[\frac{u_i(T_i)-u_i'(T_i)\cdot T_i}{|F_i|}\cdot\omega^{\delta_{i,src(r)}}+\left(\frac{u_i'(T_i)\cdot T_i\cdot\omega^{\delta_{i,src(r)}}}{|F_i|}-T_{src(r)}\cdot c\right)\cdot y_i(t+\delta_{i,src(r)})\right]\cdot\omega^t\\
&= \frac{u_i(T_i)-u_i'(T_i)\cdot T_i}{|F_i|}\cdot\frac{\omega^{\delta_{i,src(r)}}}{1-\omega}+\sum_{t=0}^{\infty}\left(\frac{u_i'(T_i)\cdot T_i\cdot\omega^{\delta_{i,src(r)}}}{|F_i|}-T_{src(r)}\cdot c\right)\cdot y_i(t+\delta_{i,src(r)})\cdot\omega^t\\
&\leq \frac{u_i(T_i)-u_i'(T_i)\cdot T_i}{|F_i|}\cdot\frac{\omega^{\delta_{i,src(r)}}}{1-\omega}+\sum_{t=0}^{\infty}\left(\frac{u_i'(T_i)\cdot T_i\cdot\omega^{\delta_{i,src(r)}}}{|F_i|}-T_{src(r)}\cdot c\right)\cdot\omega^t\\
&= \frac{u_i(T_i)-u_i'(T_i)\cdot T_i}{|F_i|}\cdot\frac{\omega^{\delta_{i,src(r)}}}{1-\omega}+\left(\frac{u_i'(T_i)\cdot T_i\cdot\omega^{\delta_{i,src(r)}}}{|F_i|}-T_{src(r)}\cdot c\right)\cdot\frac{1}{1-\omega}\\
&= \frac{u_i(T_i)}{|F_i|}\cdot\frac{\omega^{\delta_{i,src(r)}}}{1-\omega}-\frac{T_{src(r)}\cdot c}{1-\omega}
\end{aligned}
$$

where in the transition from (22) to (22), we used *Condition 2* and the fact that $y_i(t+\delta_{i,src(r)})\leq 1$. By using (21) and (22) in (20), we get the following:

$$
\begin{aligned}
\overline{\pi}_i &\leq \sum_{r\in F_i}\left[\frac{u_i(T_i)}{|F_i|}\cdot\frac{1-\omega^{\delta_{i,src(r)}}}{1-\omega}+\frac{u_i(T_i)}{|F_i|}\cdot\frac{\omega^{\delta_{i,src(r)}}}{1-\omega}-\frac{T_{src(r)}\cdot c}{1-\omega}\right]\\
&= \frac{1}{1-\omega}\cdot\sum_{r\in F_i}\left[\frac{u_i(T_i)}{|F_i|}-T_{src(r)}\cdot c\right]\\
&= \frac{1}{1-\omega}\cdot\left(u_i(T_i)-c\cdot\sum_{r\in F_i}T_{src(r)}\right)
\end{aligned}
$$

Now let us consider what payoff is achieved by node i if it fully cooperates in every time slot. In this case, since all the other nodes play TFT, every node will always fully cooperate, and hence, every node will experience a normalized throughput equal to 1 in each time slot. This can easily be derived from the i/o formulae describing the behavior of the nodes, which take a simple form due to the simplicity of the strategy function of the TFT strategy. As a consequence, we have that $y_i(t)=1$ for every t, and $x_k(t)=1$ for every k and for every t. In this case expression 14 becomes:

$$
\begin{aligned}
\overline{\pi}_i &= \sum_{t=0}^{\infty}\left[u_i(T_i\cdot y_i(t))-\sum_{r\in F_i}T_{src(r)}\cdot c\cdot\prod_{k\in\Phi(r,i)}x_k(t)\right]\omega^t\\
&= \frac{1}{1-\omega}\cdot\left(u_i(T_i)-c\cdot\sum_{r\in F_i}T_{src(r)}\right)
\end{aligned}
$$

This means that by fully cooperating, the payoff of node i reaches the upper bound expressed in (22); in other words, there is no better strategy for node i than full cooperation. □

We have derived *necessary* conditions for spontaneous cooperation from Theorem 2.1 and 2.2. The fulfillment of the three conditions of Theorem 2.3 is *sufficient* for cooperation to be the best strategy for node i. We now discuss these three conditions one by one. *Condition 1* requires that node i has a dependency loop with all of the sources for which it forwards packets. *Condition 2* means that the maximum forwarding cost for node i on every route where i is a forwarder must be smaller than its possible future benefit averaged over the number of routes where i is a forwarder. Finally, *Condition 3* requires that all forwarding nodes in the network (other than node i) play TFT. This implies that all the dependency loops of node i are reactive. We note that the reactivity of the dependency loops can be based on other reactive strategies, different from TFT (for example Anti-TFT), but in that case the analysis becomes very complex. The analysis of the case when every node plays TFT is made possible by the simplicity of the strategy function $\sigma(x) = x$, which belongs to the TFT strategy. If all three conditions of Theorem 2.3 are satisfied, then node i has an incentive to cooperate, since otherwise its defective behavior will negatively affect its own payoff. However, as we will show later, *Condition 1* is a very strong requirement that is virtually never satisfied in randomly generated scenarios.

Both the AllC and TFT strategies result in full cooperation if the conditions of Theorem 2.3 hold. However, node i should not choose AllC, because AllC is a non-reactive strategy, and this might cause other nodes to change their strategies to AllD, as we will show later. Hence, we can derive the following corollary for cooperative Nash equilibria.

Corollary 2.2. *If the first two conditions of Theorem 2.3 hold for every node in Φ, then all nodes playing TFT is a Nash equilibrium.*

In the next subsection, we study *Condition 1* of Theorem 2.3, more specifically, the probability that it is satisfied for all nodes in randomly generated scenarios. Now, we briefly comment on *Condition 2*. As it can be seen, the following factors make *Condition 2* easier to satisfy:

- *Steep utility functions:* The steeper the utility function of node i is, the larger the value of its derivative is at $\tau = T_i$, which, in turn, makes the left side of (13) larger.
- *Short dependency loops:* In *Condition 2*, $\delta_{i,src(r)} + 1$ is the length of *any* dependency loop of node i that contains the edge $(i, src(r))$. Clearly, we are interested in the shortest of such loops, because the smaller $\delta_{i,src(r)}$ is, the larger the value of $\omega^{\delta_{i,src(r)}}$ is, which, in turn, makes the left side of (13) larger. It is similarly advantageous if ω is close to 1, which means, in general, that the probability that the game will continue is higher and thus possible future payoffs count more.
- *Small extent of involvement in forwarding:* The left side of (13) is increased if the cardinality of F_i is decreased. In other words, if node i is a forwarder on a smaller number of routes, then *Condition 2* is easier to satisfy for i.

2.4 Simulation results

THESIS 2.3. *I show by means of simulations that the probability of satisfying all conditions for the existence of a cooperative equilibrium is very small in practice. In particular, among 1000 randomly generated scenarios, there was not any scenario that satisfied the condition that requires that all forwarder nodes have dependecy loops with all the source nodes whose traffic they are forwarding. Hence, in practice, with high probability, there will be some nodes in the network whose best best strategy is defecting. Yet, I also show by means of simulation that the behavior of these defectors affects only a fraction of the nodes in the network; hence, local subsets of cooperating nodes are not excluded. [J3]*

We have run a set of simulations to determine the probability that the conditions of our theorems and their corollaries hold. In particular, our goal is to estimate the probability that the first condition of Theorem 2.3 holds for every node in randomly generated scenarios[7]. In addition, we also estimate the probability that the condition of Theorem 2.1 does not hold for any of the nodes in randomly generated scenarios.

In our simulations, we randomly place nodes on a toroid[8] area. Then, for each node, we randomly choose a number of destinations and we determine a route to these destinations using a shortest path algorithm. If several shortest paths existed to a given destination, then we randomly choose a single one. From the routes, we build up the dependency graph of the network. The simulation parameters are summarized in Table 1.

Table 1: Parameter values for the simulation

Parameter	*Value*
Number of nodes	100, 150, 200
Distribution of the nodes	random uniform
Area type	Torus
Area size	1500x1500m, 1850x1850m, 2150x2150m
Radio range	200 m
Number of destinations per node	1-10
Route selection	shortest path

Note that we increase the network size and the simulation area in parallel in order to keep the node density at a constant level. All the presented results are the mean values of 1000 simulation runs.

In the first set of simulations, we investigate the probability that the first condition of Theorem 2.3 holds for every node. Among the 1000 scenarios that we generated randomly, we observed that there was not a single scenario in which the first condition of Theorem 2.3 was satisfied for all nodes. Thus, we conclude that the probability of a Nash equilibrium based on TFT as defined in Corollary 2.2 is very small.

In the second set of simulations, we investigate the proportion of random scenarios, where cooperation of all nodes is not excluded by Theorem 2.1. Figure 10 shows the proportion of scenarios, where each node in Φ has at least one dependency loop as a function of the number of routes originating at each node. We can observe that for an increasing number of routes originating at each node, the proportion of scenarios, where each node has at least one dependency loop, increases as well. Intuitively, as more routes are introduced in the network, more edges are added to the dependency graph. Hence, the probability that a dependency loop exists for each node increases. Furthermore, we can observe that the proportion of scenarios in which each node has at least one dependency loop decreases, as the network size increases. This is due to the following reason: the probability that there exists at least one node for which the condition of Theorem 2.1 holds increases as the number of nodes increases.

Figure 10 shows that the proportion of scenarios, where cooperation of all nodes is not excluded by Theorem 2.1 becomes significant only for cases in which each node is a source of a large number of routes. This implies that the necessary condition expressed by Theorem 2.1 is

[7]The second condition of Theorem 2.3 is a numerical one. Whether it is fulfilled or not very much depends on the actual utility functions and parameter values (e.g., amount of traffic and discounting factor) used. Since, by appropriately setting these parameters, the second condition of Theorem 2.3 can always be satisfied, in our analysis, we make the optimistic assumption that this condition holds for every node in Φ.

[8]We use this area type to avoid border effects. In a realistic scenario, the toroid area can be considered as an inner part of a large network.

a strong requirement for cooperation in realistic settings (i.e., for a reasonably low number of routes per node).

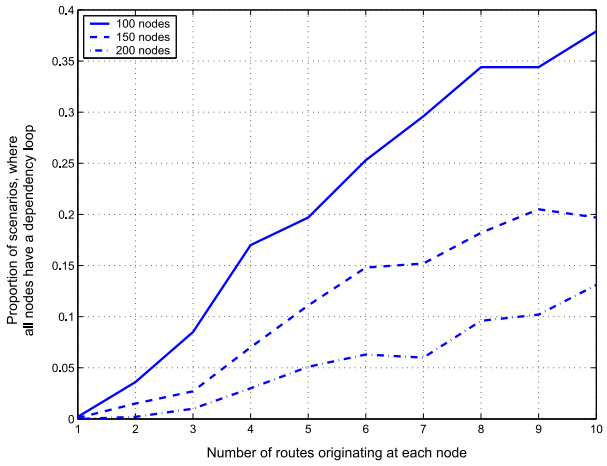


Figure 10: Proportion of scenarios, where each node that is a forwarder has at least one dependency loop.

Now let us consider the case, in which the nodes for which Theorem 2.1 holds begin to play AllD. This non-cooperative behavior can lead to an "avalanche effect" if the nodes iteratively optimize their strategies: nodes that defect can cause the defection of other nodes. We examine this avalanche effect in a simulation setting as follows.

Let us assume that each node is a source on one route. First, we identify the nodes in the set of forwarders Φ that have AllD as the best strategy due to Theorem 2.1. We denote the set of these defectors by Z_0. Then, we search for sources that are dependent on the nodes in Z_0. We denote the set of these sources by Z_0^+. Since the normalized throughput of the nodes in Z_0^+ is less than or equal to the cooperation level of any of their forwarders (including the nodes in Z_0), their best strategy becomes AllD, as well, due to Theorem 2.2. Therefore, we extend the set Z_0 of defectors, and obtain $Z_1 = Z_0 \cup Z_0^+$. We extend the set Z_k of defectors iteratively in this way until no new sources are affected (i.e., $Z_k \cup Z_k^+ = Z_k$). The remaining set $\Phi \setminus Z_k$ of nodes is not affected by the behavior of the nodes in Z_k (and hence the nodes in Z_0); this means that they are potential cooperators. Similarly, we can investigate the avalanche effect when the nodes are sources of several routes. In that case, we take the pessimistic assumption that the defection of a forwarder causes the defection of its sources. Then, we can iterate the search for the nodes that are affected by defection in the same way as above.

In Figure 11, we present the proportion of scenarios, where there exists a subset of nodes that are not affected by the defective behavior of the initial AllD players. We can see that this proportion converges rapidly to 1 as the number of routes originating at each node increases. The intuitive explanation is that increasing the number of routes per source (i.e., adding edges to the dependency graph) decreases the probability that Theorem 2.1 holds for a given node. Thus, as the number of routes per sources increases the number of forwarders that begin to play AllD decreases and so does the number of nodes affected by the avalanche effect.

Additionally, we present in Figure 12 the proportion of forwarder nodes that are not affected by the avalanche effect. The results show that if we increase the number of routes originating at each node, the average number of unaffected nodes increases rapidly. For a higher number of routes per node, this increase slows down, but we can observe that the majority of the nodes are not affected by the defective behavior of the initial AllD players.

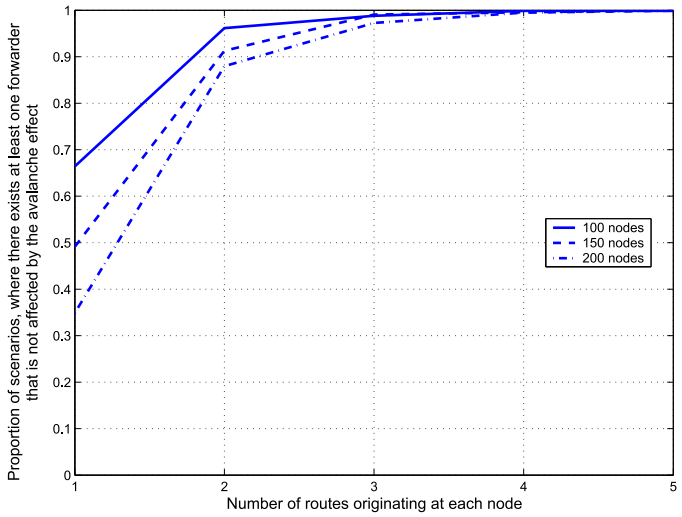


Figure 11: Proportion of scenarios, where at least one node is not affected by the defective behavior of the initial nodes.

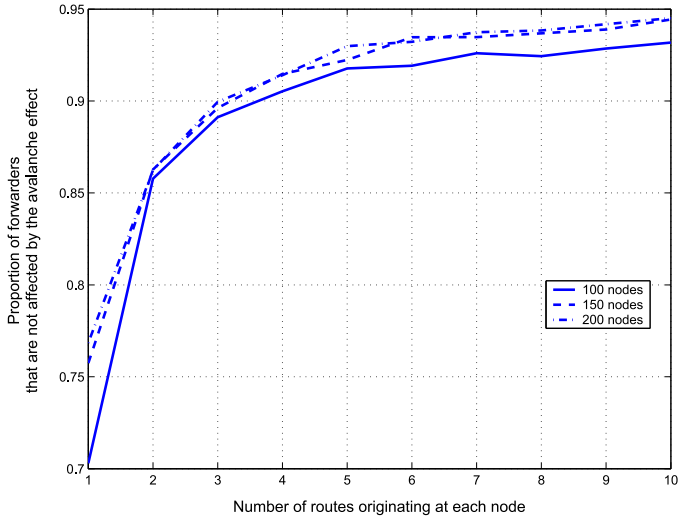


Figure 12: Average proportion of forwarder nodes that are not affected by the avalanche effect.

2.5 Summary

We have presented a game theoretic model to investigate the conditions for cooperation in wireless ad hoc networks, in the absence of incentive mechanisms. Because of the complexity of the problem, we have restricted ourselves to a static network scenario. We have then derived conditions for cooperation from the topology of the network and the existing communication routes. We have introduced the concept of dependency graph, based on which we have been able to prove several theorems. As one of the results, we have proven that cooperation solely based on the self-interest of the nodes can *in theory* exist. However, our simulation results show that *in practice*, the conditions of such cooperation are virtually never satisfied. We conclude that with a very high probability, there will be some nodes that have AllD as their best strategy and therefore, these nodes need an incentive to cooperate. We have also shown that the behavior of these defectors affects only a fraction of the nodes in the network; hence, local subsets of cooperating nodes are not excluded.

3 Wormhole detection in wireless sensor networks

THESIS GROUP 3. *I propose three new mechanisms for detecting wormhole attacks in ad hoc and sensor networks. Two of these mechanisms are based on statistical hypothesis testing and they produce probabilistic results. For these mechanisms, I use simulations to study their detection performance. The third proposed mechanism is based on the principles of distance bounding. I analyze the properties of this mechanism, in particular its resistance to attacks aiming at shortening estimated distances by means of informal reasoning. [C2, C5]*

Wireless sensor networks (WSNs) consist of a large number of sensors that monitor the environment, and a few base stations that collect the sensor readings. The sensors are usually battery powered and limited in computing and communication resources, while the base stations are considered to be more powerful. In order to reduce the overall energy consumption of the sensors, it is conceived that the sensors send their readings to the base station via multiple wireless hops. Hence, in a wireless sensor network, the sensor nodes are responsible not only for the monitoring of the environment, but also for forwarding data packets towards the base station on behalf of other sensors.

In order to implement the above described operating principle, the sensors need to be aware of their neighbors, and they must also be able to find routes to the base station. An adversary may take advantage of this, and may try to control the routes and to monitor the data packets that are sent along these routes. One way to achieve this is to set up a *wormhole* in the network. A wormhole is an out-of-band connection, controlled by the adversary, between two physical locations in the network. The two physical locations representing the two ends of the wormhole can be at any distance from each other; however, the typical case is that this distance is large. The out-of-band connection between the two ends can be a wired connection or it can be based on a long-range, directional wireless link. The adversary installs radio transceivers at both ends of the wormhole. Then, she transfers packets (possibly selectively) received from the network at one end of the wormhole to the other end via the out-of-band connection, and there, re-injects the packets into the network.

Wormholes affect route discovery mechanisms that operate on the connectivity graph. For instance, many routing protocols search for the shortest paths in the connectivity graph. With a well placed wormhole, the adversary can achieve that many of these shortest paths go through the wormhole. This gives a considerable power to the adversary, who can monitor a large fraction of the network traffic, or mount a denial-of-service attack by permanently or selectively dropping data packets passing through the wormhole so that they never reach their destinations. Therefore, in most of the applications, wormhole detection is an important requirement.

The wormhole attack is also dangerous in other types of wireless applications where direct, one-hop communication and physical proximity play an important role. An example is a wireless access control system for buildings, where each door is equipped with a contactless smart card reader, and they are opened only if a valid contactless smart card is presented to the reader. The security of such a system depends on the assumption that the personnel carefully guard their cards. Thus, if a valid card is present, then the system can safely infer that a legitimate person is present as well, and the door can be opened. Such a system can be defeated if an adversary can set up a wormhole between a card reader and a valid card that could be far away, in the pocket of a legitimate user: The adversary can relay the authentication exchange through the wormhole and gain unauthorized access. The feasibility of this kind of attack has been demonstrated in [27].

Wormhole detection mechanisms fall into two classes: the centralized mechanisms and the decentralized ones. In the centralized approach, data collected from the local neighborhood of every node are sent to a central entity (e.g., the base station in case of sensor networks). The

central entity uses the received data to construct a model of the entire network, and tries to detect inconsistencies in this model that are potential indicators of wormholes. In the decentralized approach, each node constructs a model of its own neighborhood using locally collected data; hence no central entity is needed. However, decentralized wormhole detection mechanisms often require special assumptions, such as tightly synchronized clocks, knowledge of geographical location, or existence of special hardware, e.g., directional antennas.

In this thesis group, we propose three mechanisms for wormhole detection in wireless sensor networks. Two of these are centralized mechanisms and the third one is a decentralized mechanism. Both proposed centralized mechanisms are based on hypothesis testing and they provide probabilistic results. The first mechanism, called the Neighbor Number Test (NNT), detects the increase in the number of the neighbors of the sensors, which is due to the new links created by the wormhole in the network. The second mechanism, called the All Distances Test (ADT), detects the decrease of the lengths of the shortest paths between all pairs of sensors, which is due to the shortcut links created by the wormhole in the network. Both mechanisms assume that the sensors send their neighbor list to the base station, and it is the base station that runs the algorithms on the network graph that is reconstructed from the received neighborhood information. The decentralized detection mechanism that we propose is based on an authenticated distance bounding protocol.

3.1 System and adversary models

We assume that the system consists of a large number of sensor nodes and a few base stations placed on a two dimensional surface. We assume that the base stations have no resource limitations, and they can run complex algorithms. We assume that the sensors have a fixed radio range r, and two sensors are neighbors, if they reside in the radio range of each other. We assume that the sensors run some neighbor discovery protocol, and they can determine who their neighbors are. We also assume that the sensors send their neighborhood information to the closest base station regularly in a secure way. By security we mean confidentiality, integrity, and authenticity; in other words, we assume that the adversary cannot observe and change the neighborhood information sent to the base stations by the sensors, neither can it spoof sensors and fabricate false neighborhood updates. This can be ensured by using cryptographic techniques. Note that the neighborhood information can be piggy-backed on regular data packets. In addition, as sensor networks tend to be rather static, sending only the changes in the neighborhood since the last update would reduce the overhead significantly. The base stations can pool the received neighborhood information together, and based on that, they can reconstruct the graph of the sensor network. We assume that the node density is high enough so that the network is always connected.

We assume that the adversary can set up a wormhole in the system. The wormhole is a dedicated connection between two physical locations. There are radio transceivers installed at both ends of the wormhole, and packets that are received at one end can be sent to and retransmitted at the other end. In this way, the adversary can achieve that nodes that otherwise do not reside in each other's radio range can still hear each other and establish a neighbor relationship (i.e., they can run the neighbor discovery protocol). This means that the adversary can introduce new, otherwise non-existing links in the network graph that is constructed by the base stations based on the received neighborhood information.

The wormhole is characterized by the distance between the two locations that it connects and the radio ranges of its transceivers. We assume that the receiving and the sending ranges of both transceivers are the same, and we will call this range the *radius* of the wormhole. The radius of the wormhole is not necessarily equal to the radio range of the sensors.

In principle, the adversary can drop packets carrying neighborhood information that are sent

to the base stations via the wormhole. However, consistently missing neighborhood updates can be detected by the base stations and they indicate that the system is under attack. Therefore, we assume that the adversary does not drop the neighborhood updates. In addition, by the assumptions made earlier, it cannot alter or fabricate them either.

THESIS 3.1. *I propose two centralized wormhole detection mechanisms for wireless sensor networks based on a statistical hypothesis testing approach. Both mechanisms require the nodes to send their neighbor list to a central base station, which reconstructs the network topology graph and identifies inconsistences caused by wormholes. The first mechanism (Neighbor number test) identifies distortions in the node degree distribution in the network, while the second mechanism (All distances test) identifies distortions in the distribution of the length of the shortest paths in the network. Both mechanisms use the χ^2–test as hypothesis testing method, and I describe how the parameters should be determined for it. Furthermore, I show by means of simulations, that the Neighbor number test effectively detects wormholes if the wormhole's radio range is comparable to the nodes' radio range, but its detection accuracy is not acceptable, when the wormhole's radio range is significantly smaller than the nodes' radio range. Moreover, I show that the All distances test performs better than the Neighbor number test in general, and it can detect wormholes with small radio ranges, although detection accuracy depends very much on the node density in the network. [C2]*

3.2 Neighbor number test (NNT)

Our first detection mechanism is based on the fact that by introducing new links into the network graph, the adversary increases the number of neighbors of the nodes within its radius. If the distribution of the placement of the nodes is given, then it is possible to compute the hypothetical distribution of the number of neighbors. Then, the base stations can use statistical tests to decide if the network graph constructed from the neighborhood information that is received from the sensors corresponds to this hypothetical distribution. In order to illustrate this idea, let us consider the example depicted in Figure 13, where the dark bars correspond to the hypothetical distribution of the number of neighbors, and the light bars show the actual distribution in the network graph reconstructed from the sensors' neighborhood updates. One can see that the probability of higher neighbor numbers (15-20) is increased with respect to the hypothetical distribution, and the idea of the proposed mechanism is to detect this increase by using statistical tests.

Based on the above observations, the NNT algorithm is given as follows:

1. The base station computes the expected histogram of the neighbor numbers using the hypothetical distribution of the number of neighbors.
2. The base station collects the neighborhood updates from the sensors, constructs the network graph, and computes the histogram of the real neighbor numbers in the graph.
3. The base station compares the two histograms with the χ^2–test.
4. If the computed χ^2 number is larger than a preset threshold that corresponds to a given significance level, then a wormhole is indicated.

Computing the parameters for the χ^2–test

Assuming that the sensors are placed uniformly at random on the plane, the probability of two nodes being neighbors is

$$q = \frac{r^2 \cdot \pi}{T}$$

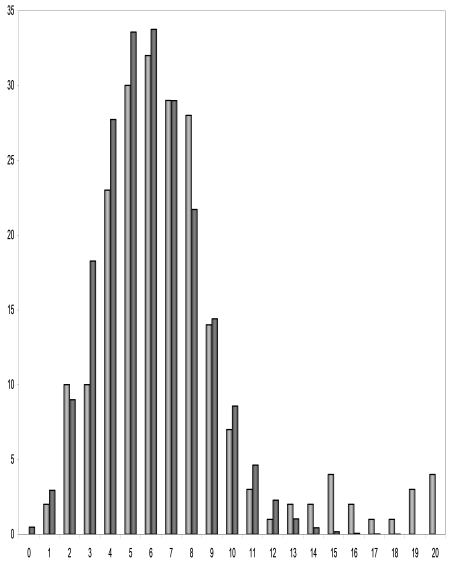


Figure 13: Hypothetical (dark) and real (light) distributions of the number of neighbors

where r is the radio range of the sensor nodes and T is the size of the area where the sensor network is deployed. The probability $p(k)$ of having exactly k neighbors is

$$p(k) = \binom{N}{k} \cdot q^k \cdot (1-q)^{N-k}$$

where $N+1$ is the total number of nodes in the network. Let us partition the set $\{0, 1, 2, \ldots\}$ into subsets $B_1, B_2, \cdots, B_m$, such that $e(i) = (N+1)\sum_{k \in B_i} p(k)$ be larger than 5 (a requirement needed by the χ^2–test [7]). The χ^2 number is then computed using the following formula:

$$\chi^2 = \sum_{\forall i} \frac{r(i) - e(i)}{e(i)}$$

where $r(i)$ is the real number of nodes with number of neighbors in B_i. If χ^2 is below the threshold that corresponds to a given significance level (this threshold can be looked up in published tables of χ^2 values), then the hypothesis is accepted, and no wormhole is indicated. Otherwise the hypothesis is rejected, and a wormhole is indicated.

3.3 All distances test (ADT)

Our second detection mechanism is based on the fact that the wormhole shortens the paths in the network, or more precisely, it distorts the distribution of the length of the shortest paths between all pairs of nodes. This is illustrated by the example depicted in Figure 14, where the dark bars represent the hypothetical distribution of the length of the shortest paths and the light bars represent the real distribution. As it can be seen, the two distributions are different, and in the real distribution, shorter paths are more likely than in the hypothetical one. The idea is to detect this difference with statistical tests.

The ADT algorithm is very similar to the NNT algorithm:

1. The base station computes the histogram of the length of the shortest paths between all pairs of nodes in the hypothetical case when there is no wormhole in the system using the knowledge of the distribution of the node placement.

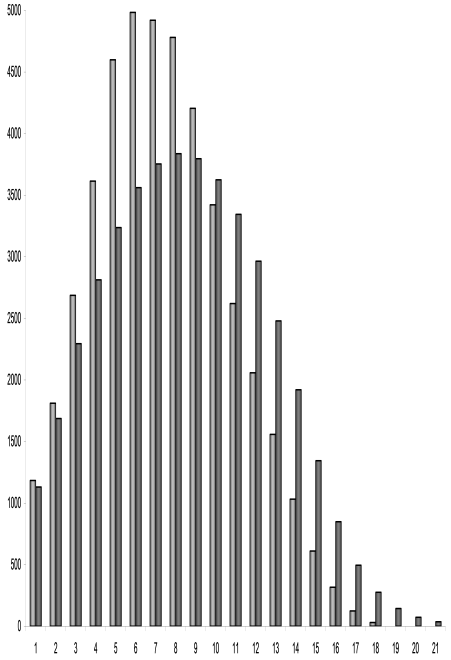


Figure 14: Hypothetical (dark) and real (light) distributions of the length of the shortest paths between all pairs of nodes

2. The base station collects the neighborhood information from the sensors, and computes the histogram of the length of the shortest paths in the real network.
3. The base station compares the two histograms with the χ^2–test.
4. If the computed χ^2 number is larger than a preset threshold that corresponds to a given significance level, then a wormhole is indicated.

Computing the parameters for the χ^2–test

In this case, we were not able to derive a close formula that describes the hypothetical distribution of the length of the shortest paths. Instead, we propose to estimate that distribution by randomly placing nodes on the plane according to the distribution of the node placement, and compute the lengths of the shortest paths between all pairs of nodes in the resulting graph. We propose to repeat the experience many times and average the normalized histograms obtained in these experiences. Once the hypothetical distribution is estimated in this way, the χ^2–test can be used in a similar way as we described above.

3.4 Simulation results

In order to evaluate the effectiveness of the proposed centralized mechanisms, we built a simulator that places 300 sensor nodes uniformly at random on a 500 m $\times$ 500 m flat area with one base station in the middle, and it also places a wormhole randomly in the same area. The simulator permits us to set three parameters: the radio range of the sensors, the radius of the wormhole, and the distance between the affected areas at the two ends of the wormhole.

We chose two extreme values for the radio range of the sensor nodes: 40 m and 70 m. The expected neighbor number is 5.9 in the 40 m case, and 18.5 in the 70 m case. Then, we split up the range between 5.9 and 18.5 evenly into 5 intervals to get the six radio range values that we used in our simulations (see Table 2).

Number of nodes	300
Extent of territory	500 m × 500 m
Number of simulation runs	100
Radio range of sensor nodes	40 m, 47 m, 54 m, 60 m, 65 m, 70 m
Radio range of the wormhole	16 m, 50 m
Distance between the affected areas at the two end wormhole	20 m, 50 m, 100 m, 200 m, 300 m, 400 m

Table 2: Simulation parameters

We set the radius of the wormhole to 16 m or to 50 m (see Table 2). These two values have been selected in such a way that the number of nodes affected by the wormhole differs significantly in the two cases. When the radius of the wormhole is 16 m, one node is affected (falls in the wormhole's range) on both ends of the wormhole on average, whereas when the radius of the wormhole is 50 m, 9.4 nodes are affected on both ends on average.

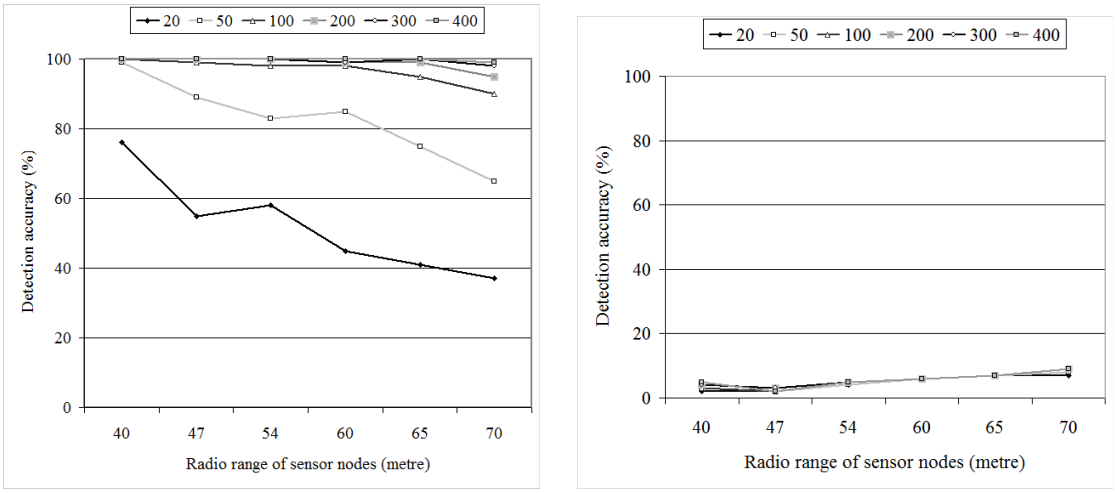
Finally, we varied the distance between the affected areas at the two ends of the wormhole between 20 m and 400 m (see Table 2).

A given combination of the possible parameter values define a test case. For each test case we run 100 simulations and averaged the results. For each radio range setting, we first determined the rate of the false positive alarms (i.e., the percentage of the simulation runs where the algorithms indicate a wormhole when there is no wormhole in the system). Then, we placed wormholes with different parameters in the system and determined the accuracy of both of our centralized wormhole detection mechanisms (i.e., the percentage of simulation runs where the wormhole is detected when there is indeed a wormhole in the system). The results are presented below.

Results of the neighbor number test (NNT)

The results of the NNT algorithm are shown on Figures 15 and 16. Figure 15(a) shows the accuracy of the detection as a function of the radio range of the sensors when the radius of the wormhole is 50 m. As it can be seen, the detection accuracy decreases as the sensors' radio range increases. The reason is that in the case of larger radio ranges, the sensors have more real neighbors, and therefore, the increase in the number of neighbors caused by the wormhole becomes less significant, and consequently, more difficult to detect. We can also observe that the detection accuracy is better when the areas affected by the wormhole are more distant from each other, although increasing this distance above 100 m has no real influence on the results. In fact, if the distance between the affected areas is smaller than the radio range of the sensors, then it is possible that two affected nodes that do not belong to the same affected areas are already real neighbors, and therefore, the wormhole does not create a new link between them. In other words, the larger the distance between the affected areas is, the higher the probability is that the wormhole introduces new links into the graph, and by doing so it increases the number of neighbors of the affected nodes.

Figure 15(b) shows the accuracy of the detection as a function of the radio range of the sensors when the radius of the wormhole is 16 m. It is clear from the figure that the NNT algorithm does not work in this case, as the accuracy of the detection is unacceptably low. The



(a) (b)

Figure 15: Detection accuracy plotted against the radio range of the sensor nodes. The different curves belong to different distances between the areas affected by the wormhole with a radius of 50 m (a) and 16 m (b)

huge difference between the performance in the 50 m case and that in the 16 m case can be explained with the large difference in the number of the affected nodes in the two cases. As we described earlier, when the radius of the wormhole is 16 m, on average one node is affected at both ends on the wormhole. Hence, practically, such a wormhole creates a single new link in the graph, which is extremely difficult detect with statistical techniques. On the other hand, as the average number of affected nodes is around 10 at both ends of the wormhole when the radius is 50 m, the number of new links introduced in the graph is around 100. More importantly, around 20 nodes out of the total of 300 have around 10 more neighbors due to the wormhole, and this can be detected by the NNT algorithm.

Figure 16 shows the percentage of the false positive alarms as a function of the radio range of the sensors. As it can be seen, the NNT algorithm performs quite well regarding the false positive alarms. Indeed, the percentage of the false positive alarms is determined by the selected significance level of the χ^2–test, which in our case was 0.025.

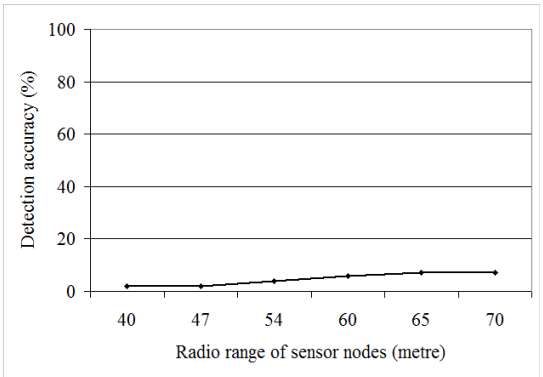


Figure 16: Percentage of false positive wormhole detections plotted against the radio range of sensor nodes

In summary, the NNT algorithm detects the wormhole reasonably well if the radius of the wormhole is comparable to or larger than the radio range of the sensors, but it performs very

badly if the radius of the wormhole is small. We note, however, that a smaller wormhole radius has smaller effect on the system in terms of the number of sensors that send measurement data to the base station through the wormhole. In order to illustrate this, we constructed the minimum spanning tree rooted at the base station, and counted the number of shortest paths between the base station and the sensors that contain a link created by the wormhole. The result is shown in Figure 17. As it can be seen, when the radius of the wormhole is 16 m, the number of concerned paths is between 0 and 50, whereas in the case of a 50 m radius, the number of concerned paths is between 100 and 200. Thus, the adversary can monitor the measurements of more sensors when the radius of the wormhole is larger, but in that case, it can also be detected more accurately by the NNT algorithm.

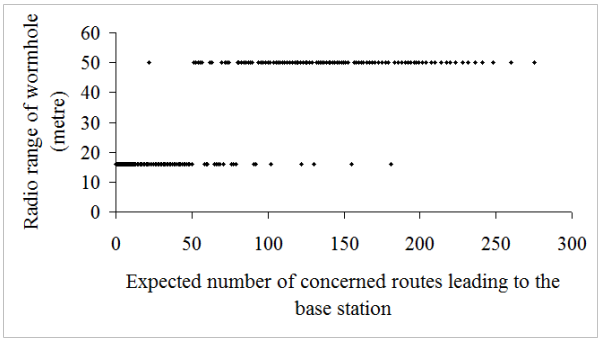


Figure 17: The effect of the wormhole on the number of the controlled shortest paths plotted against the radius of the wormhole

Results of the all distances test (ADT)

The results of the ADT algorithm are shown on Figures 18 and 19. Figure 18(a) shows the accuracy of the detection as a function of the sensors' radio range when the radius of the wormhole is 50 m, whereas Figure 18(b) shows the same when the radius of the wormhole is 16 m. Similar to the NNT algorithm, the ADT algorithm performs better when the radius of the wormhole is larger. However, unlike the NNT algorithm, the ADT algorithm is not completely unusable in the case when the radius of the wormhole is 16 m. Rather, its performance depends on the distance between the areas affected by the wormhole: the higher this distance is, the more accurate the detection is. Moreover, when the distance between the affected areas is 400 m, the accuracy is close to 100% . The explanation for this is quite obvious: a longer wormhole reduces the length of the shortest paths between more distant nodes, and thus overall, it represents a larger decrease in the average length of the shortest paths between all pairs of nodes.

Regarding the percentage of the false positive alarms (Figure 19), the ADT algorithm performs quite well except for small radio ranges.

3.5 Mutual Authenticated Distance-bounding

THESIS 3.2. *I propose a decentralized wormhole detection mechanism that uses a new distance bounding protocol that features mutual authentication of the protocol participants and a commitment phase that prevents attacks aiming at shortening the estimated distance between the participants. I show by informal reasoning that the protocol is suitable for the purpose of wormhole detection, as wormholes typically make distances appear shorter than they really are. [C5]*

The main idea of *distance-bounding* is simple but very powerful. It is based on the facts that

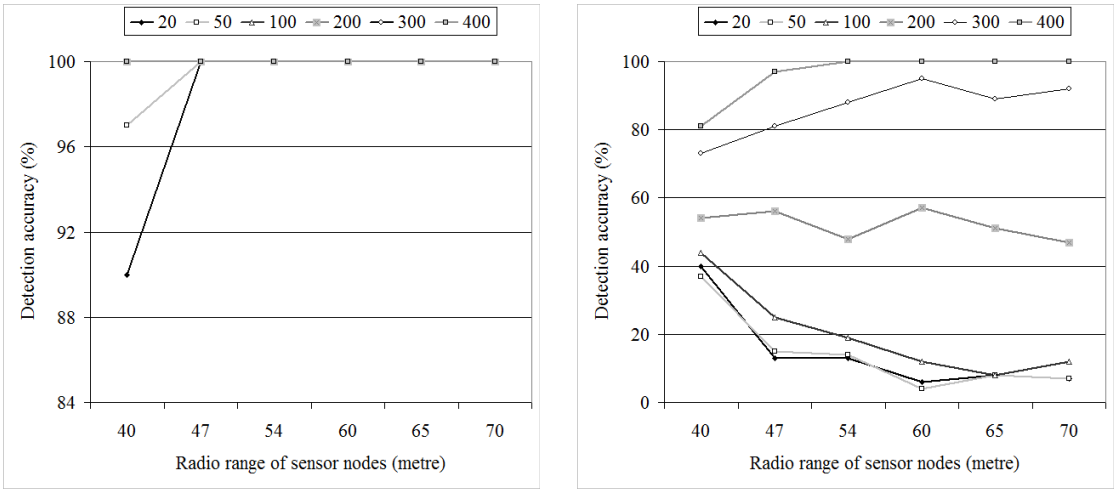


Figure 18: Detection accuracy plotted against the radio range of the sensor nodes. The different curves belong to different distances between the areas affected by the wormhole with a radius of 50 m (a) and 16 m (b)

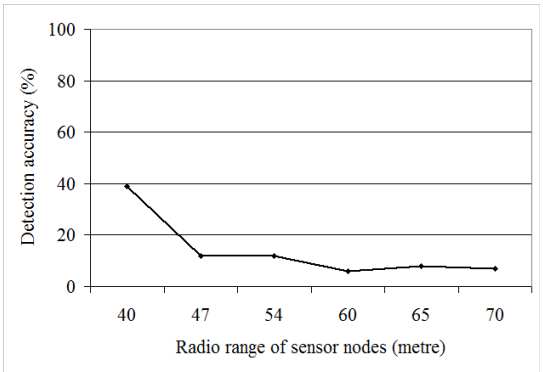


Figure 19: Percentage of false positive wormhole detections plotted against the radio range of the sensor nodes

electro-magnetic waves propagate nearly with the speed of light and with current technology it is easy to measure local timings with nanosecond precision. The distance bounding technique essentially consists of a series of rapid bit exchanges between the two nodes. Each bit sent by the first node is considered to be a challenge for which the other node is required to send a one bit response immediately. By locally measuring the time between sending out the challenges and receiving the responses, the first node can estimate its real physical distance to the other node, assuming that the messages travel with the speed of light and the processing delay at the other node is negligible.

Note that the estimated distance is only an upper bound on the real distance between the nodes, because the second node could be closer, but it can delay the responses in order to appear to be further. Even if the nodes are trusted for not delaying their responses, an active adversary can delay the messages between the parties, and hence, the estimated distance will still be just an upper bound on the real distance. However, in the case of a wormhole attack, the adversary's goal is not to make the two nodes believe that they are far away from each other. On the contrary, the adversary wants the two nodes to believe that they are within each

other's range, when in reality they are not. In order to achieve that the estimated distance is smaller than the nodes' real distance, the adversary should arrange that the messages travel faster than the speed of light, which is impossible. Thus, distance-bounding can be used for wormhole detection.

We slightly modify the above described distance-bounding technique such that it allows both nodes to measure the distance between them simultaneously and it uses symmetric key cryptographic primitives for authentication purposes. In order for this to work, it is assumed that each pair of nodes share a symmetric key. We call the resulting protocol Mutual Authenticated Distance-bounding, or shortly MAD.

Let x and y denote the two nodes in the protocol, and let their shared key be k_{xy}. We will denote the message authentication function controlled by the key k_{xy} by $mac_{k_{xy}}$. The operation of the protocol is summarized in Figure 20, and it is explained as follows:

- **Initialization phase:**
  Both x and y generate uniformly at random two numbers. The numbers of x are denoted by r and r', and the numbers of y are denoted by s and s'. Numbers r and s are ℓ bits long, and r' and s' are ℓ' bits long (i.e., $r, s \in \{0,1\}^{\ell}$ and $r', s' \in \{0,1\}^{\ell'}$) Both x and y compute a commitment to the generated numbers by using a collision resistant one-way hash function H: $c_x = H(r||r')$ and $c_y = H(s||s')$. Finally, x sends c_x to y and y sends c_y to x. Note that the random numbers can be generated and the commitments can be computed well before running the protocol.

- **Distance-bounding phase:**
  Let the bits of r and s be denoted by r_i and s_i ($i = 1, 2, \ldots, \ell$), respectively. The following two steps are repeated ℓ times, for $i = 1, 2, \ldots, \ell$:

  - x sends bit α_i to y immediately after it received β_{i-1} from y (except for α_1 which is sent without receiving any bit from y), where $\alpha_1 = r_1$ and $\alpha_i = r_i \oplus \beta_{i-1}$ for $i > 1$;
  - y sends bit $\beta_i = s_i \oplus \alpha_i$ to x immediately after it received α_i from x.

  x measures the times between sending α_i and receiving β_i, and y measures the times between sending β_i and receiving α_{i+1}. From the measured times, they both estimate their distance.

- **Authentication phase:**
  Node x computes the bits $s_i = \alpha_i \oplus \beta_i$, and the MAC

  $$\mu_x = mac_{k_{xy}}(x||y||r_1||s_1|| \ldots ||r_\ell||s_\ell)$$

  Similarly, y computes the bits $r_1 = \alpha_1$ and $r_i = \alpha_i \oplus \beta_{i-1}$ for $i > 1$, and the MAC

  $$\mu_y = mac_{k_{xy}}(y||x||s_1||r_1|| \ldots ||s_\ell||r_\ell)$$

  Finally, x sends $r'||\mu_x$ to y and y sends $s'||\mu_y$ to x. Node x verifies that the commitment c_y and the MAC μ_y of y are correct, and y verifies that the commitment c_x and the MAC μ_x of x are correct.

In the above protocol, the MAC ensures the authenticity of the exchange: both x and y can believe that they ran the distance-bounding phase with the other, and thus, the distance that they estimate is really the distance between x and y. Committing to r and s in the initialization phase ensures that the protocol is successful only if exactly the bits of r and s are exchanged. As r and s are random, an adversary cannot try to cheat x by predicting the bits of s and

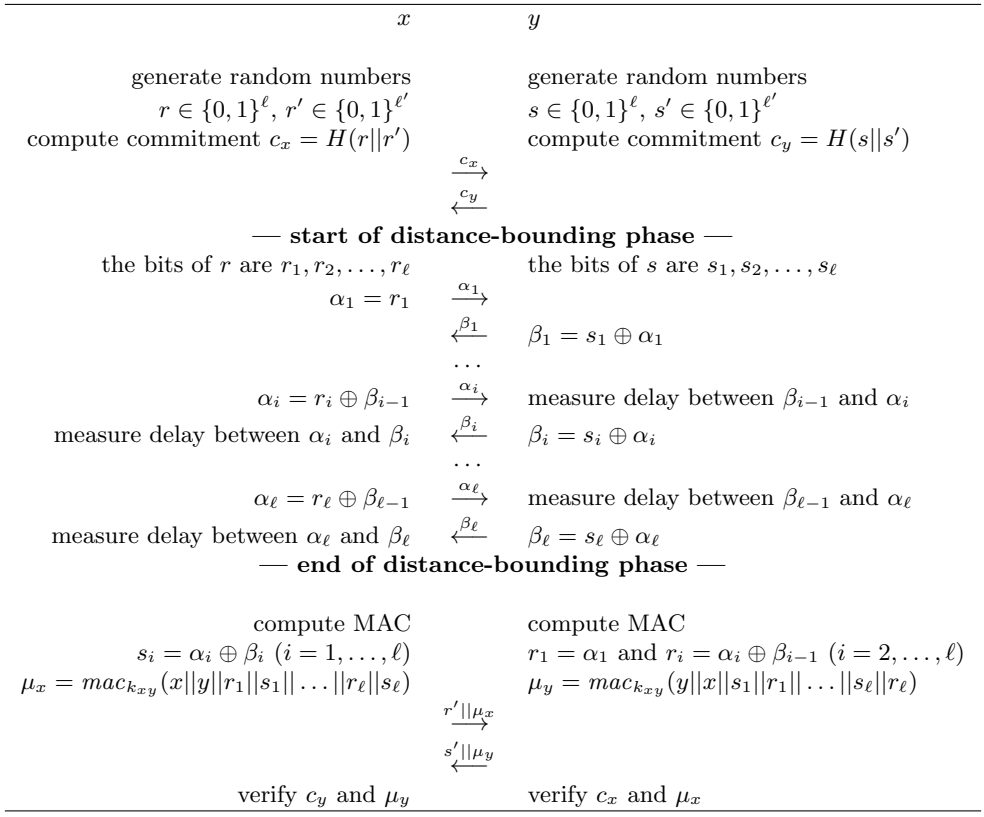


Figure 20: The Mutual Authenticated Distance-bounding (MAD) protocol.

responding earlier than y, and similarly it cannot cheat y either. More precisely, the probability that such an attack succeeds is $2^{-\ell}$ and hence decreases exponentially in ℓ.

The advantage of MAD is that it does not require the localization of the nodes or the synchronization of their clocks. MAD still requires, however, special hardware in the nodes in order to quickly switch the radio from receive mode into send mode. In addition, it needs a special medium access control protocol that allows for the transmission of bits without any delay.

3.6 Summary

In this thesis group, we have studied the problem of wormhole detection in wireless sensor networks. We proposed two centralized wormhole detection mechanisms that are based on hypothesis testing, and that provide probabilistic results. The first mechanism, called the Neighbor Number Test (NNT), detects the increase in the number of the neighbors of the sensors, which is due to the new links created by the wormhole in the network. The second mechanism, called the All Distances Test (ADT), detects the decrease of the lengths of the shortest paths between all pairs of sensors, which is due to the shortcut links created by the wormhole in the network. Both mechanisms assume that the sensors send their neighbor list to the base station, and it is the base station that runs the algorithms on the network graph constructed from the received neighborhood information.

We investigated the effectiveness of the two proposed mechanisms by means of simulation. Our results show that both mechanisms can detect the wormhole with high accuracy when the radius of the wormhole is comparable to the radio range of the sensors. In addition, the ADT algorithm performs better than the NNT algorithm when the radius of the wormhole is small

(compared to the radio range of the sensors). In terms of false alarms, both algorithms perform reasonably well.

We also proposed a decentralized wormhole detection mechanism that combines the idea of distance-bounding and mutual authentication of nodes. Distance-bounding allows the nodes that run the protocol to estimate the real physical distance between them, therefore, the approach can be used for detecting wormholes. In addition, the advantage of the proposed approach is that, unklike other decentralized approaches, it does not require the localization of the nodes or the synchronization of their clocks.

4 Securing coding based distributed storage in wireless sensor networks

THESIS GROUP 4. *I propose a new algorithm to detect pollution attacks in coding based distributed storage schemes, and two new algorithms to recover from such attacks. I measure the performance of the proposed algorithms in terms of success rate and communication and computing overhead. My results show that the first recovery algorithm can be used in practice in small to medium sized networks (10-60 source nodes and 100-600 storage nodes), while the second recovery algorithm scales up to larger systems (100 source nodes and 1000 storage nodes). [C1, J2]*

In many wireless sensor network (WSN) applications, there are multiple, distributed sources that generate data that must be stored efficiently in multiple storage nodes, each having constrained communication, computation, and storage capabilities. Using the principles of network coding [3, 20, 21, 29] and storing encoded data instead of raw data, one can increase the efficiency of the system. Suppose we have k source nodes and n storage nodes. Instead of storing raw data packets, each storage node stores a linear combination of a subset of the k data packets. Random coding techniques (distributed erasure codes, fountain codes) introduced in [16, 17, 18, 19] ensure that, for appropriately selected parameters, a collector node can reconstruct all the k data packets with high probability by downloading the encoded packets from *any* k storage nodes and solving a system of linear equations (s.l.e.). Thus, the collector node can retrieve the required data from k nearby nodes, which results in decreased energy consumption, and hence, longer network lifetime. Note that these are primary design criteria in WSNs.

While coding may increase the efficiency of distributed storage systems in a benign environment, it has a potential problem in hostile environments, where an adversary may attack the storage nodes. In particular, the problem that we are interested in in this thesis group is the so called *pollution attack*, whereby the adversary modifies some of the stored encoded data, which results in erroneous decoding of a large part of the original data upon retrieval. Note that these coding schemes mix (typically, linearly combine) blocks of the original data, therefore, a single corrupted encoded block can affect the decoding of multiple data blocks. This amplification effect of the pollution attack is particularly annoying and undesirable.

An approach to prevent the pollution attack is to require the source nodes to digitally sign [28] (or hash [22]) the data blocks before they are injected in the system. However, the digital signature scheme must have some homomorphic properties that allows for the combination of signed data blocks. Unfortunately, homomorphic signature schemes are computationally expensive, and they need a public key infrastructure (PKI) for the management of the signature verification keys. These problems hinder their usage in practical applications; in particular, due to the large computational complexity they cannot be used in sensor networks.

Our main contribution is a novel non-cryptographic approach to counteract pollution attacks in coding based distributed storage systems in WSNs. Compared to other approaches in the same vein, we do not add redundancy to the data packets, but rather, we take advantage of the inherent redundancy provided by the coding scheme itself that is designed for the distributed storage system. To the best of our knowledge, our proposal is the first error detection/correction method that does not require any new functionality at the source nodes or at the storage nodes.

Our proposal is more practical than the approach based on homomorphic digital signatures. First of all, we need neither a PKI, nor any cryptographic key management scheme, as we do not use cryptography at all. The practical value of this feature should not be underestimated. Second, while our approach also requires intensive computational effort, this is required only for the entity that retrieves information from the distributed storage system. In wireless sensor networks, where the computational overhead really matters, this entity is typically the base sta-

tion, which is usually assumed to be powerful enough. In contrast to this, in the approach based on homomorphic digital signatures, the source nodes and the storage nodes need to perform intensive computation, and those are typically resource constrained sensor nodes.

In order to measure the performance of our algorithms, we calculate the probability of success together with the complexity of the algorithms. Two complexity measures are considered: the computational complexity, measured in the number of s.l.e.'s that need to be solved, and the communication complexity, measured in the number of encoded packets that need to be downloaded when data is retrieved from the distributed storage system. We propose an attack detection algorithm that has optimal communication and computational complexity in the given system model. We also propose a recovery algorithm with very low computational complexity, and another recovery algorithm with optimal communication complexity, which has also feasible computational complexity for small to medium size practical systems.

4.1 System and adversary models

System model

The general model of the distributed storage systems that we consider in this work is taken from [17] and it is illustrated in Figure 21. The system consists of k source nodes, n storage nodes, and one or more collector nodes. Note that these are roles, and therefore, the sets of source nodes, storage nodes, and collector nodes may overlap. Only the collector node is assumed to be a powerful computer (base station), while source and storage nodes may be low capacity devices.

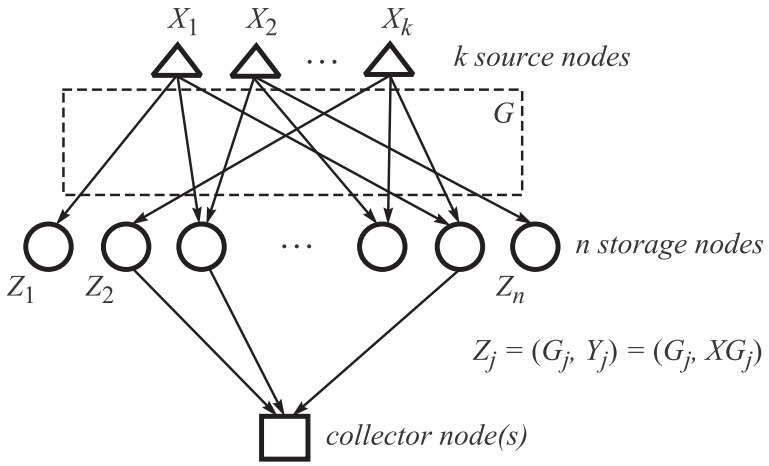


Figure 21: System model

Each source node i generates a data block X_i, and transfers it to some randomly selected subset of the storage nodes. Each storage node j computes a random linear combination of all the data blocks that it receives; the result is a single code block Y_j. Formally, we can write that $Y_j = XG_j$, where $X = (X_1, X_2, \ldots, X_k)$ is the row vector of all the data blocks, and $G_j = (g_{1j}, g_{2j}, \ldots, g_{kj})^\mathsf{T}$ is a column vector, the non-zero elements of which are the random coefficients used in the linear combination. Here, $g_{ij} \in GF(q)$ for all $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n$, and for some q. Each storage node j stores the pair $Z_j = (G_j, Y_j)$, which represents the equation $Y_j = XG_j$. The entire system is represented by the system of linear equations (s.l.e.) $Y = XG$, where $Y = (Y_1, Y_2, \ldots, Y_n)$ is the row vector of all code blocks, and $G = (G_1, G_2, \ldots, G_n)$ is

a $k \times n$ matrix that contains the coefficient vectors in its columns. Matrix G is also called generator matrix.

For appropriately selected values of k and q, any $k \times k$ submatrix of G is non-singular with high probability. According to [17], the probability of non-singularity is at least $(1 - \frac{k}{q})c_2(k)$, where $c_2(k) \to 1$, if $k \to \infty$. Larger values of q increase the probability of successful decoding, but makes the overhead of storage higher. [17] also shows that storage nodes required to store $O(\ln k)$ coefficients. E.g. if $k = 100$ and $q = 2^{20}$, the probability of singularity is $\approx 10^{-4}$, while the average overhead of a storage node is 92 bits. Therefore, the collector node can reconstruct all the data blocks with high probability by downloading the equations from any k storage nodes and solving the obtained s.l.e. for X. In the rest of this presentation, we assume that this property of G holds.

In fact, each data block X_i can itself be a column vector of m symbols $(x_{1i}, x_{2i}, \ldots, x_{mi})^\mathsf{T}$, where $x_{\ell i} \in GF(q)$ for all $i = 1, 2, \ldots, k$ and $\ell = 1, 2, \ldots, m$. In that case, each code block Y_j is also a column vector $(y_{1j}, y_{2j}, \ldots, y_{mj})^\mathsf{T}$ of m symbols in $GF(q)$. The linear combination $Y_j = XG_j$ is computed in a symbol-by-symbol manner, meaning that $y_{\ell j} = \sum_{i=1}^{k} x_{\ell i} g_{ij}$ for all $j = 1, 2, \ldots, n$ and $\ell = 1, 2, \ldots m$. Thus, one can think of X and Y in the s.l.e. $Y = XG$ as matrices of size $m \times k$ and $m \times n$, respectively.

Adversary model

We assume that the adversary has access to t storage nodes, and she can observe and modify the equations stored by them. This means that if the adversary has access to storage node j, then she can modify both G_j and Y_j stored by node j. Let $G^* = G + \Delta G$ and $Y^* = Y + \Delta Y$ be the modified generator matrix and the modified code block vector after an attack, where the modifications made by the adversary are contained in matrix ΔG and vector ΔY. We further allow the adversary to compromise the communication links of the t storage nodes. It gives more possibility to the adversary, but does not extend the possible effect of the attack. For simplicity, we refer to nodes that store modified data as compromised nodes, and not distinguish them upon the way of modification.

Note that the adversary has no access to the source nodes, rather she aims at compromising the output of the storage system. The rationale behind this assumption is that storage nodes are exposed to attacks for an extended period of time, whereas the source nodes must be attacked during the limited time period of data generation. Data distribution from the source nodes to the storage nodes typically takes place on a wireless channel, that is exposed to various attacks. Accordingly, our applied model of adversary is realistic in most cases.

Recall that when reconstructing the data blocks, the collector node chooses the k storage nodes, from which it downloads the k linear equations, randomly. Therefore, the adversary has no information on which storage nodes will be chosen when she performs the attack. At the same time, the collector node does not know which storage nodes are compromised. In the sequel, we will assume without loss of generality that the adversary randomly chooses the t storage nodes to be compromised, and the collector node downloads the equations of the first k storage nodes, where the order of the storage nodes is defined randomly by the collector node. Thus, the set of equations downloaded by the collector node is $Z^*_{1..k} = (G^*_{1..k}, Y^*_{1..k})$, where $G^*_{1..k} = (G^*_1, G^*_2, \ldots, G^*_k)$ and $Y^*_{1..k} = (Y^*_1, Y^*_2, \ldots, Y^*_k)$.

Let us now investigate the effect of an attack. The collector node solves the s.l.e. $Y^*_{1..k} = XG^*_{1..k}$ for X, and obtains the result $X^* = Y^*_{1..k}(G^*_{1..k})^{-1}$. Let us suppose for the moment that the adversary modifies only the code blocks, meaning that $G^* = G$. In this case, $X^* = Y^*_{1..k}(G_{1..k})^{-1}$.

The modification induced by the attack in the decoded data blocks can be computed as follows:

$$\begin{aligned}\Delta X &= X^* - X \\ &= Y^*_{1..k}(G_{1..k})^{-1} - X \\ &= (Y_{1..k} + \Delta Y_{1..k})(G_{1..k})^{-1} - X \\ &= \Delta Y_{1..k}(G_{1..k})^{-1}\end{aligned}$$

where in the last step we used that $Y_{1..k}(G_{1..k})^{-1} = X$. This means that (a) if a given row of $\Delta Y_{1..k}$ contains only zeros, then the corresponding row of ΔX will contain only zeros too, and (b) a non-zero element in a given row of $\Delta Y_{1..k}$ will affect the entire corresponding row in ΔX. Thus, a modification made by the adversary in a given row in any of the first k code blocks will, in general, affect all decoded data blocks, but the effect will be limited to the corresponding row.

Now, let us suppose that the adversary modifies only the coefficient vectors, meaning that $Y^* = Y$. In this case, $X^* = Y_{1..k}(G^*_{1..k})^{-1}$. If at least one of the first k coefficient vectors has been modified by the adversary, then $G^*_{1..k} \neq G_{1..k}$, and thus, $(G^*_{1..k})^{-1}$ can be completely different from $(G_{1..k})^{-1}$. Therefore, in general, such a modification affects all decoded data blocks in every row.

If the adversary modifies both the coefficient vectors and the code blocks, then these effects are combined. In the general case, the modification induced by the attack on the decoded data blocks can be derived as follows:

$$\begin{aligned}X + \Delta X &= (Y_{1..k} + \Delta Y_{1..k})(G^*_{1..k})^{-1} \\ (X + \Delta X)G^*_{1..k} &= Y_{1..k} + \Delta Y_{1..k} \\ X\Delta G_{1..k} + \Delta X G^*_{1..k} &= \Delta Y_{1..k} \\ \Delta X &= (\Delta Y_{1..k} - X\Delta G_{1..k})(G^*_{1..k})^{-1}\end{aligned}$$

where in the second step we used that $G^*_{1..k} = G_{1..k} + \Delta G_{1..k}$ and $XG_{1..k} = Y_{1..k}$.

The above formulas imply the following observation. If $\Delta Y_{1...k}$ is controlled by the adversary, meaning that all downloaded equations are from compromised nodes, the value of ΔX can be chosen by the adversary. The adversary can reconstruct X from the contents of the nodes, so she is able to enforce arbitrary $X^* = X + \Delta X$ solution by loading $Y^*_i = X^*_i G_i$ as the modified content of the i-th compromised storage node. As a result, the adversary can not only destroy the original data block vectors, but she can also enforce a particular value. This scenario may occur, if $t \geq k$.

Actually, these observations illustrate the amplification effect of the pollution attack: a small amount of modifications in the stored coded information can result in a large amount of modifications in the decoded data. In the worst case all data blocks are entirely destroyed. This is highly non-desirable, and requires the development of some countermeasures. Below, we address this problem by proposing mechanisms to detect and recover from such attacks.

4.2 Attack detection

THESIS 4.1. *I propose a new algorithm to detect pollution attacks in coding based distributed storage schemes. The algorithm is optimal in terms of communication overhead and computing complexity, and its false negative detection rate can be made negligibly small by appropriate parameter selection. Its false positive detection rate is* $\frac{t}{n-k}$*, where* k *is the number of source nodes,* n *is the number of storage nodes, and* t *is the number of compromised storage nodes. Hence, the false positive detection rate is not negligible, but false alarms can be handled with the recovery algortihms that I propose later in this thesis group. [C1, J2]*

Principle

The basic idea of our attack detection mechanism is the following: We observe that it is very unlikely that the adversary will compromise all the first k equations. Indeed, the probability of this event is around $(t/n)^k$. Thus, some parts of $Y^*_{1..k}$ and $G^*_{1..k}$ are not controlled by the adversary, and for this reason, she cannot enforce a particular solution $X^* = Y^*_{1..k}(G^*_{1..k})^{-1}$. Indeed, X^* will be a random vector in most of the cases, except if all the first k equations are intact, in which case $X^* = X$ will hold.

Now, suppose that we have an additional intact equation: $Y_{k+1} = XG_{k+1}$ (i.e., the collector downloaded $Z_{k+1} = (G_{k+1}, Y_{k+1})$). If X^* is random, then it will not satisfy the additional intact equation with high probability, while it will satisfy it with probability 1 if $X^* = X$. Thus, we can detect if the decoded data block vector X^* is polluted with the help of an additional intact equation.

Algorithm

The proposed attack detection algorithm works in the following way: The collector downloads the first k equations $Z^*_{1..k}$ and computes $X^* = Y^*_{1..k}(G^*_{1..k})^{-1}$. Then, the collector downloads the next equation Z^*_{k+1}. If $Y^*_{k+1} = X^*G^*_{k+1}$, then no attack is detected (and the collector accepts X^* as the correct solution). Otherwise, if $Y^*_{k+1} \neq X^*G^*_{k+1}$, an attack is signaled.

Analysis

In this subsection, we investigate the complexity of the attack detection algorithm, as well as its false negative and false positive error probabilities.

Complexity: We measure the communication complexity in the number of downloaded equations and the computational complexity in the number of s.l.e.'s that we need to solve. Thus, the communication complexity of the proposed attack detection algorithm is $k+1$, and its computational complexity is 1. As the collector needs to download k equations and solve one s.l.e. in any case, the incurred overhead of the attack detection is extremely small: 1 more equation to download.

Probability of a false negative decision: Let us assume for the moment that the adversary does not modify the coefficient vectors, meaning that $G^* = G$. As we saw earlier, in this case, the collector obtains the solution $X^* = X + \Delta Y_{1..k}G^{-1}_{1..k} = X + \Delta X$.

If we further assume that the additional equation that we use for detection is intact, then we have $Z^*_{k+1} = Z_{k+1} = (G_{k+1}, Y_{k+1})$. In this case, the false negative error probability, denoted by P_{fneg}, can be computed as follows:

$$\begin{aligned} P_{fneg} &= \Pr\{Y_{k+1} = X^*G_{k+1} | \Delta Y_{1..k} \neq 0\} \\ &= \Pr\{Y_{k+1} = (X + \Delta X)G_{k+1} | \Delta Y_{1..k} \neq 0\} \\ &= \Pr\{\Delta X G_{k+1} = 0 | \Delta Y_{1..k} \neq 0\} \end{aligned} \tag{22}$$

where in the last step we used that $Y_{k+1} = XG_{k+1}$.

Recall that if $\Delta Y_{1..k}$ has a non-zero element in the i-th row (and $G_{1..k}$ is intact), then ΔX also has some non-zero elements in the i-th row. Otherwise, if the i-th row of $\Delta Y_{1..k}$ contains only zeros, then the i-th row of ΔX contains only zeros too.

We can write the i-th element of $\Delta X G_{k+1}$ as

$$\sum_{\ell=1}^{k} \Delta x_{i\ell} g_{\ell(k+1)} \tag{23}$$

By the argument above, (23) is a non-trivial linear combination of the elements of G_{k+1}. However, the elements of G_{k+1} are chosen randomly, therefore, the probability of (23) being 0 is equal to $1/q$.

From this, it follows that

$$P_{fneg} = \frac{1}{q^{t'}} \tag{24}$$

where t' is the number of rows in $\Delta Y_{1..k}$ that contain non-zero elements. Clearly, in order to maximize the error probability (and hence minimize the success probability) of the detection, the adversary must make all modifications to the code blocks in a single row[9].

Next, we keep the assumption that the adversary does not modify the coefficient vectors (hence $G^* = G$), but we assume that the code block of the additional equation that we use for detection is attacked, meaning that $Z^*_{k+1} = (G_{k+1}, Y^*_{k+1}) = (G_{k+1}, Y_{k+1} + \Delta Y_{k+1})$. In this case, a simple derivation similar to the previous case can be used to arrive to the following result:

$$P_{fneg} = \Pr\{\Delta X G_{k+1} = \Delta Y_{k+1} | \Delta Y_{1..k} \neq 0\} \tag{25}$$

Recall from the previous discussion that the i-th row of ΔX contains only zeros if the i-th row of $\Delta Y_{1..k}$ contains only zeros. In this case, the i-th element of $\Delta X G_{k+1}$ must be a zero too. Thus, if the i-th element in ΔY_{k+1} is not zero, then the above error probability is 0 (i.e., we can detect the attack even though the additional equation used for detection is not intact). On the other hand, if ΔY_{k+1} contains zeros in every row where $\Delta Y_{1..k}$ contains only zeros, then due to the randomness of G_{k+1}, we get again that $P_{fneg} = 1/q^{t'}$, where t' is the number of rows in $\Delta Y_{1..k}$ that contain non-zero elements.

Finally, let us consider the general case when the adversary may modify both the coefficient vectors and the code blocks, hence $\Delta G \neq 0$ and $\Delta Y \neq 0$. Recall that if $\Delta G_{1..k} \neq 0$, then the solution $X^* = Y^*_{1..k}(G^*_{1..k})^{-1}$ obtained from the first k equations is a random vector. It follows that the equation $Y^*_{k+1} = X^* G^*_{k+1}$ holds with probability around $1/q^m$, and thus

$$P_{fneg} \;=\; \Pr\{Y^*_{k+1} = X^* G^*_{k+1} | \Delta G_{1..k} \neq 0\} \;\approx\; \frac{1}{q^m} \tag{26}$$

The conclusion of this analysis is that the probability P_{fneg} of false negative detection is maximized if the adversary makes modifications only in a single row of the code block matrix Y and leaves the coefficient matrix G intact. In this case, $P_{fneg} = 1/q$. Hence, if q is chosen sufficiently large, then the probability of not detecting a pollution attack can be made negligible.

Probability of a false positive decision: Let us assume that the first k equations downloaded by the collector node are intact, meaning that $Z^*_{1..k} = Z_{1..k}$. Thus, the collector computes the correct solution $X^* = Y^*_{1..k}(G^*_{1..k})^{-1} = Y_{1..k}(G_{1..k})^{-1} = X$. If the additional equation downloaded for attack detection is also intact (i.e., $Z^*_{k+1} = Z_{k+1}$), then no attack is detected as $Y^*_{k+1} = Y_{k+1} = XG_{k+1} = X^* G^*_{k+1}$. Thus, an attack may be signaled only in the case when the additional equation is not intact. From this, a good approximation of the probability of a false positive decision, denoted by P_{fpos}, is the following:

$$P_{fpos} \;\approx\; \Pr\{\Delta Z_{k+1} \neq 0 | \Delta Z_{1..k} = 0\} \tag{27}$$

Given that the first k equations are intact, the probability that the $(k+1)$-st equation is

[9]Note that if the code blocks contain standard error detection elements, such as a CRC checksum, then at least 2 rows must be changed by the adversary in every attacked code block. Consequently, in that case, we have that $P_{fneg} \leq 1/q^2$.

also intact is

$$\frac{\binom{n-k-1}{t}}{\binom{n-k}{t}} = \frac{n-k-t}{n-k} \tag{28}$$

where t is the number of randomly chosen storage nodes that are attacked by the adversary. From this, we get that

$$P_{fpos} = 1 - \frac{n-k-t}{n-k} = \frac{t}{n-k} \tag{29}$$

While P_{fpos} is not negligible, false positive decisions do not have serious effects. Indeed, when the attack detection algorithm signals an attack, the recovery procedures described in the next section are executed. These procedures try to recover the original data block vector, and as we will see, they succeed in a few steps when the number of attacked equations is small (which is the true by definition in case of a false positive decision of the attack detection algorithm).

4.3 Recovery from attack

Principle

When the collector node detects that the originally downloaded set $S = Z^*_{1..k}$ of equations is polluted, it can download more equations and use them to *clean* the polluted set S. The basic idea of cleaning is the following: Let us denote the set of equations downloaded for cleaning by C, and let e be an additional equation. We use the equations in C to replace a subset of size $|C|$ of the equations in S. We denote the resulting new set of equations by S'. Then, we run our attack detection mechanism on S' with equation e used for testing. In other words, we solve the s.l.e. corresponding to S' and check if the solution satisfies equation e. If no attack is detected, then we accept the obtained solution as the correct data block vector. Otherwise, we take S again, replace another subset of size $|C|$ of its equations, and run the attack detection again. We repeat these steps until either the cleaning succeeds or all possible subsets of size $|C|$ of set S has been replaced.

Note that if e is intact, C contains only intact equations, and the number of the attacked equations in S is not greater than $|C|$, then the above described procedure eventually succeeds, because eventually we will replace all the attacked equations in S by the intact equations in C. In case of failure, either e is attacked, or C contains an attacked equation, or the number of attacked equations in S is greater than $|C|$. In this case, we may download another set C' of equations such that $|C'| > |C|$, as well as another testing equation e', and try the cleaning of S again.

In the rest of this subsection, we propose two specific recovery algorithms based on the principle described above. As we will see, the first algorithm is optimized for communication complexity, however, its computational complexity does not scale well with k. Nevertheless, it may still be usable for smaller systems. The second algorithm that we propose has improved computational complexity, however, in general, it has a higher communication complexity than the first algorithm has, and it can recover only from attacks where the number of the compromised storage nodes is limited.

THESIS 4.2. *I propose a new algorithm aiming at the recovery from pollution attacks in coding based distributed storage schemes. The algorithm is using a cleaning set whose size is iteratively increased. The success probability of the algorithm is 1, if $t < n - k$, and 0 otherwise, where k is the number of source nodes, n is the number of storage nodes, and t is the number of*

compromised storage nodes. Hence, as k can be an order of magnitude smaller than n, $n-k$ is close to n, and thus, the algorithm is successful even if a large fraction of the storage nodes is attacked. The communication complexity of the algorithm is approximately $\frac{kp+1}{1-p}$, where $p = t/n$. The computational complexity of the algorithm is exponential (see expressions (31) and (33), and Figure 22), but my numerical analysis shows that it can still work in practice for small to medium size systems (i.e., 10-50 source nodes, 100-600 storage nodes, and tolerating 50-10% of compromised storage nodes). [C1, J2]

Algorithm 1

The basic idea of our first algorithm is to start the cleaning with a cleaning set C of size one (i.e., to assume first that there is only one attacked equation in set S), and then, if cleaning fails, to increase the size of C iteratively. In this way, sooner or later, we arrive to a cleaning set C that contains as many intact equations as the number of attacked equations in S. In each iteration, we select all possible subsets of the equations in C and replace with them all possible subsets of equations in S. Thus, eventually, we replace the attacked equations with the intact ones, and arrive to a clean set.

The pseudo-code of the algorithm is presented in Table 3. Its operation is explained as follows: The algorithm first downloads $Z^*_{1..k+1}$ (line 1) and runs the attack detection algorithm on $Z^*_{1..k}$ using Z^*_{k+1} as the testing equation (line 2). If no attack is detected, then $Z^*_{1..k}$ is clean and the algorithm stops (line 3). Otherwise, the algorithm starts the cleaning of $S = Z^*_{1..k}$ (lines 5–24). This is an iterative process, where in each iteration (lines 7-24), exactly one new equation is downloaded (line 8). The newly downloaded equation, denoted by e, becomes the testing equation used for attack detection in the current iteration (line 10). The rest of the equations downloaded so far, not counting the equations in S, constitute the cleaning set denoted by C (line 9). The algorithm takes every possible subset C' of C, such that $|C'| = \tau$ is not greater than k (lines 12–13), and uses the equations in C' to replace τ equations in S in all possible ways (lines 14–16). After each replacement, the attack detection mechanism is executed on the resulting set S' of equations using e as the testing equation (line 17). If no attack is detected, then S' is clean and the algorithm stops (line 18).

Analysis of Algorithm 1

Below, we first analyze the success probability of the algorithm, and then, we analyze its communication complexity and computational complexity.

Success probability: It is easy to see that the algorithm succeeds iff the number t' of the attacked equations in $S = Z^*_{1..k}$ is smaller than the number of the intact equations in the remaining set $Z^*_{k+1..n}$. On the one hand, if this condition holds, then we have at least $t'+1$ intact equations in $Z^*_{k+1..n}$, and therefore, as we continue downloading more and more equations for cleaning, we eventually reach a state where the cleaning set C contains at least t' intact equations and the last downloaded equation e used for attack detection is also intact. In this case, eventually, all the attacked equations in S will be replaced by intact equations from C, hence S will be cleaned. In addition, as e is intact, the attack detection mechanism will indicate no attack, and we can actually realize that S is cleaned.

On the other hand, if t' is not smaller than the number of the intact equations in $Z^*_{k+1..n}$, then either the cleaning set C contains fewer than t' intact equations, and hence, S cannot be cleaned, or C contains exactly t' intact equations and S can be cleaned, but we have no more intact equation for attack detection purposes, and therefore, we cannot realize that S is cleaned.

Given that there are t attacked equations all together, and t' of them are in $Z^*_{1..k}$, we get that the number of intact equations in $Z^*_{k+1..n}$ is $(n-k)-(t-t')$. Hence, the algorithm succeeds

```
1   download Z*_{1..k+1}
2   if attack_detection(Z*_{1..k}, Z*_{k+1}) = no attack
3      return Z*_{1..k}
4   endif

5   let S = Z*_{1..k}
6   let w = 1
7   while w < n − k
8      download Z*_{k+w+1}
9      let C = Z*_{k+1..k+w}
10     let e = Z*_{k+w+1}
11     for τ = 1 to min(w, k)
12        for every possible selection s_1
           of τ elements out of w elements
13           let C' be the subset of equations
              determined by s_1 in C
14           for every possible selection s_2
              of τ elements out of k elements
15              let S' = S
16              replace the equations determined by s_2
                in S' with the equations in C'
17              if attack_detection(S', e) = no attack
18                 return S'
19              end if
20           end for
21        end for
22     end for
23     let w = w + 1
24  end while
```

Table 3: Pseudo-code of Algorithm 1.

iff $t' < (n-k) - (t-t')$, or equivalently, $t < n-k$. Thus, we get that

$$P_{success} = \begin{cases} 1 & \text{if } t < n-k \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

As k can be an order of magnitude smaller than n, $n-k$ is close to n, and thus, the algorithm is successful even if a large fraction of the equations is attacked.

Communication complexity: Recall that we measure the communication complexity in the number of the downloaded equations. As the algorithm downloads a new equation in every iteration, its communication complexity depends on the number of the iterations it performs. More precisely, if the algorithm performs R iterations, then its communication complexity is $(k+1)+R$, because it downloads $k+1$ equations at the beginning before the iterative phase is started. As k is a fixed parameter, we are interested in the characterization of R.

The algorithm stops as soon as the following two conditions hold: (a) the number of intact equations in the cleaning set C is equal to the number of attacked equations in S, and (b) the last downloaded equation e used for attack detection is intact. Indeed, if condition (a) is satisfied, then eventually the intact equations in C will be used to replace the attacked equations

in S, hence S will be cleaned. If, in addition, condition (b) is satisfied, then the attack detection mechanism will indicate no attack, and we can actually realize that S is cleaned. Thus, R is the number of equations needed to be downloaded to satisfy the two conditions above.

It must be clear that if S contains t' attacked equations, then $C \cup \{e\}$ must contain at least $t'+1$ intact equations, as otherwise, we cannot clean S and realize that it has been cleaned at the same time. Thus, R is minimal in the sense that for $R' < R$ downloaded equations, $C \cup \{e\}$ contains fewer than $t'+1$ intact equations, and hence, the algorithm cannot succeed. This means that our algorithm is optimal in terms of communication complexity.

We give an estimation of R in the following way. Let $p = t/n$, and let W_1 denote the number of equations that need to be downloaded in order for the downloaded set of equations to contain exactly the same number of intact equations, on average, as the number of attacked equations in S. The average number of attacked equations in set S is approximately kp. The average number of intact equations among the W_1 equations is approximately $W_1(1-p)$. Hence, we get that $W_1 \approx kp/(1-p)$. Furthermore, let W_2 denote the average number of equations that need to be downloaded until we download an intact equation. Clearly, $W_2 \approx 1/(1-p)$. Thus, when $W_1 + W_2$ equations are downloaded, both conditions (a) and (b) are satisfied. In other words, a good estimate of R is

$$R \;\approx\; W_1 + W_2 \;\approx\; \frac{kp+1}{1-p} \tag{31}$$

Computational complexity: Recall that we measure the computational complexity in the number of s.l.e.'s that need to be solved. In our case, each call to the attack detection algorithm requires the solution of an s.l.e.

The worst case computational complexity $\mathcal{P}_{worst}$ of the algorithm can be easily determined by inspecting the structure of the nested loops in the algorithm:

$$\mathcal{P}_{worst} \approx \sum_{w=1}^{R} \sum_{\tau=1}^{\min(w,k)} \binom{w}{\tau} \binom{k}{\tau} \tag{32}$$

where R is the number of iterations, which we can estimate according to (31).

For the derivation of the average case computational complexity $\mathcal{P}_{avg}$, we assume that the number of the attacked equations in S is t', where the average value of t' is kt/n. We make the following observations:

- All but the last iterations of the algorithm execute fully. (term (33) in the sum below)
- In the last iteration, the loops that try to clean S with $\tau < t'$ equations from C also execute fully. (term (34) in the sum below)
- When we use $\tau = t'$ equations from C for cleaning, we have to process on average half of the possible selections of t' equations from C until we end up with the subset that contains the t' intact equations of C. For all those selections, the inner loop executes fully and we must process all the possible selections of t' equations from S. (term (35) in the sum below)
- Finally, when we select the subset of C that contains the t' intact equations, we have to process on average half of the possible selections of t' equations from S until we end up with the t' attacked equations of S. (term (36) in the sum below)

Thus, we get that

$$\mathcal{P}_{avg} \approx \sum_{w=1}^{R-1} \sum_{\tau=1}^{\min(w,k)} \left(\begin{array}{c} w \\ \tau \end{array} \right) \left(\begin{array}{c} k \\ \tau \end{array} \right) + \tag{33}$$

$$\sum_{\tau=1}^{t'-1} \left(\begin{array}{c} R \\ \tau \end{array} \right) \left(\begin{array}{c} k \\ \tau \end{array} \right) + \tag{34}$$

$$\frac{1}{2} \left(\begin{array}{c} R \\ t' \end{array} \right) \left(\begin{array}{c} k \\ t' \end{array} \right) + \tag{35}$$

$$\frac{1}{2} \left(\begin{array}{c} k \\ t' \end{array} \right) \tag{36}$$

Figure 22 shows the average computational complexity of Algorithm 1 as a function of the number t of attacked equations. The different curves belong to different values of n and k, and the computation is based on the formula given above. Note the logarithmic scale of the y axis.

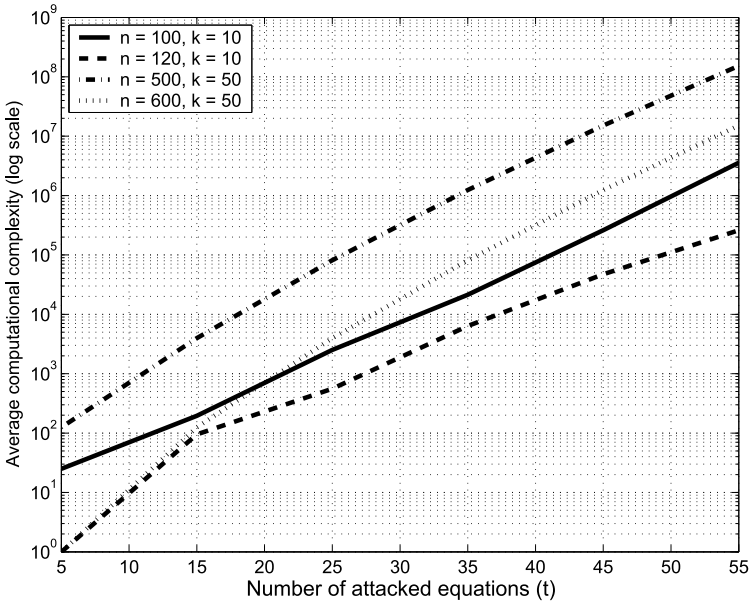


Figure 22: Average computational complexity of Algorithm 1 as a function of the number t of attacked equations. The different curves belong to different values of n and k.

As we can see, the computational complexity of Algorithm 1 increases rapidly with the number t of attacked equations. Still, for the presented values of n, k, and t, it does not exceed $10^9 \approx 2^{30}$, which is still feasible. Thus, for small systems, where k is in the range of 10 – 50, Algorithm 1 provides a practical solution: it succeeds in recovering from attacks even if the number t of the attacked equations is very large, its communication complexity is optimal, and it is still computationally feasible up to $t \approx 55$ attacked equations. Note that in the case of $n = 100$, $t \approx 55$ means that more than half of the storage nodes are compromised, yet Algorithm 1 can recover from the attack and it is practically feasible. In the case of $n = 500$, Algorithm 1 can cope only with a weaker attacker that can compromise around 10% of the storage nodes.

While Algorithm 1 is a good choice for small scale systems ($n = 100$ – 600 and $k = 10$ – 50), it is computationally infeasible for larger systems (e.g., when k is around 100) even if we assume that t is limited.

THESIS 4.3. *I propose another algorithm for recovering from pollution attacks in coding based distributed storage schemes that uses a fixed size cleaning set, and hence, it has reduced computational complexity. I show, by means of simulations, that the success rate of the algorithm is close to 1 up to 10% compromised storage nodes, and then it decreases quickly. The communication complexity of the algorithm grows with the number of compromised storage nodes, but it remains below $n/2$ up to 10% compromised nodes, and it is close to optimal up to 5% compromised nodes. The computational complexity of the algorihm is better than that of my first recovery algorithm, in particular, with the same amount of computation, the second algorithm can handle an order of magnitude larger systems (100 source nodes and 1000 storage nodes) up to 5-10% compromised storage nodes. [J2]*

Algorithm 2

Contrary to our first algorithm, where the size of the cleaning set is iteratively increased, our second algorithm uses a fixed size cleaning set C. In this way, the number of the possible selections of the different subsets of C does not grow, and hence, the computational complexity of the algorithm scales better with k. Instead of iteratively increasing C, this algorithm changes the fixed size sets S and C in each iteration. In effect, S and C consist of the equations that are taken from a fixed size window that slides over Z^*.

The pseudo-code of the algorithm is presented in Table 4. First, we download the equations $Z^*_{1..k+1}$ and perform attack detection in a way similar to Algorithm 1 (lines 1–4). If no attack is detected, then the algorithm stops; otherwise, we start an iterative cleaning process (lines 5–26). As we said above, in this algorithm, the size w of the cleaning set C is a fixed value $\lceil \alpha k \rceil$ (line 5), where α is an input parameter. We download the equations $Z^*_{k+2..k+w}$ (line 6), and initialize the set S to be cleaned with $Z^*_{1..k}$ and the cleaning set C with $Z^*_{k+1..k+w}$. Both sets change in each iteration, and we use variables i_S and i_C to point to the first equations of them in the current iteration. Similarly, i_e points to the equation that we use in attack detection for testing. Variables i_S, i_C, and i_e are initialized (line 7) and the iteration starts. In each iteration (lines 8–26), we download exactly one new equation (line 9), which becomes the equation that is used as the testing equation in attack detection (line 12). The algorithm takes every possible subset C' of C, such that $|C'| = \tau$ is not greater than τ_{max} (lines 13–15), and uses the equations in C' to replace τ equations in S in all possible ways (lines 16–18). Here τ_{max} is another input parameter that limits the computational complexity of the algorithm by limiting the size of the subsets of the equations that we choose from C and replace in S. After each replacement, the attack detection mechanism is executed on the resulting set S' of equations using e as the testing equation (line 19). If no attack is detected, then S' is clean and the algorithm stops (line 20). Otherwise, we increment each of our pointers i_S, i_C, and i_e (line 25), and continue the iteration. Note that set $S \cup C \cup \{e\}$ consists of the equations in a sliding window of size $k + w + 1$ that slides over Z^* until either cleaning is successful or we downloaded all equations in Z^*.

Analysis of Algorithm 2 by simulations

Algorithm 2 is more difficult to examine analytically, therefore, we used simulations, written in Matlab, to investigate its performance. In our simulations, we set $n = 1000$ and $k = 100$, and we range the value of τ_{max} over the values $\{4, 5, 6\}$. For each value of τ_{max}, we set

$$\alpha = \frac{\tau_{max}}{k - \tau_{max}} \tag{37}$$

The rationale behind this setting of α is the following: Intuitively, we are prepared to clean at most τ_{max} attacked equation in S. If we assume that S, which has size k, contains τ_{max}

1 **download** $Z^*_{1..k+1}$
2 **if** *attack_detection*($Z^*_{1..k}$, Z^*_{k+1}) = no attack
3 **return** $Z^*_{1..k}$
4 **endif**

5 **let** $w = \lceil \alpha k \rceil$
6 **download** $Z^*_{k+2..k+w}$
7 **let** $i_S = 1$, $i_C = i_S + k$, $i_e = i_C + w$
8 **while** $i_e \leq n$
9 **download** $Z^*_{i_e}$
10 **let** $S = Z^*_{i_S..i_S+k-1}$
11 **let** $C = Z^*_{i_C..i_C+w-1}$
12 **let** $e = Z^*_{i_e}$
13 **for** $\tau = 1$ **to** τ_{max}
14 **for** every possible selection s_1
of τ elements out of w elements
15 **let** C' be the subset of equations
determined by s_1 in C
16 **for** every possible selection s_2
of τ elements out of k elements
17 **let** $S' = S$
18 *replace* the equations determined by s_2
in S' with the equations in C'
19 **if** *attack_detection*(S', e) = no attack
20 **return** S'
21 **end if**
22 **end for**
23 **end for**
24 **end for**
25 **let** $i_S = i_S + 1$, $i_C = i_S + k$, $i_e = i_C + w$
26 **end while**

Table 4: Pseudo-code of Algorithm 2.

attacked equation, then we may estimate the probability that a given equation in Z^* is attacked as τ_{max}/k. Thus, the number of intact equations in C, which has size w, can be estimated as $w(1 - \tau_{max}/k)$. In order to be able to clean S, the number of intact equations in C must be at least τ_{max}. Thus, we must have that

$$\tau_{max} \leq w \left(1 - \frac{\tau_{max}}{k}\right) \tag{38}$$

from which

$$w \geq \frac{\tau_{max}}{1 - \frac{\tau_{max}}{k}} = \frac{\tau_{max}}{k - \tau_{max}} k \tag{39}$$

Moreover, for each setting of τ_{max} and α, we range the number t of attacked equations form 10 to 150 with a step size of 10. For each setting of the parameters, we run 100 simulations, where the t attacked equations are chosen uniformly at random in the set Z^* of n equations.

We are interested in the success probability of the algorithm, which we estimate as the fraction of the simulation runs, for a given setting of the parameters, where the algorithm succeeds.

In addition, we are interested in the average communication and computational complexity of the algorithm, which we obtain as the mean of the communication and computational complexities, respectively, of the simulation runs for a given setting of the parameters.

Success probability: Figure 23 shows the success probability of Algorithm 2 as the function of the number t of the attacked equations. The different curves belong to different values of τ_{max}.

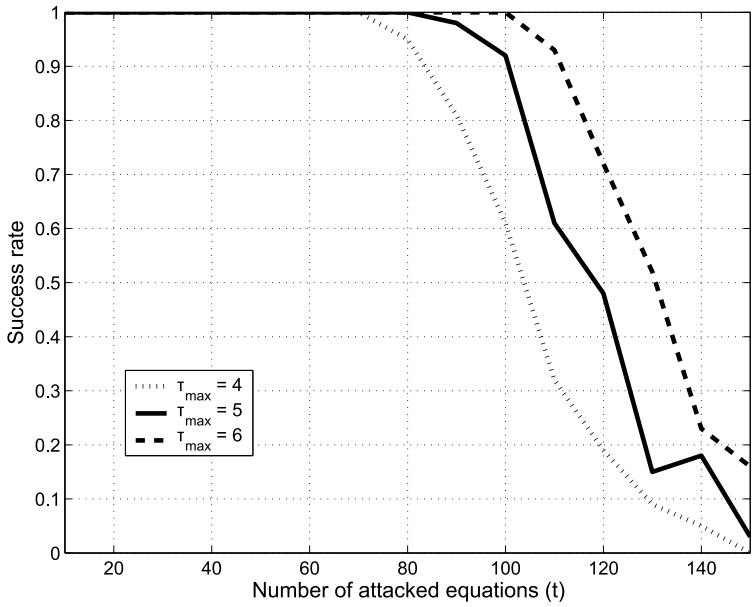


Figure 23: Success rate of Algorithm 2 as a function of the number t of attacked equations. $n = 1000$ and $k = 100$.

As we can see, the success probability of the algorithm is larger than 90% until a threshold value of t, and begins to decrease rapidly after the threshold. This threshold value is approximately $t = 85$, $t = 100$, and $t = 110$, for $\tau_{max} = 4$, $\tau_{max} = 5$, and $\tau_{max} = 6$, respectively. Thus, as we expected, if we increase τ_{max}, the algorithm ensures recovery from stronger attacks that involve more attacked equations. Unfortunately, as we will see below, the computational complexity increases too.

Recall that in case of Algorithm 1, the success probability remained one until the threshold $t = n - k - 1$, which would be $t = 899$ for $n = 1000$ and $k = 100$. This threshold is much larger than the threshold values that we got for Algorithm 2. Despite of this, the threshold values that we obtained are still surprisingly large given that the algorithm is prepared to handle much smaller number of attacked equations. Indeed, when $\tau_{max} = 4$, the algorithm is prepared to clean 4 attacked equations in a set of size $k = 100$, which means 40 attacked equations in the entire set of size $n = 1000$. However, the algorithm succeeds with high probability even if the number of attacked equations is around 85. A similar observation can be made for the other values of τ_{max}.

The reason of this is that when $t = 85$, the average number of attacked equations in a set of size $k = 100$ is 8.5, but this means that there are sets with a smaller number of attacked equations. Apparently, we can find a set with not more than 4 attacked equations with a rather high probability among the sets that we obtain by sliding a window of size $k = 100$ over the entire set Z^* of equations. A similar argument applies for the other cases.

Communication complexity: Figure 24 shows the average communication complexity (i.e., the number of the downloaded equations) of Algorithm 2. The different curves belong to dif-

ferent values of τ_{max}. We truncated the plot at $t = 120$, because above that value, the success probability of the algorithm is rather poor anyway, hence, we are not really interested in its complexity.

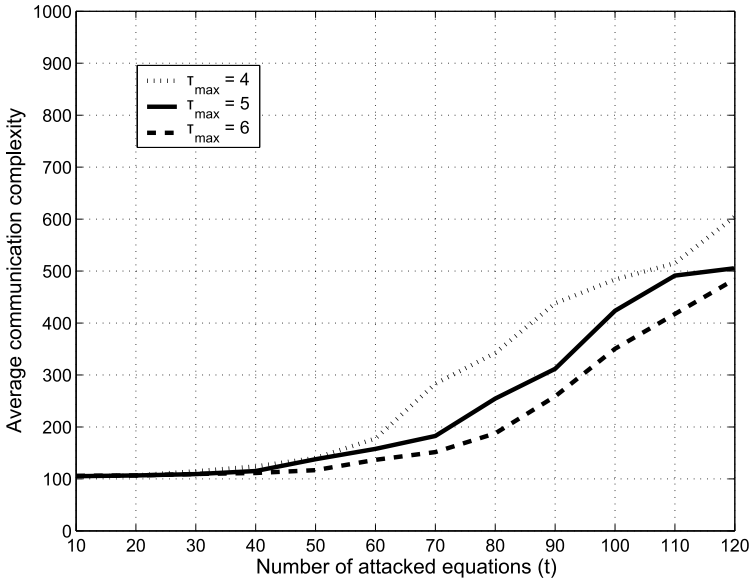


Figure 24: Average communication complexity of Algorithm 2 as a function of the number t of attacked equations. $n = 1000$ and $k = 100$.

As we expected, the average communication complexity increases as the number t of the attacked equations increases, because it becomes more difficult to find, at the same time, a set S of k equations that contains no more than a fixed τ_{max} attacked equations, and a set C of αk equations that contains at least τ_{max} intact equations. However, on average, the number of the downloaded equations is smaller than half of the total number n of equations, and the standard deviation is also acceptably small. In particular, when the number t of attacked equations is around 50 (i.e., only 5% of the storage nodes are compromised), the communication overhead is very small.

We can also observe that the communication complexity increases as τ_{max} decreases. Unfortunately, as we will se below, the price of this decrease is the substantially increased computational complexity.

Computational complexity: Figure 25 shows the computational complexity (i.e., the number of s.l.e.'s that need to be solved) of Algorithm 2 as a function of the number t of attacked equations. The different curves belong to different values of τ_{max}. Note the logarithmic scale of the y axis.

We can observe that the computational complexity increases quickly as the number t of the attacked equations increases, as well as with the increase of τ_{max}. Indeed, incrementing τ_{max} by one results, roughly, in an order of magnitude more computations. The best trade-off seems to be the $\tau_{max} = 4$ case, where Algorithm 2 can handle up to $t = 50$ attacked equations (i.e., up to 5% of the total number of equations) with a very low communication overhead, and still reasonable computational complexity ($10^9 \approx 2^{30}$ s.l.e.'s to solve).

4.4 Summary

In this thesis group, we addressed the problem of pollution attacks in coding based distributed storage schemes, and we proposed specific algorithms for detecting and recovering from such

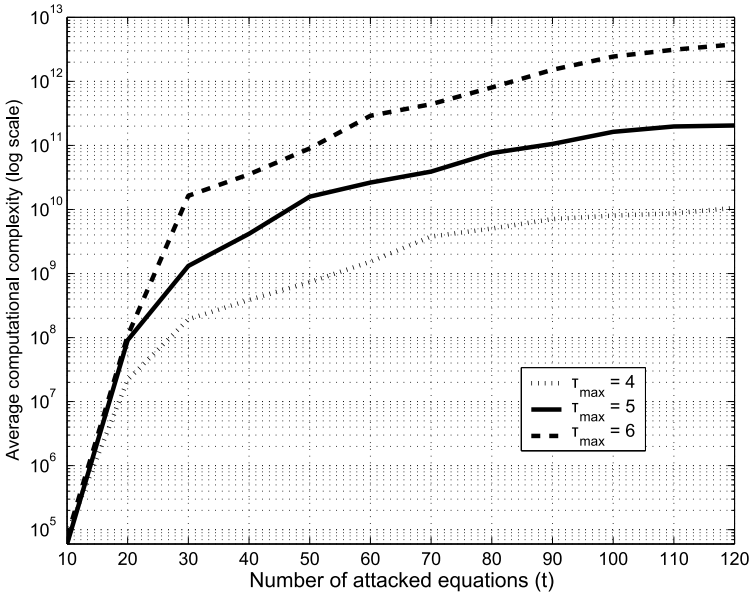


Figure 25: Average computational complexity of Algorithm 2 as a function of the number t of attacked equations. $n = 1000$ and $k = 100$.

attacks. A salient feature of the proposed algorithms is that they are not based on cryptographic checksums or digital signatures, which are traditionally used for providing integrity services. Instead, we take advantage of the inherent redundancy in such distributed storage systems. In particular, our approach is to obtain more encoded packets than strictly necessary for the decoding of the original data, and to use those additional encoded packets for attack detection and recovery purposes. Both detection and recovery require only solving systems of linear equations over a finite field.

The attack detection algorithm that we proposed is effective and extremely efficient both in terms of communication and computational overhead. In addition, we proposed two recovery algorithms. The first algorithm is optimal in terms of communication complexity, and it ensures recovery from attacks even if a large fraction of the encoded packets are modified, but it does not scale up to large systems in terms of computational complexity. It is still a practical solution, though, for smaller systems. The second algorithm that we proposed scales better, but it is less effective in terms of recovery capabilities and less efficient in terms of communication overhead.

5 Efficient private authentication in resouce constrained environments

THESIS GROUP 5. *I propose a new method to design optimized key-trees for tree-based private authentication schemes. For this, I propose a benchmark metric to measure the level of privacy provided by a given key-tree, and I propose a key-tree construction algorithm that maximizes this metric under the constraint of keeping the authentication delay in the system below a given threshold. I also give an approximation of the achieved level of privacy when any number c of the system's members are compromised. I show, by means of simulations, that the approximation formula is sharp. This approximation can, thus, be used to compare different key-trees in terms of the level of privacy that they achieve. [C3]*

Entity authentication is the process whereby a party (the prover) corroborates its identity to another party (the verifier). Entity authentication is often based on authentication protocols in which the parties pass messages to each other. These protocols are engineered in such a way that they resist various types of impersonation and replay attacks [6]. However, less attention is paid to the requirement of preserving the privacy of the parties (typically that of the prover) with respect to an eavesdropping third party. Indeed, in many of the well-known and widely used authentication protocols (e.g., [25]) the identity of the prover is sent in cleartext, and hence, it is revealed to an eavesdropper.

One approach to solve this problem is based on public key cryptography, and it consists in encrypting the identity information of the prover with the public key of the verifier so that no one but the verifier can learn the prover's identity. Another approach, also based on public key techniques, is that the parties first run an anonymous Diffie-Hellman key exchange and establish a confidential channel, through which the prover can send its identity and authentication information to the verifier in a second step. These approaches provide appropriate solution to the problem only if the parties can afford public key cryptography. In many applications, such as in wireless sensor networks or in case of low cost RFID tags, this is not the case.

The problem of using symmetric key encryption to hide the identity of the prover is that the verifier does not know which symmetric key it should use to decrypt the encrypted identity, because the appropriate key cannot be retrieved without the identity. The verifier may try all possible keys in its key database until one of them properly decrypts the encrypted identity, but this would increase the authentication delay if the number of potential provers is large. Long authentication delays are usually not desirable, moreover, in some cases, they may not even be acceptable. As an example, let us consider contactless smart card based electronic tickets in public transportation: the number of smart cards in the system (i.e., the number of potential provers) may be very large in big cities, while the time needed to authenticate a card should be short in order to ensure a high throughput of passengers and avoid long queues at entry points.

Molnar and Wagner proposed an elegant approach to privacy protecting authentication [32] that is based on symmetric key cryptography while still ensuring short authentication delays. More precisely, the complexity of the authentication procedure in the Molnar-Wagner scheme is logarithmic in the number of potential provers, in contrast with the linear complexity of the naïve key search approach. The main idea of Molnar and Wagner is to use key-trees (see Figure 26 for illustration). A key-tree is a tree where a unique key is assigned to each edge. The leaves of the tree represent the potential provers, which we will call members in the sequel. Each member possesses the keys assigned to the edges of the path starting from the root and ending in the leaf that corresponds to the given member. The verifier knows all keys in the tree. In order to authenticate itself, a member uses all of its keys, one after the other, starting from the first level of the tree and proceeding towards lower levels. The verifier first determines which first level key has been used. For this, it needs to search through the first level keys only.

Once the first key is identified, the verifier continues by determining which second level key has been used. However, for this, it needs to search through those second level keys only that reside below the already identified first level key in the tree. This process is continued until all keys are identified, which at the end, identify the authenticating member. The key point is that the verifier can reduce the search space considerably each time a key is identified, because it should consider only the subtree below the recently identified key.

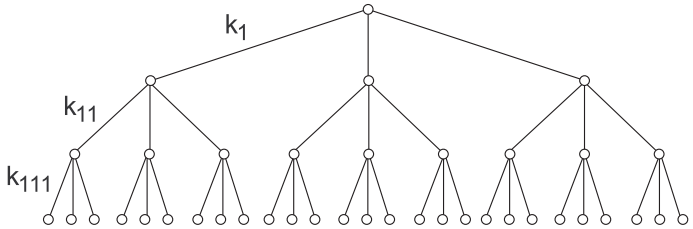


Figure 26: Illustration of a key-tree. There is a unique key assigned to each edge. Each leaf represents a member of the system that possesses the keys assigned to the edges of the path starting from the root and ending in the given leaf. For instance, the member that belongs to the leftmost leaf in the figure possesses the keys k_1, k_{11}, and k_{111}.

The problem of the above described tree-based approach is that upper level keys in the tree are used by many members, and therefore, if a member is compromised and its keys become known to the adversary, then the adversary gains partial knowledge of the key of other members too. This obviously reduces the privacy provided by the system to its members, since by observing the authentication of an uncompromised member, the adversary can recognize the usage of some compromised keys, and therefore its uncertainty regarding the identity of the authenticating member is reduced (it may be able to determine which subtree the member belongs to).

One interesting observation is that the naïve, linear key search approach can be viewed as a special case of the key-tree based approach, where the key-tree has a single level and each member has a single key. Regarding the above described problem of compromised members, the naïve approach is in fact optimal, because compromising a member does not reveal any key information of other members. At the same time, as we saw above, the authentication delay is the worst in this case. On the other hand, in case of a binary key-tree, we can observe that the compromise of a single member *strongly* affects the privacy of the other members, while at the same time, the binary tree is very advantageous in terms of authentication delay. Thus, there seems to be a trade-off between the level of privacy provided by the system and the authentication delay, which depends on the parameters of the key-tree, but it is far from obvious to see how the optimal key-tree should look like.

In this thesis group, we address this problem, and we show how to find optimal key-trees. More precisely, our main contributions are the following:

- We propose a benchmark metric for measuring the resistance of the system to a single compromised member based on the concept of anonymity sets.
- We introduce the idea of using different branching factors at different levels of the key-tree; the advantage is that the system's resistance to single member compromise can be increased while still keeping the authentication delay short.
- We propose an algorithm for determining the optimal parameters of the key-tree, where optimal means that resistance to single member compromise is maximized, while the authentication delay is kept below a predefined threshold.

- In the general case, when any member can be compromised, we give a lower bound on the level of privacy provided by the system, and present some simulation results that show that this lower bound is sharp. This allows us to compare different systems based on their lower bounds.

In summary, we propose *practically usable techniques* for designers of key-tree based authentication systems.

5.1 Resistance to single member compromise

THESIS 5.1. *I propose a benchmark metric for measuring the resistance of the system to a single compromised member based on the concept of anonymity sets. I propose a new algorithm for determining the optimal parameters of the key-tree, where optimal means that resistance to single member compromise is maximized, while the authentication delay is kept below a predefined threshold. I prove (Theorem 5.1) the optimality of the proposed key-tree construction algorithm. [C3]*

There are different ways to measure the level of anonymity provided by a system [15, 37]. Here we will use the concept of anonymity sets [13]. The anonymity set of a member v is the set of members that are indistinguishable from v from the adversary's point of view. The size of the anonymity set is a good measure of the level of privacy provided for v, because it is related to the level of uncertainty of the adversary. Clearly, the larger the anonymity set is, the higher the level of privacy is. The minimum size of the anonymity set is 1, and its maximum size is equal to the number of all members in the system. In order to make the privacy measure independent of the number of members, one can divide the anonymity set size by the total number of members, and obtain a normalized privacy measure between 0 and 1. Such normalization makes the comparison of different systems easier.

Now, let us consider a key-tree with ℓ levels and branching factors $b_1, b_2, \ldots, b_\ell$ at the levels, and let us assume that exactly one member is compromised (see Figure 27 for illustration). Knowledge of the compromised keys allows the adversary to partition the members into partitions $P_0, P_1, P_2, \ldots$, where

- P_0 contains the compromised member only,
- P_1 contains the members the parent of which is the same as that of the compromised member, and that are not in P_0,
- P_2 contains the members the grandparent of which is the same as that of the compromised member, and that are not in $P_0 \cup P_1$,
- etc.

Members of a given partition are indistinguishable for the adversary, while it can distinguish between members that belong to different partitions. Hence, each partition is the anonymity set of its members.

The level of privacy provided by the system can be characterized by the level of privacy provided to a randomly selected member, or in other words, by the expected size of the anonymity set of a randomly selected member. By definition, the expected anonymity set size is:

$$\bar{S} = \sum_{i=0}^{\ell} \frac{|P_i|}{N} |P_i| = \sum_{i=0}^{\ell} \frac{|P_i|^2}{N} \tag{40}$$

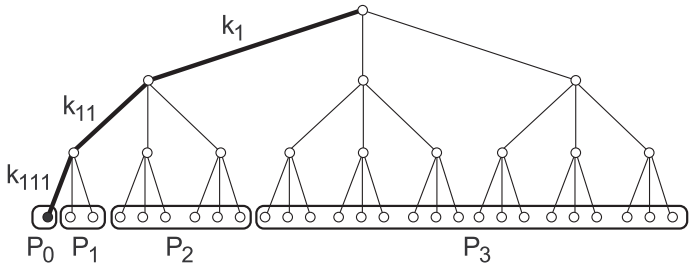


Figure 27: Illustration of what happens when a single member is compromised. Without loss of generality, we assume that the member corresponding to the leftmost leaf in the figure is compromised. This means that the keys k_1, k_{11}, and k_{111} become known to the adversary. This knowledge of the adversary partitions the set of members into anonymity sets P_0, P_1, ... of different sizes. Members that belong to the same partition are indistinguishable to the adversary, while it can distinguish between members that belong to different partitions. For instance, the adversary can recognize a member in partition P_1 by observing the usage of k_1 and k_{11} but not that of k_{111}, where each of these keys are known to the adversary. Members in P_3 are recognized by not being able to observe the usage of any of the keys known to the adversary.

where N is the total number of members, and $|P_i|/N$ is the probability of selecting a member from partition P_i. We define the *resistance to single member compromise*, denoted by R, as the normalized expected anonymity set size, which can be computed as follows:

$$\begin{aligned} R &= \frac{\bar{S}}{N} = \sum_{i=0}^{\ell} \frac{|P_i|^2}{N^2} \\ &= \frac{1}{N^2}\left(1 + (b_\ell - 1)^2 + ((b_{\ell-1} - 1)b_\ell)^2 + \ldots + ((b_1 - 1)b_2 b_3 \ldots b_\ell)^2\right) \\ &= \frac{1}{N^2}\left(1 + (b_\ell - 1)^2 + \sum_{i=1}^{\ell-1}(b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2\right) \end{aligned} \tag{41}$$

where we used that

$$\begin{aligned} |P_0| &= 1 \\ |P_1| &= b_\ell - 1 \\ |P_2| &= (b_{\ell-1} - 1)b_\ell \\ |P_3| &= (b_{\ell-2} - 1)b_{\ell-1}b_\ell \\ \ldots &\quad \ldots \\ |P_\ell| &= (b_1 - 1)b_2 b_3 \ldots b_\ell \end{aligned}$$

As its name indicates, R characterizes the loss of privacy due to the compromise of a single member of the system. If R is close to 1, then the expected anonymity set size is close to the total number of members, and hence, the loss of privacy is small. On the other hand, if R is close to 0, then the loss of privacy is high, as the expected anonymity set size is small. We use R as a benchmark metric based on which different systems can be compared.

Obviously, a system with greater R is better, and therefore, we would like to maximize R. However, there are some constraints. We define the *maximum authentication delay*, denoted by D, as the number of basic operations needed to authenticate any member in the worst case. The maximum authentication delay in case of key-tree based authentication can be computed

as $D = \sum_{i=1}^{\ell} b_i$. In most practical cases, there is an upper bound D_{max} on the maximum authentication delay allowed in the system. Therefore, in practice, the designer's task is to maximize R under the constraint that $D \leq D_{max}$.

5.2 Optimal trees in case of single member compromise

The problem of finding the best branching factor vector can be described as an optimization problem as follows: *Given the total number N of members and the upper bound D_{max} on the maximum authentication delay, find a branching factor vector $B = (b_1, b_2, \ldots b_\ell)$ such that $R(B)$ is maximal subject to the following constraints:*

$$\prod_{i=1}^{\ell} b_i = N \tag{42}$$

$$\sum_{i=1}^{\ell} b_i \leq D_{max} \tag{43}$$

We analyze this optimization problem through a series of lemmas that will lead to an algorithm that solves the problem. Our first lemma states that we can always improve a branching factor vector by ordering its elements in decreasing order, and hence, in the sequel we will consider only ordered vectors:

Lemma 5.1. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let B be a branching factor vector and let B^* be the vector that consists of the sorted permutation of the elements of B in decreasing order. If B satisfies the constraints of the optimization problem defined above, then B^* also satisfies them, and $R(B^*) \geq R(B)$.*

The following lemma provides a lower bound and an upper bound for the resistance to single member compromise:

Lemma 5.2. *Let $B = (b_1, b_2, \ldots b_\ell)$ be a sorted branching factor vector (i.e., $b_1 \geq b_2 \geq \ldots \geq b_\ell$). We can give the following lower and upper bounds on $R(B)$:*

$$\left(1 - \frac{1}{b_1}\right)^2 \leq R(B) \leq \left(1 - \frac{1}{b_1}\right)^2 + \frac{4}{3b_1^2} \tag{44}$$

Let us consider the bounds in Lemma 5.2. Note that the branching factor vector is ordered, therefore, b_1 is not smaller than any other b_i. We can observe that if we increase b_1, then the difference between the upper and the lower bounds decreases, and $R(B)$ gets closer to 1. Intuitively, this implies that in order to find the solution to the optimization problem, b_1 should be maximized. The following lemma underpins this intuition formally:

Lemma 5.3. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$ be two sorted branching factor vectors that satisfy the constraints of the optimization problem defined above. Then, $b_1 > b'_1$ implies $R(B) \geq R(B')$.*

Lemma 5.3 states that given two branching factor vectors, the one with the larger first element is always at least as good as the other. The next lemma generalizes this result by stating that given two branching factor vectors the first j elements of which are equal, the vector with the larger $(j+1)$-st element is always at least as good as the other.

Lemma 5.4. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$ be two sorted branching factor vectors such that $b_i = b'_i$ for all $1 \leq i \leq j$ for some $j < \min(\ell, \ell')$, and both B and B' satisfy the constraints of the optimization problem defined above. Then, $b_{j+1} > b'_{j+1}$ implies $R(B) \geq R(B')$.*

We will now present an algorithm that finds the solution to the optimization problem. However, before doing that, we need to introduce some further notations. Let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$. Then

- $\prod(B)$ denotes $\prod_{i=1}^{\ell} b_i$;
- $\sum(B)$ denotes $\sum_{i=1}^{\ell} b_i$;
- $\{B\}$ denotes the set $\{b_1, b_2, \ldots, b_\ell\}$ of the elements of B;
- $B' \subseteq B$ means that $\{B'\} \subseteq \{B\}$;
- if $B' \subseteq B$, then $B \setminus B'$ denotes the vector that consists of the elements of $\{B\} \setminus \{B'\}$ in decreasing order;
- if b is a positive integer, then $b|B$ denotes the vector $(b, b_1, b_2, \ldots, b_\ell)$.

We define our algorithm as a recursive function f, which takes two input parameters, a vector B of positive integers, and another positive integer d, and returns a vector of positive integers. In order to compute the optimal branching factor vector for a given N and D_{max}, f should be called with the vector that contains the prime factors of N, and D_{max}. For instance, if $N = 27000$ and $D_{max} = 90$, then f should be called with $B = (5, 5, 5, 3, 3, 3, 2, 2, 2)$ and $d = 90$. Function f will then return the optimal branching factor vector.

Function f is defined as follows:

$f(B, d)$
1 if $\sum(B) > d$ then exit (no solution exists)
2 else find $B' \subseteq B$ such that
$\prod(B') + \sum(B \setminus B') \leq d$ and
$\prod(B')$ is maximal
3 if $B' = B$ then return $(\prod(B'))$
4 else return $\prod(B')|f(B \setminus B', d - \prod(B'))$

The operation of the algorithm can be described as follows: The algorithm starts with a branching factor vector consisting of the prime factors of N. This vector satisfies the first constraint of the optimization problem by definition. If it does not satisfy the second constraint (i.e., it does not respect the upper bound on the maximum authentication delay), then no solution exists. Otherwise, the algorithm successively improves the branching factor vector by maximizing its elements, starting with the first element, and then proceeding to the next elements, one after the other. Maximization of an element is done by joining as yet unused prime factors until the resulting divisor of N cannot be further increased without violating the constraints of the optimization problem.

Theorem 5.1. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let B be a vector that contains the prime factors of N. Then, $f(B, D_{max})$ is an optimal branching factor vector for N and D_{max}.*

Proof. We will give a sketch of the proof. Let $B^* = f(B, D_{max})$, and let us assume that there is another branching factor vector $B' \neq B^*$ that also satisfies the constraints of the optimization problem and $R(B') > R(B^*)$. We will show that this leads to a contradiction, hence B^* should be optimal.

Let $B^* = (b_1^*, b_2^*, \ldots, b_{\ell^*}^*)$ and $B' = (b_1', b_2', \ldots, b_{\ell'}')$. Recall that B^* is obtained by first maximizing the first element in the vector, therefore, $b_1^* \geq b_1'$ must hold. If $b_1^* > b_1'$, then $R(B^*) \geq R(B')$ by Lemma 5.3, and thus, B' cannot be a better vector than B^*. This means that $b_1^* = b_1'$ must hold.

We know that once b_1^* is determined, our algorithm continues by maximizing the next element of B^*. Hence, $b_2^* \geq b_2'$ must hold. If $b_2^* > b_2'$, then $R(B^*) \geq R(B')$ by Lemma 5.4, and thus, B' cannot be a better vector than B^*. This means that $b_2^* = b_2'$ must hold too.

By repeating this argument, finally, we arrive to the conclusion that $B^* = B'$ must hold, which is a contradiction. □

5.3 Analysis of the general case

THESIS 5.2. *I derive an approximation $\tilde{S}$ for the average anonymity set size of a randomly selected member when any number c of members are compromised as follows:*

$$\tilde{S} = \sum_{i=1}^{L} \sum_{k=1}^{b_i - 1} k N_i \binom{b_i - 1}{k - 1} (1 - q_i)^k q_i^{b_i - k} + 1 \cdot p + N \cdot (1 - p)^N$$

where L is the depth of the key-tree, $b_1, b_2, \ldots, b_L$ are the branching factors at the different levels in the tree, $N = b_1 \cdot b_2 \cdot \ldots \cdot b_L$ is the total number of leaves, $N_i = \frac{N}{b_1 \cdot b_2 \cdot \ldots \cdot b_i}$ is the number of leaves in a sub-tree below an edge at level i, $p = c/N$ is the probability of any member being compromised, and $q_i = 1 - (1 - p)^{N_i}$ is the probability of an edge at level i being compromised (meaning that at least one leaf in the sub-tree below that edge is compromised). I show, by means of simulations, that the above approximation formula is sharp. [C3]

So far, we have studied the case of a single compromised member. This was useful, because it allowed us to compare different key-trees and to derive a key-tree construction method. However, one may still be interested in what level of privacy is provided by a system in the general case when any number of members could be compromised. In this section, we address this problem.

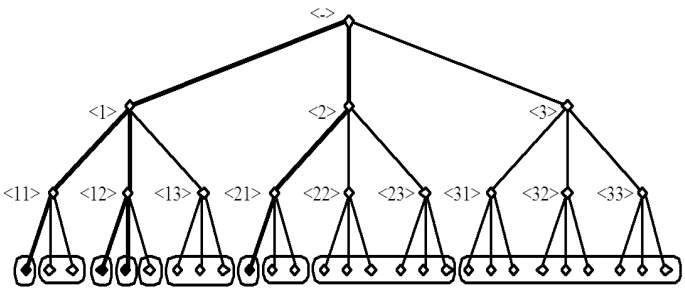


Figure 28: Illustration of what happens when several members are compromised. Just as in the case of a single compromised member, the members are partitioned into anonymity sets, but now the resulting partitions depend on the number of the compromised members, as well as on their positions in the tree. Nevertheless, the expected size of the anonymity set of a randomly selected member is still a good metric for the level of privacy provided by the system, although, in this general case, it is more difficult to compute.

We are interested in the expected anonymity set size $\bar{S}$ of a randomly selected member T in the general case when c randomly selected members are compromised in the tree. Instead of directly computing $\bar{S}$, we will estimate it by assuming that each member of the tree is compromised with probability $p = c/N$. Thus, the number of compromised members in the tree becomes a random variable with expected value c. Furthermore, without loss of generality, we assume that T is represented by the left most leaf of the tree.

Let us denote the branching factors of the tree by $b_1, b_2, \ldots, b_L$, where L is the depth (number of levels) of the tree. We say that an edge of the tree is compromised if there is a compromised leaf in the sub-tree below the given edge. The probability that an edge at level i of the tree is compromised is

$$q_i = 1 - (1-p)^{N_i}$$

where $N_i = \frac{N}{b_1 \cdot b_2 \cdot \ldots \cdot b_i}$ is the number of leaves in the sub-tree below the given edge.

The probability that the anonymity set size of the selected member T is exactly k, for $k = 1, 2, \ldots, b_L - 1$, is

$$(1-q_L)\binom{b_L-1}{k-1}(1-q_L)^{k-1}q_L^{b_L-k} \quad = \quad \binom{b_L-1}{k-1}(1-q_L)^{k}q_L^{b_L-k}$$

Note that if the anonymity set size of T is larger than or equal to b_L, then it can only be the multiple of b_L. Hence, the probability that the anonymity set size of T is not a multiple of b_L is 0, while the probability that it is equal to kb_L ($k = 1, 2, \ldots, b_{L-1} - 1$) is

$$\binom{b_{L-1}-1}{k-1}(1-q_{L-1})^{k}q_{L-1}^{b_{L-1}-k}$$

By the same argument, the probability that the anonymity set size of T is not a multiple of $b_L b_{L-1} \ldots b_{i+1}$ is 0, while the probability that it is equal to $kb_L b_{L-1} \ldots b_{i+1} = kN_i$ ($k = 1, 2, \ldots, b_i - 1$) is

$$\binom{b_i-1}{k-1}(1-q_i)^{k}q_i^{b_i-k}$$

From this, we get that the expected size of T's anonymity set is

$$\tilde{S} = \sum_{i=1}^{L}\sum_{k=1}^{b_i-1} kN_i\binom{b_i-1}{k-1}(1-q_i)^{k}q_i^{b_i-k} + 1 \cdot p + N \cdot (1-p)^N \tag{45}$$

where the term $1 \cdot p$ covers the case when T itself is compromised (this happens with probability p and in that case the anonymity set size is 1), and the term $N \cdot (1-p)^N$ covers the case when no member is compromised (this happens with probability $(1-p)^N$ and results in the anonymity set size of N).

In order to see how well $\tilde{S}$ estimates $\bar{S}$, we run some simulations. The simulation parameters were the following:

- total number of members $N = 27000$;
- upper bound on the maximum authentication delay $D_{max} = 90$;
- branching factor vector: $(30, 30, 30)$;
- we varied the number c of compromised members between 1 and 270 with a step size of one.

For each value of c, we run 100 simulations. In each simulation run, the c compromised members were chosen uniformly at random from the set of all members. We computed the exact value of the normalized expected anonymity set size $\bar{S}/N$ using the expression (40). Finally, we averaged the obtained values over all simulation runs. Moreover, for every c, we also computed the estimated value $\tilde{S}/N$ using the expression (45).

The simulation results are shown in Figure 29. The figure does not show the confidence interwalls, because they are very small (in the range of 10^{-4} for all simulations) and thus they could be hardly visible. As we can see, $\tilde{S}/N$ approximates $\bar{S}/N$ very well.

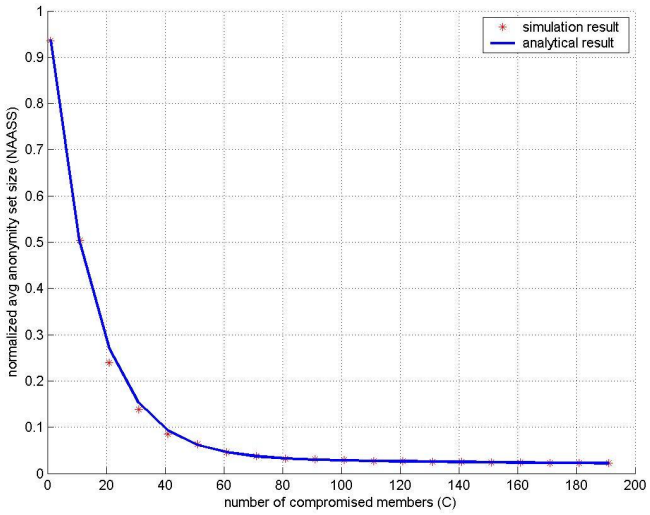


Figure 29: Simulation results for branching factor vector $(30, 30, 30)$. As we can see, $\tilde{S}/N$ approximates $\bar{S}/N$ very well.

In Figure 30, we plot $\tilde{S}/N$ for two trees that have the same number of branches at the first level, but they differ in the other branching factors. As we can see, the curves are almost perfectly overlapping. Thus, we can conclude that, just as in the case of a single compromised member, in the general case too, the level of privacy provided by the system essentially depends on the value of the first element of the branching factor vector.

Thus, a practical design principle for key-tree based private authentication systems is to maximize the branching factor at the first level of the key-tree. Further optimization by adjusting the branching factors of the lower levels may still be possible, but the gain is not significant; what really counts is the branching factor at the first level.

5.4 Summary

Key-trees provide an efficient solution for private authentication in the symmetric key setting. However, the level of privacy provided by key-tree based systems decreases considerably if some members are compromised. We showed that this loss of privacy can be minimized by the careful design of the tree. In particular, a good practical design principle is to maximize the branching factor at the first level of the tree such that the resulting tree still respects the constraint on the maximum authentication delay in the system. Once the branching factor at the first level is maximized, the tree can be further optimized by maximizing the branching factors at the successive levels, but the improvement achieved in this way is not really significant; what really counts is the branching factor at the first level.

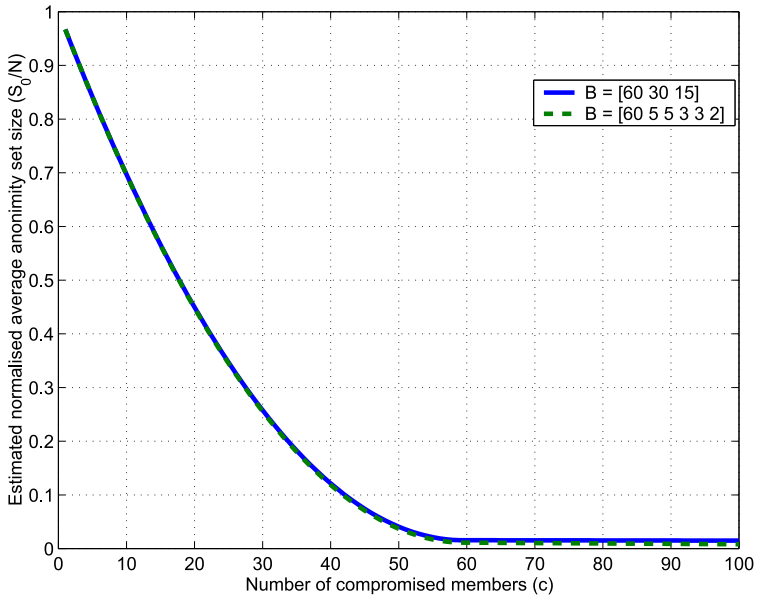


Figure 30: The value of $\tilde{S}/N$ as a function of c for different branching factor vectors that have the same first element. We can see that $\tilde{S}/N$ is almost the same for the two vectors. We can conclude that $\tilde{S}/N$ is essentially determined by the value of the first element of the branching factor vector.

References

[1] G. Ács, L. Buttyán, and I. Vajda. Provable security of on-demand distance vector routing in wireless ad hoc networks. In *Proceedings of the European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, Visegrad, Hungary, July 2005.

[2] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing (TMC)*, 5(11), November 2006.

[3] R. Ahlswede, N. Cai, S.-Y. Robert Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[4] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[5] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1998.

[6] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[7] I.N. Bronstein, K.A. Semendjajew, G. Musiol, and H. Muehlig. *Handbook of Mathematics*. Springer, 2004.

[8] S. Buchegger and J-Y. Le Boudec. Performance analysis of the CONFIDANT protocol. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2002.

[9] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[10] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad hoc WANs. In *Proceedings of the 1st ACM/IEEE International Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc)*, August 2000.

[11] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 8(5), October 2003.

[12] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2004.

[13] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[14] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Internet RFC 3626, October 2003.

[15] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Designing Privacy Enhancing Technologies*, pages 54–68. Springer LNCS 2482, 2002.

[16] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed data storage in sensor networks using decentralized erasure codes. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, November 2004.

[17] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In *Proceedings of the 4th IEEE Symposium on Information Processing in Sensor Networks (IPSN '05)*, 2005.

[18] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Transactions on Networking*, 14(SI):2809–2816, 2006.

[19] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed fountain codes for networked storage. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, 2006.

[20] C. Fragouli, J.-Y. Le Boudec, and J. Widmer. Network coding: an instant primer. *SIGCOMM Computing and Communications Review*, 36(1):63–68, 2006.

[21] T. Ho, R. Kötter, M. Medard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE Information Theory Symposium (ISIT)*, June 2003.

[22] T. Ho, B. Leong, R. Kötter, M. Medard, M. Effros, and D. Karger. Byzantine modification detection in multicast networks using randomized network coding. In *Proceedings of the IEEE Symposium on Information Theory (ISIT)*, June 2004.

[23] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2(3):28–39, May/June 2004.

[24] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the ACM Conference on Mobile Computing and Networking (Mobicom)*, 2002.

[25] ISO. Mechanisms using symmetric encipherment algorithms. ISO 9798-2.

[26] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielienski and H. Korth, editors, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[27] Z. Kfir and A. Wool. Picking virtual pockets using relay attacks on contactless smart card systems. In *Proceedings of the IEEE Conference on Security and Privacy in Communication Networks (SecureComm)*, 2005.

[28] M. N. Krohn, M. J. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 2004.

[29] S. Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[30] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, August 2000.

[31] P. Michiardi and R. Molva. Core: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. In *Proceedings of the International Conference on Communication and Multimedia Security*, September 2002.

[32] D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2004.

[33] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.

[34] C. Perkins, E. Belding-Royer, and S. Das. Ad-hoc on-demand distance vector (AODV) routing. Internet RFC 3561, July 2003.

[35] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 2000.

[36] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 2001.

[37] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the Privacy Enhancing Technologies (PET) Workshop*. Springer LNCS, 2002.

[38] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM'03*, March-April 2003.

[39] A. Urpi, M. Bonuccelli, and S. Giordano. Modeling cooperation in mobile ad hoc networks: A formal description of selfishness. In *Proceedings of WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, March 2003.

[40] S. Zhong, Y. R. Yang, and J. Chen. Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks. In *Proceedings of IEEE INFOCOM'03*, March-April 2003.

Publication of new results

International Journal Papers

[J1] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing (TMC)*, 5(11), November 2006.

[J2] L. Buttyán, L. Czap, and I. Vajda. Detection and recovery from pollution attacks in coding based distributed storage schemes. *IEEE Transactions on Dependable and Secure Computing*, 8(6), November/December 2011.

[J3] M. Félegyházi, J. Hubaux, and L. Buttyán. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5), May 2006.

International Conference and Workshop Papers

[C1] L. Buttyán, L. Czap, and I. Vajda. Securing coding based distributed storage in wireless sensor networks. In *Proceedings of the IEEE Workshop on Wireless and Sensor Network Security (WSNS)*, Atlanta, Georgia, USA, October 2008.

[C2] L. Buttyán, L. Dóra, and I. Vajda. Statistical wormhole detection in sensor networks. In *Proceedings of the European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*. Springer, July 2005.

[C3] L. Buttyán, T. Holczer, and I. Vajda. Optimal key-trees for tree-based private authentication. In *Proceedings of the International Workshop on Privacy Enhancing Technologies (PET 2006)*. Springer, June 2006.

[C4] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2004.

[C5] S. Capkun, L. Buttyán, and J. Hubaux. SECTOR: Secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2003.

[C6] M. Félegyházi, L. Buttyán, and J. Hubaux. Equilibrium analysis of packet forwarding strategies in wireless ad hoc networks – the static case. In *Proceedings of the International Conference on Personal Wireless Communication (PWC'03)*, September 2003.