ABSTRACT

Title of dissertation: COVERT CHANNELS
AND ANONYMOUS COMMUNICATION
IN AD HOC NETWORKS

Song Li
Doctor of Philosophy, 2007

Dissertation directed by: Professor Anthony Ephremides
Department of Electrical
and Computer Engineering

Ad-hoc wireless networks distinguish themselves from their traditional wired counterparts by three unique characteristics: mobility, lack of infrastructure, and shared wireless channel. These properties have gained popularity in various military and civilian applications, but have also introduced challenging problems in terms of ensuring satisfying network performance and network security. Ad hoc networks are a fertile ground for new threats and security problems. We start by demonstrating how new covert attacks can be launched by using the ad hoc network protocols. In particular, nodes in ad-hoc wireless networks have to cooperate with each other in order to accomplish many networking functions such as routing and channel access. We observe that covert information can be conveyed during the cooperation procedure. It is very difficult to eliminate or even detect these covert channels. Simulation results show that performance of these covert channels depends on various network characteristics. Anonymous communication has been considered as one possible way of fighting covert threats. In fact, anonymity and privacy by them-

selves have attracted intensive attention as important societal issues and desirable security features. One of the key components in most anonymous routing protocols is anonymous trapdoors, for which we propose a new construction scheme based on pairing-based cryptographies. More careful analysis has shown that anonymity could be in conflict with other secure properties and secure mechanisms, such as accountability and intrusion detection. We propose a solution that can flexibly trade off anonymity against accountability according to the needs of individual applications. The basic idea is to distribute the real identity of a given user among a set of pseudonyms in such a way that only a sufficient number of pseudonyms can lead to the recovery of the identity. Users authenticate each other anonymously under pseudonyms. When the number of times a user is caught misbehaving exceeds the threshold, the user's real identity can be recovered from the pseudonyms that had been used. Thus, accountability is enforced.

As conclusion, we propose to jointly investigate and incorporate all different secure properties by using various secure mechanisms across multiple protocol layers of the network.

COVERT CHANNELS AND ANONYMOUS COMMUNICATION IN AD HOC NETWORKS

by

Song Li

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor Anthony Ephremides, Chair/Advisor
Professor Alexandria Barg,
Professor Sennur Ulukus
Professor Richard La
Professor Michael Fu

Acknowledgements

I am deeply thankful to my advisor, Dr. Anthony Ephremides, for his mentoring and support throughout my doctoral studies. I truly appreciate his patience and tolerance during my journey. Dr. Ephremides, thank you! Your insight, knowledge, and enthusiasm inspired me to explore the world of engineering and science.

I would like to take this opportunity to thank my committee members Dr. Alexander Barg, Dr. Sennur Ulukus, Dr. Richard La, and Dr. Michael Fu, for your time and help on my research.

Finally, I would not be where I am now without the support of my parents and Ming, as well as the dear friends' encouragement. This dissertation is dedicated to them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ad-hoc wireless networks distinguish themselves from their traditional wired counterparts by three unique characteristics: mobility, lack of infrastructure, and shared wireless channel. These properties have gained popularity in various military and civilian applications, but have also introduced challenging problems in terms of ensuring satisfying network performance and network security. Ad hoc networks are a fertile ground for new threats and security problems.

In this dissertation, we focus on two security problems: *covert channels* and *anonymous communication*, both related to *hiding information* in wireless ad hoc networks.

1.1 Covert Channels in Ad Hoc Networks

Covert channels are concealed communication paths whose usage or even the very existence is not anticipated in the original design of a communication system[1]. Covert communication happens when one user intentionally manipulates and embeds information into some properties of the system in such a way that the extra information can be detected by specific designated users in the system.

Most of the past studies on covert channels have been concentrated on multi-level computer systems or wired computer networks. However, wireless commu-

nication networks involve a fundamentally new and ubiquitous environment with new and different variables that can be manipulated in order to convey covert information. The absence of fixed infrastructure and central control puts the burden of network organization and maintenance on the terminal nodes themselves. The decentralized style of operations empowers the nodes to manipulate the system individually. Since the nodes control their own behavior, they may manipulate the system parameters through legitimate operations in such a way that covert information can be embedded and conveyed. At the same time, the high degree of randomness in ad-hoc networks, due to factors such as node mobility, provides camouflage for the covert operations.

Anonymous communication was first suggested in [2] as one possible way of fighting covert threats. In fact, anonymity and privacy by themselves have attracted intensive attention as desirable security features.

1.2 Anonymous Communication in Ad Hoc Networks

Privacy has always been an important real world societal issue. User privacy in the cyberspace is needed for various reasons from enabling the e-commerce applications to supporting the freedom of speech ([3, 4, 5, 6]). Preserving privacy under the inherently open wireless communication networks ([7, 8, 9, 10, 11, 12, 13, 14]) has demonstrated even more challenging problems. The open wireless medium vulnerable to both eavesdropping and jamming has posed great challenges in protecting the communication between users.

In Chapter 3, we propose a new construction scheme of *anonymous trapdoors*. Anonymous trapdoor is utilized in most of the current wireless anonymous communication schemes ([11, 7, 13]) to hide the receiver identities. An anonymous trapdoor is a special token generated by a trapdoor function. The function is difficult to invert unless you are the designated receiver who has some secret information related to the trapdoor. In other words, only the designated receiver can open the trapdoor. Anonymity further requires that the other users not only cannot open the trapdoor, but neither can they recognize who has the capability.

1.3 Anonymous Authentication with Distributed Anonymity Revocation

Unfortunately, anonymity can be misused. A compromised user can abuse anonymity and launch malicious attacks without being detected. On the other hand, accountability ensures that events of interest can be linked to specific users such that responsibility can be assigned if something goes wrong. Without accountability, it would be impossible to know who caused the observed or suspected malfunction and what counteractions should be taken against whom in order to contain the damage.

Obviously, anonymity and accountability are two conflicting properties by definition. Existing anonymous communication schemes combat this problem by revoking anonymity when misbehavior is detected. Revocation of anonymity depends on the existence of centralized authority(CA), who maintains the mapping between user identity and user pseudonyms. However, in ad hoc networks, it cannot always

be assumed that such a centralized control is constantly available. It is crucial for the network users to be able to identify the misbehaving nodes by themselves and take proper counteractions. Current anonymous communication protocols for ad hoc networks, ANODR [11] and MASK [14] for example, either do not support anonymity revocation at all [11] or rely on some centralized control to do so [14].

We propose an anonymous authentication architecture with distributed anonymity revocation. The anonymity revocation protocol does not depend on the existence of any on-line trusted third parties. Instead, given enough information about a user's pseudonyms, anybody can revoke the anonymity of that user.

1.4 Outline of Thesis

This dissertation is structured as follows. It starts with Chapter 2 by demonstrating how new covert attacks can be launched through manipulating the ad hoc network protocols. We investigate the ad-hoc wireless networks' susceptibility to covert channels through the use of standard routing and MAC protocols. To support anonymous communication, a new construction scheme of anonymous trapdoors is introduced in Chapter 3 with detailed analysis of its properties. Based on the same cryptographic primitives used for the trapdoor, an anonymous authentication architecture with distributed anonymity revocation is introduced in Chapter 4. We discuss future work and conclude in Chapter 5.

Chapter 2

Covert Channels in Ad Hoc Networks

This chapter investigates ad-hoc wireless networks' susceptibility to covert channels that can be formed through manipulating the network protocols. It is very difficult to eliminate or even detect these covert channels. Simulation results show that performance of these covert channels depends on various network characteristics. Countermeasures against these covert channels are needed and also should adapt to the network changes to take full effect.

2.1 Motivation

Covert channels are concealed communication paths whose usage or even the very existence is not anticipated in the original design of a communication system[1]. Covert communication happens when one user intentionally manipulates and embeds information into some properties of the system in such a way that the extra information can be detected by specific designated users in the system. A covert channel, to be useful, does not need high bit rate or high capacity or even low loss rate. It is generally satisfactory if it can transmit a few bits per second with some positive probability. For example, only a few bits are needed to disclose the time of an attack or the PIN number of a personal bank account. However, a covert channel, to be effective, must be difficult to detect. This is a paramount requirement.

Existence of the covert channels violates both secrecy and integrity properties of trusted systems [15]. For example, in a multilevel trusted system, users with higher secret clearance are not allowed to send data to users with lower clearance. But a covert channel existing between the two different clearance levels can be used to communicate sensitive information. Under the "Trusted Computer System Evaluation Criteria (TCSEC)" of US Department of Defense [16], rigorous covert analysis is required for high assurance security systems.

Most of the past studies on covert channels have been concentrated on multilevel computer systems or wired computer networks. However, wireless communication networks involve a fundamentally new and ubiquitous environment with new and different variables that can be manipulated in order to convey covert information. The absence of fixed infrastructure and central control puts the burden of network organization and maintenance on the terminal nodes themselves. The decentralized style of operations empowers the nodes to manipulate the system individually. Since the nodes control their own behavior, they may manipulate the system parameters through legitimate operations in such a way that covert information can be embedded and conveyed. At the same time, the high degree of randomness in ad-hoc networks, due to factors such as node mobility, provides camouflage for the covert operations. It is hard to tell whether the nodes' activities are legitimate responses to the network changes or unexpected covert operations. The shared wireless medium facilitates the covert reception procedure further. For most cases, a covert receiver only needs to passively monitor the channel to obtain the covert information and hence is fully protected from detection.

This chapter demonstrates the ad-hoc wireless networks' susceptibility to covert channels through the use of standard routing and MAC protocols [17].

2.2 Covert Operations through the Use of Reactive Routing Protocols

One of the most basic problems in the operation of ad-hoc networks that has attracted a great deal of attention is to design route algorithms that can dynamically adapt to network topology changes and provide correct routes between communicating users in a timely and efficient manner. There have been many such algorithms developed over the past 10 to 15 years (e.g. see [18, 19, 20, 21, 22, 23, 24]). A class of routing protocols well-suited to ad-hoc networks is the so-called reactive protocols, like AODV [21] and DSR [22]. Both of these are rooted in the logic of the algorithm developed in [19, 20]. With that logic, in lieu of periodic flooding of routing information, reactive initiation of routes occurs according to need. A source sends route request messages only when the source needs to communicate with a destination provided that a valid route to the destination is not already available. The route request is broadcast in the network until it reaches the destination or some intermediate users who possess a route to the desired destination.

Covert channels can be built by taking advantage of the on-demand mechanism which allows the nodes to manipulate their own routing packets. In this section, I use the AODV protocol as an example. Four covert transmission mechanisms are presented that make use of different entries and properties of the routing packets.

Three of them have some limitations for their implementation. The fourth, however, seems to be perfectly implementable and is further evaluated through simulation. The first three will be briefly discussed to illustrate how covertness can be achieved, but will focus mostly on the fourth one. Performance of such covert channels depends on multiple factors, such as network size, user mobility, traffic rate, and transmission range. Simulation results show that the mobility of users, usually harmful to data network performance, turns out to be beneficial to the covert communication. Other on-demand routing protocols, including most of the secure routing protocols, share similar susceptibility properties to covert channel attacks. I focus on the AODV protocol since it is one of the most prominent protocols under consideration today, and not because I want to single it out in terms of vulnerability.

2.2.1 Overview of AODV

This section reviews those characterizations of AODV that are needed for the description of the covert attacks. For full details of the protocol, please refer to [21].

2.2.1.1 The On-demand Mechanism

The main idea of AODV is to avoid the large amount of periodic route control traffic by issuing route queries based on nodes' actual demands. In fact the root of this idea can be found in [19, 20]. When a source needs to communicate with some destination, for which a valid route is not already available, the source initiates a route discovery procedure by broadcasting a *route request* for the destination. The

route request propagates through the network until it reaches the destination node or some intermediate nodes that can provide the reply, provided the network is connected.

2.2.1.2 Sequence Number

The AODV protocol inherits the concept of destination *sequence number* from the destination sequenced distance vector (DSDV) routing algorithm [18] to ensure that the discovered routes are free from loops. Each node i in the network maintains a nondecreasing sequence number Seq_i. Node i may increment its sequence number under only two circumstances:

- Immediately before it initiates a route discovery;
- Immediately before it originates a route reply as a destination node in response to a route request by another node.

In the routing table, each route entry is associated with the last known destination's sequence number. When a route request is constructed, this last known destination sequence number is attached to the route request (an unknown sequence number flag is set if no sequence number is known). The sequence number of the source is also enclosed in the route request to facilitate establishing the reverse route. Let "Src" and "Dest" stand for source and destination. The route request (RREQ) and route reply (RREP) messages contain the following information respectively:

$< ID_{Dest}$, Seq_{Dest}, ID_{Src}, Seq_{Src}, Hop Count, $\ldots >$ and

$< ID_{Dest}$, Seq_{Dest}, ID_{Src}, Hop Count, lifetime, $\ldots >$.

The dots indicate additional fields that are not of interest here. When a node is presented with two valid routes to the same destination, it always chooses the fresher route, i.e. the one associated with larger destination sequence number. If the two routes are of the same freshness, the one with smaller hop count is chosen. The *Lifetime* value specifies the time after which the routing information is invalid. It is further explained next.

2.2.1.3 Route Table Management

In order to control the dissemination of stale route information, each valid route is also associated with a *Lifetime* value, which is the time after which the route is invalidated. The lifetime of a valid route is initially determined from the route control packets and later updated whenever the route is used to transmit data. An invalidated route is removed from the route table after a fixed period of time.

2.2.1.4 Route Maintenance

Movement of nodes causes link breaks and new node encounters. Once a neighbor is found to be unreachable, the node first invalidates all the routes that have that neighbor as their next hop. Then it forwards to all the affected upstream active neighbors a *route error* message, which contains those disconnected destination identities along with their last known sequence numbers. Receivers of the error message subsequently follow the same procedure and relay the error message to their upstream active neighbors. When the error message arrives at the source, the

source re-initiates the route discovery process if the route is still needed.

2.2.1.5 Expanding Ring Search Technique

The AODV protocol uses an additional feature, called *expanding ring search*, to control the broadcast of the route requests. The source node initially puts a *time to live* (TTL) value in the IP header of the route request and sets a timeout for receiving the route reply. The TTL value specifies how many hops the route request can travel away from the source. When the route request times out without receiving a reply, the route request is broadcast again with an incremented TTL value. This continues until the TTL value reaches a threshold. Later attempts eventually set their TTL values to be the network diameter so that they can traverse the entire network, if necessary.

2.2.2 Covert Channels in AODV

The AODV protocol provides routes between mobile nodes reactively by executing the route discovery process as outlined above. However, the route information carried in the route control packets can be used to convey additional information, which is independent of the original intention of the protocol design.

There are four covert channels immediately obvious in the use of AODV. Three of them have some limitations for their implementation. The fourth, however, seems to be perfectly implementable. The reason the first three are discussed is to show the rich variety of possibilities for covert transmission and to demonstrate the limitations

of some obvious covertness possibility in ad hoc networks.

2.2.2.1 Timing the Route Request

If a node can distinguish delays between successive route requests originated by a source, extra information can be deduced from the timing chosen by the source. It is quite common for covert channels to rely on timing information.

Implementing this channel requires synchronization between the source and the covert information receiver, which is not easily achieved in an ad-hoc wireless network. Plus, such a channel is seriously noisy since the source can never guarantee an exact time when its route request arrives at a particular node. In a multi-hop ad-hoc wireless network, the route requests can even arrive out of their transmission order. So, relying on the timing of route requests for covert transmission is problematic.

2.2.2.2 The Source Sequence Number in the Route Request

There are two different ways to convey covert information through manipulating the source sequence number field in the route requests.

The first one is to embed the covert information into the increments of the source sequence number between successive route requests. Before constructing a route request, the node increases its sequence number to a specific value such that the increment represents the covert symbol to be transmitted. However, this covert operation is easily detectable by the arbitrary size of increase in the node's sequence

number. Also, use of this covert mechanism may lead to the rapid exhaustion of the size of the sequence number field in the control packets.

The second choice is to embed the covert information into the increments of the source sequence number within a fixed period of time. A node controls the increment by constructing a number of route requests within this period. Similar to the timing channel described in Section 2.2.2.1, this covert channel requires synchronization between the covert transmitter and receiver, since they have to agree on the time to start counting. It also suffers from the field size exhaustion deficiency mentioned above. So, this covert channel choice is problematic as well.

2.2.2.3 The Lifetime Field in the Route Reply

When a node constructs route replies as an intermediate node, the lifetime entry in the route requests is computed from the corresponding lifetime entry in its routing table. The lifetime entry in the routing table indicates when is the last time that the route was used. Receivers of this route reply may derive extra information through looking into how recently the route was used by its constructor.

Exploiting this channel requires the covert transmitter to construct the route replies regularly which means the covert transmitter has to receive the corresponding route requests regularly. However, other nodes' demand for routes can not be directly controlled by the covert transmitter and receiver. Even if the other nodes do send the request, the reply is unicast back to the corresponding inquirer. So the probability that the covert receiver misses the reply is high. If the covert re-

ceiver sends these route requests itself, the frequent sending of the requests may cause exposure and discovery. Thus, this alternative for covertness has also serious drawbacks.

2.2.2.4 The Destination ID Field in the Route Request

Covert information also can be embedded in the destination identity of route request messages. For a network with N nodes, up to $log_2(N-1)$ bits of information can be deduced by noting which destination is requested at any given time.

This covert channel does not require synchronization between the covert transmitter and receiver. Order of reception is enabled through the source sequence number contained in the route request. Plus, the covert information is carried by the route requests which are broadcast in the network. So the probability of loss is expected to be lower than that of the previous methods. For these reasons, this covert channel is considered a fruitful option for covert communication and hence, it is examined in detail.

The following assumptions have been made:

1. The covert transmitter and receiver share an alphabet which can be composed of any node IDs except the covert transmitter's ID;
2. Covert symbol i is transmitted, if the covert transmitter originates a route request for destination i;
3. The covert transmitter controls its own demands for routes. Other than that, it complies with AODV;

4. Apart from the covert transmitter, all the other nodes issue route discoveries based on their actual demands.
5. The covert receiver passively monitors the channel and observes route requests issued by the covert transmitter, if these requests reach the covert receiver.

Assumption (3) allows the covert transmitter to generate route requests for any destination. However, the covert transmitter has to comply with the AODV protocol when sending the route requests. For example, a route request can be sent only when no valid route to the destination is already available, and the covert transmitter can not exceed the route request transmission rate specified by the AODV protocol. Otherwise, use of the covert channel is easily detectable by monitoring whether the AODV protocol is violated by any network users.

Covert transmission depends on the availability of the route to the intended node. When the route with the desired destination ID (as required by the covert message) is not already available at the covert transmitter, the covert transmitter can construct and broadcast a route request for the destination. The covert symbol is thus broadcast as part of the request. If a valid route to the intended node is already available, the covert transmission process is stuck. Notice that although the covert transmitter does not generate any RREQs when it is stuck, it still may construct RREPs as the destination in response to RREQs by other nodes. The covert transmitter increases its sequence number by 1 each time it constructs a new RREQ or a new RREP as the destination.

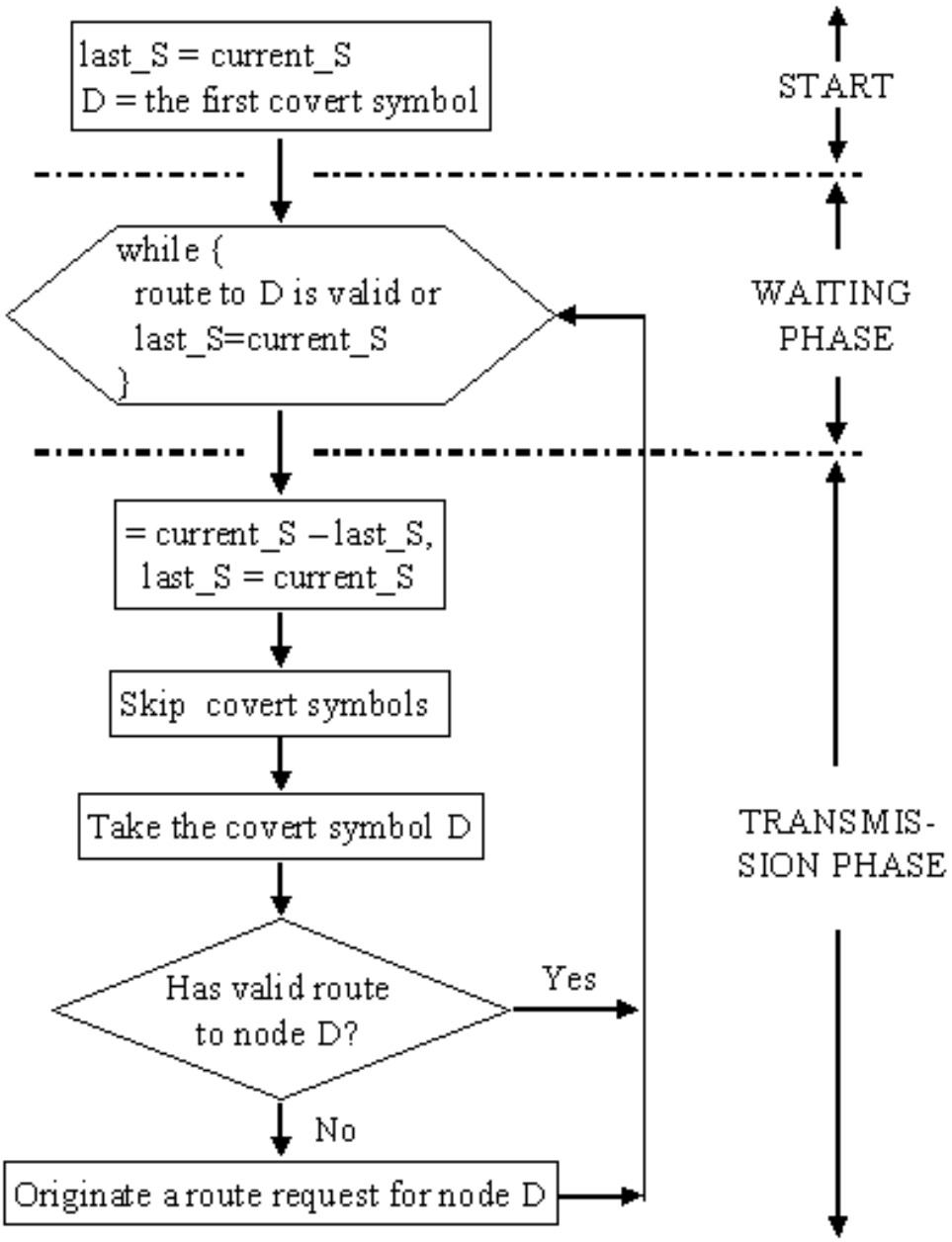


Figure 2.1: the covert transmission procedure through the use of AODV

Fig. 2.1 shows the covert transmission procedure as consisting of an initialization phase and two operational phases, separated by dash-dotted lines. In the figure, current_S stands for the sequence number of the covert transmitter, last_S is the sequence number of the covert transmitter at the most recent time the covert transmitter tried to make a covert transmission, and D is the covert symbol to transmit. The covert transmission procedure can be described as the following:

1. **Transmitting:** The covert transmitter reads in the next covert symbol to transmit. Assume it is D. The covert transmitter checks if it already has a valid route to the desired destination D. If not, the covert transmitter constructs and sends out a RREQ for destination D, and goes back to the beginning of this step to read the next covert symbol to transmit. If a valid route to the desired destination D is already available, the covert transmission is stuck. The covert transmitter enters the waiting phase.
2. **Waiting:** the covert transmitter stays in this phase until either the route to the destination D becomes invalid or it receives a RREQ from another node for which it is the requested destination. In the first case, the covert transmitter directly goes back to the beginning of step (1) to resume the covert transmission. In the second case, the covert transmitter first constructs and sends back the corresponding RREP, and then goes back to step (1). Either way, the covert symbol D that the covert transmission was stuck upon is skipped without ever been transmitted.

As it will be further explained next, in order to enable reordering at the covert

receiver's side, the covert transmitter's sequence number is used as index of the covert symbols. One sequence number identifies one covert symbol's position in the covert stream. In case that the covert transmitter's sequence number is increased by more than 1 between two consecutive covert transmissions, for instance by $\Delta + 1$, then Δ covert symbols in the covert stream have to be skipped correspondingly without being transmitted.

Covert reception is a passive procedure in which the covert receiver monitors the channel and observes route requests originated by the covert transmitter (as well as by other nodes). Although this covert channel is noise free, it is subject to symbol loss. First of all, the route requests are not guaranteed to be captured by the covert receiver for reasons such as intermediate nodes constructing the reply and stopping the forwarding of the route requests, or the expanding ring search technique controlling the dissemination of the route requests, or the covert transmitter and covert receiver simply not being connected. Additionally, the skipping of covert symbols during the transmission procedure is another cause of symbol loss.

So, each RREQ from the covert transmitter carries a covert symbol and the corresponding sequence number of the covert transmitter when the covert symbol was injected into the network. The covert receiver arranges the received covert symbols in the increasing order of the covert transmitter's sequence number. Any gap between two successive source sequence numbers implies loss of covert symbols.

In the next section, performance of this covert channel is evaluated under various network conditions through simulation. But before that, I would like to reemphasize here that it is not only AODV that is vulnerable to covert communi-

cations. Other on demand routing protocols have similar vulnerabilities, including most of the secure routing protocols. It is discussed in Section 2.2.5 the vulnerability of other reactive routing protocols to covert attacks.

2.2.3 Performance Evaluation

Performance of the covert channel is evaluated through simulation. A packet level discrete event simulator is developed to measure the covert channel performance under a variety of conditions. Although the simulation results are only isolated data points which can not describe the covert channel completely, the purpose is to demonstrate quantitatively some of the covert channel characteristics.

2.2.3.1 Simulation

The network is simulated as a group of nodes moving around in a $500 \times 500m^2$ square area according to the random way point mobility model [22]. Traffic streams are generated at each node according to independent Poisson processes with the same traffic rate in terms of packets per second. The destination is randomly selected among all the nodes in the network except the source itself. All packets are 64 bytes long.

Simplified assumption has been made about the MAC protocol that there is no MAC layer channel contention. Inclusion of a detailed MAC protocol would have a moderate effect on the performance of the covert channel, but it is not considered here. All the nodes have a common maximum transmission range. The

channel capacity is 2Mbits/sec. Transmission is successful if the receiver is within the maximum transmission range of a transmitter. Otherwise, packets are lost. Since the purpose is to evaluate the routing protocol's susceptibility to covert communications, these are reasonable MAC layer simplifications.

The simulated network contains N nodes. Each node has a unique ID that takes the values $0, 1, \ldots, N-1$. Node 0 is the covert transmitter and node 1 is the covert receiver. The covert transmitter and covert receiver share an alphabet composed of IDs of all the nodes except the covert transmitter, i.e. $1, 2, \ldots, N-1$. It is assumed that the input symbols are equally probable.

The AODV protocol parameters are chosen according to the default values given in [21], except that most of the simulation results are obtained without applying the expanding ring search technique. The purpose here is to study the fundamental potential of AODV for hosting covert communication.The expanding ring search technique is considered as a protocol parameter that may affect this potential. Its effect is studied by running a separate set of simulations with the expanding ring search technique applied.

The performance of interest includes transmission rate, channel throughput, probability of loss, and detectability of the covert communication. Denote $Symbol_{Transmitted}$ and $Symbol_{Received}$ to be the average number of covert symbols transmitted and received per second, respectively. The *transmission rate* T is defined as the average number of covert bits transmitted per second and the *covert throughput* R as the

average number of covert bits received per second, which are given by:

$$T = Symbol_{Transmitted} log_2(N-1) \quad (2.1)$$

$$R = Symbol_{Received} log_2(N-1). \quad (2.2)$$

The probability of loss P is calculated as

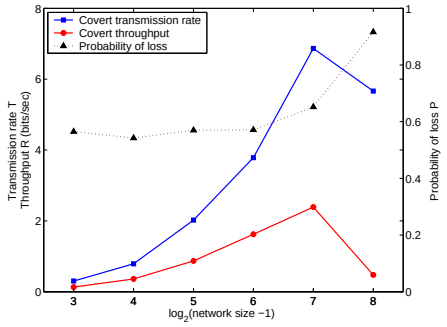
$$P = (T-R)/T \quad (2.3)$$

The covert communication is implemented without applying any physical layer channel coding/decoding algorithms. The throughput may increase if appropriate coding is applied.

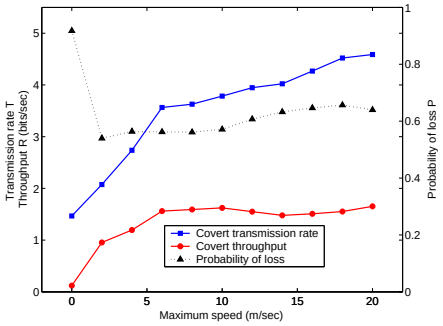
2.2.3.2 Simulation Results

Simulation results are obtained under multiple varying parameters, namely, the network size, the maximum speed and maximum pause time of the nodes, the traffic rate, and the transmission range. To study the effect of each of them, the simulation results are organized in groups. Within each group, there is only one varying parameter with the others fixed to their default values. The default values of those varying parameters are given in TABLE 2.1. For different network sizes, the maximum transmission range is calculated in such a way that there is on average π neighbors within a node's transmission range.

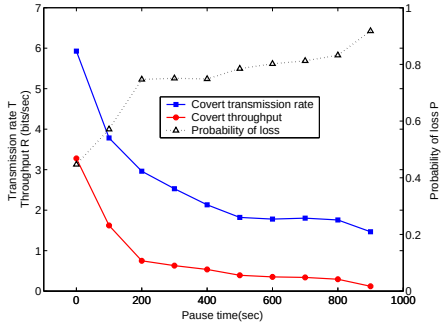
Each data point is the average of 10 independent runs. Each was executed for 900 seconds of simulation time, except that for the case of network size of 257, the simulation time was 125 seconds.



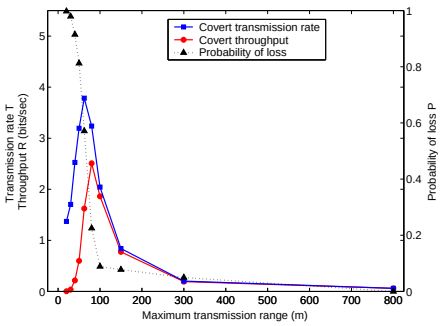
(a) Varying Network Size



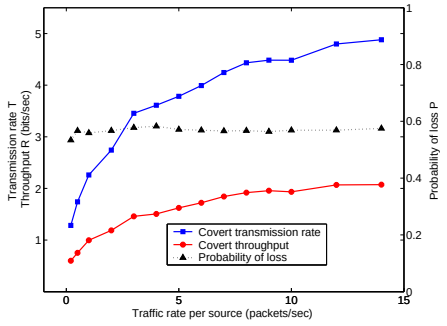
(b) Varying Maximum Speed



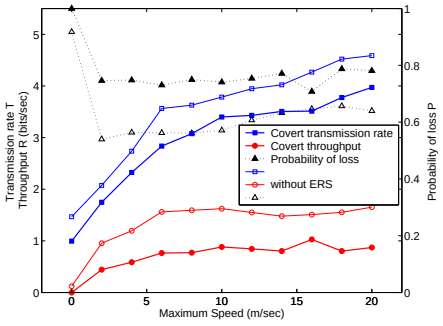
(c) Varying Pausetime



(d) Varying Transmission Range



(e) Varying Traffic Rate



(f) Expanding Ring Search

Figure 2.2: Covert Channel Performance in AODV

Table 2.1: Default Values of the Simulation Parameters

Parameters	$DefaultValues$
network size	65
pause time	100 *sec*
max speed	10 m/s
max transmission range	62.5 m
traffic rate per source	5 *packets/sec*

Fig. 2.2(a) presents the covert channel performance relative to the network size. Larger network size means larger input alphabet. The covert transmission rate and throughput increases as the network size grows. However, growth of the network also means larger distance between the covert transmitter and covert receiver, in terms of hop count. The route requests are more likely to be answered by intermediate nodes. The covert transmission gets less chance to be freed from stuck status. As a result, the probability of loss is greater and the channel throughput decreases.

Fig. 2.2(b) and 2.2(c) illustrates the effect of nodes mobility on the covert channel. Higher mobility of the nodes causes more link breakage, thus providing more excuses for covert transmission while fewer intermediate nodes are able to generate the route replies. The covert channel performs better as the network mobility increases.

Fig. 2.2(d) presents the covert channel performance in regard to the transmission range. When the transmission range is small, the network is so poorly connected that the route requests can not reach every node. The probability of loss is close to 1. Plus, if the covert transmitter can not receive a route request for itself, the covert

transmitter tends to stay stuck for a longer time. The transmission rate is low and the covert throughput is even lower. As the transmission range increases, the probability of loss decreases and transmission rate increases. But, when the transmission range becomes 800m, the network is fully connected. The covert transmitter has no excuses for covert transmission and the covert throughput drops back to zero

The presence of network traffic is a prerequisite for covert information transmission since it generates the need for routes, including the route to the covert transmitter. Note that when covert transmission is stuck, more RREQs for the covert transmitter means more opportunities to come out of the stuck state. So, the transmission rate increases as the traffic rate increases. This is reflected in Fig. 2.2(e).

All of the above results are obtained without applying the expanding ring search technique. In Fig. 2.2(f) these results are compared against the case of using the technique. The TTL start value, increment step, and threshold are equal to 1, 2, and 7 respectively, as suggested in [21]. Since the expanding ring search technique suppresses the distribution of the route requests, it increases the probability of loss. Also, the transmission rate is lower because the covert transmitter is less likely to receive route requests for itself. The covert channel throughput is reduced to about half of its original value.

2.2.4 Detectability

Consider the detection of covert communications as composed of three components: determination of whether there is a covert transmission going on, determination of the identity of the covert receiver, and determination of the identity of the covert transmitter. Our covert channel provides absolute protection on the covert receiver's identity and it is very difficult to decide whether the covert channel is actually in use.

As described in the previous parts of this chapter, the reception procedure of the covert operation is simply to passively monitor the route requests originated by the covert transmitter. So the covert receiver is fully protected from exposure.

To detect the covert transmissions and the covert transmitter ID, there are two ways: through observing some abnormality in the network or observing an abnormality in the behavior of the covert transmitter. Since the covert transmission affects only the covert transmitter's own routing and data transmission process, its effect on the whole network is expected to be rather unnoticeable. Finding out which node is the covert transmitter might be somewhat easier.

What the covert transmitter does for covert transmission is manipulating its demand for routes. It is important for the covert transmitter to justify its demands by actually using the routes. Otherwise, the destination may become suspicious when it notices that no traffic is ever received from the route requested. The covert transmitter has to use the routes to send some legitimate traffic. For example, the covert transmitter may send a data query if the destination hosts a database, or

click a couple of links if the destination hosts a website.

When a route is broken or when the covert transmitter increases its sequence number, the covert transmitter always checks for opportunities for covert transmission. One may observe that the covert transmitter constructs more route requests immediately after constructing a route reply as the destination. The covert transmitter can combat this kind of detection by delaying the route request for a random period of time, at the cost of smaller covert channel throughput and longer delay.

Also, without extra caution, the covert transmitter may issue route requests at a different rate from other nodes. When the network is stable, the covert transmission tends to get stuck, and the covert transmitter might issue less route requests than the other nodes. When the network is mobile, the covert transmitter has many excuses for covert transmissions and might issue more route requests than the other nodes. The covert transmitter can take counter actions against this type of detection by regulating its rate of issuing route requests and keeping it close to the rates of other nodes, possibly at the cost of decreasing covert channel throughput.

2.2.5 Covert Channels in Other Reactive Routing Protocols

As mentioned earlier, it is not only AODV that is vulnerable to covert communications. Other on demand routing protocols have similar vulnerabilities, including most of the secure routing protocols. In this section, several other reactive routing protocols are analyzed in terms of their vulnerability to covert attacks.

The Dynamic Source Routing (DSR [22]) protocol, in contrast to the AODV,

uses source routing instead of hop-by-hop routing. Each packet carries, in its header, the complete sequence of nodes that compose the route to the destination. To reduce the cost of route discovery, nodes cache the routes that they have learned or overheard from others. As demonstrated in [25], the aggressive caching mechanism used by DSR brings less routing overhead, i.e. lower number of routing packets, than that of AODV. Since in my scheme, the covert information is embedded in the routing packets, it is expected that the covert operation does not perform as well in DSR as in AODV.

The Zone Routing Protocol (ZRP [24]) is a hybrid routing protocol that executes reactively within a zone of certain radius from the source, and proactively for destinations that are not in the zone. Note that when the zone radius goes from one hop to infinite hops, the ZRP execution mode changes from pure reactive to pure proactive. The covert channel capacity would decrease then from some positive value to zero. The relationship between the zone radius and the covert channel capacity is an interesting problem.

Secure routing has received increasing attention. A group of secure routing protocols (Secure-AODV [26], Ariadne [27], SRP [28]) have been designed with the purpose of protecting the route discovery procedures from malicious node behavior. One general method used is to extend routing packets with digital signatures so that the integrity and authenticity properties of the routing information are, hopefully, guaranteed. But, the contents of the routing packets remain the same and are broadcast in the network unencrypted. The covert transmitter still has the freedom of embedding covert bits into its routing packets as described before. Thus, from

the point of view of immunity-against-covert-channel, these secure routing protocols are not better than their nonsecured versions. What is more, the secure routing protocols generally discourage route caching and intermediate nodes replying. This in turn encourages broadcasting of the route control packets and might result in better performance for the covert channel. What is even worse is that the cryptographic techniques that have been used may bring in even more covert channels. When digital signatures are used, covert ideas similar to those in [29] also apply here.

Other secure mechanisms have been proposed without involving cryptographic techniques in the route discovery procedures. Instead of protecting the routing packets, the ultimate goal is to guarantee correct data packet forwarding. As long as the route discovery procedures remain unchanged, the same covert channels can be embedded in these protocols. In addition, cryptographic tools may still be used for other purposes, such as to generate digital signatures [30]. The same ideas as in [29] may apply.

The ANODR (ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks [11]) protocol is designed to provide "untraceable" routes between a source and its destination. Identity of a particular transmitter is hidden from the other nodes. Our covert operations no longer work here, because the covert receiver can not even tell which packets are from the covert transmitter. So, ANODR is less vulnerable to covert operations. But such immunity comes at a cost. As shown in [11], ANODR performs worse than AODV in many aspects, including packet delivery ratio, end-to-end delay, and control overheads. However, it does offer immunity to the particular covert channel attack described here. The reduction in

performance is the price paid to acquire this immunity.

In this section, it is discussed in detail how covert operation can be achieved through the use of routing protocols. However, covert operation can also be implemented through the use of MAC protocols. In the next section, a class of MAC layer covert operations will be described. When the MAC layer covert channel operates conservatively, the covert operations can be absolutely undetectable. However, by nature of the MAC layer, these covert channels have the limitation that covert communication can only happen between nodes that are neighbors of each other.

2.3 Covert Operation through the Use of Splitting Algorithms

Nodes in ad-hoc networks have to share the wireless channel. Media Access Control (MAC) protocols are designed to address this issue. Depending on how the access to the channel is coordinated among active nodes, MAC protocols can be classified as contention-free where a dedicated server/node arranges the channel access among all the nodes in a centralized way, or contention-based where individual nodes make their own transmission decisions and resolve collisions by carefully choosing the retransmission time.

We focus on the contention-based class of MAC protocols since it offers individual nodes more powerful control over the system. A node can embed extra covert information into the system by controlling its own actions during the collision resolution procedure. It is presented in detail how covert operation can be implemented based on one specific class of collision resolution algorithms, referred

to as splitting algorithms [31]. We realize that splitting algorithms have not been used in the design of ad-hoc network MAC protocols, and in fact, they have not been implemented yet as MAC protocols in any real application. However, there is no serious reason why they may not be adopted in the future. In addition, the basic idea of splitting algorithms is simple and easy to explain. They offer the best platform to present the covert ideas described in this dissertation. These ideas can then be incorporated in other MAC protocols in some modified form.

2.3.1 Overview of the Splitting Algorithm

Assume the classical collision channel that is slotted with instant feedback of 'i(idle)', 's(success)', and 'c(collision)'. Collision happens when two or more nodes transmit in the same slot. The basic idea of a splitting algorithm is to divide collided nodes into smaller subsets, each of which then retransmits in turn. Successive collisions result in nodes splitting into smaller subsets, thus the probability of collision happening again is reduced. This procedure continues until all the collided packets are successfully transmitted, and this period is referred to as one collision resolution period (CRP). Such algorithms have been intensely studied in the past [32, 33, 34].

Splitting algorithms can differ from each other in several aspects, from the number of subsets they split into, to the handling of new arrivals during a CRP, etc. I start from the basic form, and introduce two of the modified versions later.

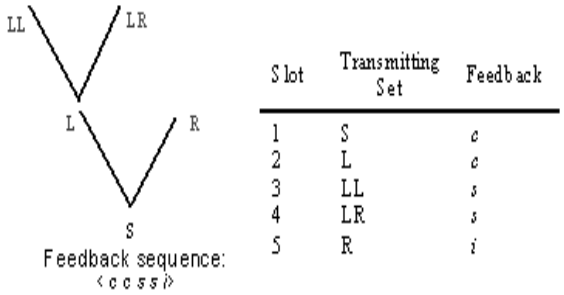


Figure 2.3: The Basic Binary Tree Algorithm

2.3.1.1 The Basic Binary Tree Algorithm

In the basic binary tree algorithm, after a collision, the competing nodes decide independently to join one of two subsets with equal probability. Transmissions among the two subsets are resolved in turn. Packets that arrived during the current CRP are blocked and wait for transmission until the new CRP starts.

This procedure can be better represented by a tree structure. Fig. 2.3 shows an example of a two-node collision. Denote the two subsets as left and right subset. After the first collision, the two nodes happened to join the same left subset. Collision happened again which means further splitting. This time, they joined different subsets. And two successes were observed. Coming back to the first right subset, since there is no remaining blocked node, the channel is idle.

One important feature of this algorithm is that any node in the system can keep track of and reconstruct the splitting tree by monitoring the channel feedback. Later, I will show that this is the essential property for the realization of the covert channels.

2.3.1.2 Improvement 1

This improvement consists of two parts. The first part is based on the observation that if an idle is observed in the first subset's transmission, it is guaranteed that the second subset is going to suffer a collision. So one time slot can be saved by splitting the second set before the actual collision occurs. Another observation is that if the first subset suffered a collision, then the second subset is expected to contain a small number of packets. This has motivated the second part of this improvement: instead of coming back to resolve the second subset, the algorithm can merge the second subset into the waiting group and work on it in the next CRP.

2.3.1.3 Improvement 2

In the basic binary tree algorithm, during a CRP, the new packet arrivals are blocked and get to be transmitted in the beginning of the next CRP. In the event that the previous CRP has taken a very long time, the number of waiting packets is expected to be large. They are going to continue to collide with each other before they are split into small enough subsets. One possible solution is to directly split this waiting set, i.e. root of the tree, into multiple j subsets. By estimating how many nodes are in the root set, the number j is chosen such that the expected number of packets per subset is slightly greater than one.

2.3.1.4 Unblocked algorithms

All the splitting algorithms described above require every node to monitor the channel feedback and to keep track of each CRP. This is undesirable when receivers are turned off, especially in wireless networks that strive to save battery power. One way to avoid this disadvantage is to transmit new packets immediately in the next slot after their arrival. This way, only currently transmitting nodes need to track the collision resolution procedure. Since the new arrivals are no longer blocked, this type of algorithms is called unblocked stack algorithms.

2.3.2 Covert Operations through the Use of Splitting Algorithms

Covert transmission can be realized via controlling the splitting procedure. Upon collision, the covert transmitter decides which subset to join according to the covert symbol it wished to transmit. For example, '1' is transmitted if it joins the left subset, and '0' is transmitted if it joins the right subset. In other words, the covert transmitter deviates from the rules followed by the other nodes but presents legitimate behavior that would correspond to the actual protocol rules. Its decisions, unknown to anyone else, are actually based on the covert symbol it wishes to transmit. Using the same example for the basic binary tree algorithm and assuming that one of the two collided nodes is a covert transmitter, Fig. 2.4 demonstrates how two covert bits "10" are transmitted in a CRP.

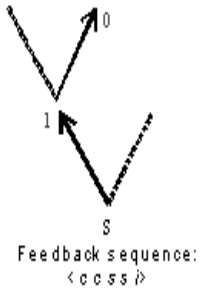


Figure 2.4: Covert Operation through MAC Protocol

To receive the covert information, the covert receiver needs only to passively monitor the channel feedback. It does not need to actively participate in the channel access competition. In the presented example, the covert receiver detects a successful transmission from the covert transmitter in the fourth slot. From the past channel feedback, the covert receiver can decide that the corresponding transmitting subset is "LR", thus concludes that covert bits '1' and '0' were transmitted. This is because any receiver can retrieve the "left"−"right" pattern in the transmission of any packet.

Given that the covert source has the same probabilistic distribution as the random splitting process, single use of this covert channel is impossible to detect, which makes this covert channel meet its first-priority requirement. Another obvious observation is that performance of this covert channel depends on how often the covert transmitter transmits and how often the covert transmitter meets collisions when transmitting. In fact, the covert transmitter can smoothly adapt its transmission rate and choose to run at higher rate at the risk of been detected through

the suspiciously atypical and aggressive transmission behavior. Three different operation modes are proposed that allow the channel undetectability to be traded off against throughput.

2.3.2.1 The Conservative Mode of Covert Operation

Assume that packet arrivals at each node are identically and independently distributed. The covert transmitter transmits only when it has a packet to send. The covert transmitter is different from the other nodes only in the way that it makes its splitting decisions according to the covert source requirement instead of the agreed-upon protocol method. Given that the covert source has the same probabilistic distribution as the protocol splitting process, use of this covert channel is absolutely undetectable (the used protocol rule is based on random decision or on time of arrival —as in the FCFS version [33]— which is also random).

2.3.2.2 The Aggressive Mode of Covert Operation

One limitation of the conservative mode is that occurrence of the covert transmission depends on actual packet arrivals. If no new packet has arrived at the covert transmitter before the start of a new CRP, the covert transmitter will not be able to do covert transmission in that CRP. The aggressive mode solves this problem by allowing the covert transmitter to generate new packets such that the covert transmitter can participate in each and every CRP. Under this mode, the covert transmitter transmits a packet in the first slot of every CRP.

Obviously, running in the aggressive mode exposes the covert transmitter in a way that may facilitate its detection. Note that the covert receiver always remains safe from detection.

2.3.2.3 The Strategic Mode of Covert Operation

The aggressive mode is mostly useful when the traffic is light. It is observed that the covert transmitter's effort is wasted if nobody collides with it. But the covert transmitter still suffers from the high risk of exposure by transmitting those packets. It would be more risk-throughput efficient if collisions can be guaranteed for the extra packets transmitted. A simple strategy can be taken by letting the covert transmitter jump into a CRP when it observes collision in the first slot of that CRP. The covert transmitter simply pretends that it is one of the originally collided nodes.

More complicated strategies can provide intermediate covert transmission rate by adapting the dummy packet generation rate according to the covert transmitter's eagerness to transmit and its willingness to get exposed.

2.3.3 Properties of the Covert Channel in Splitting Algorithms

In this section, I summarize some of the properties of the covert channel through the use of splitting algorithms. Performance of the covert channel is evaluated through simulation next in Section 2.3.5.

This covert channel is error free. Given correct channel feedback, the covert

receiver always can successfully track the CRP.

Throughput of the covert channel depends on multiple factors. First, upon each collision, the covert transmitter can embed at most $log_2 s$ covert bits in its split decision. The covert throughput is upper-bounded by $log_2 s$ bits per slot, where s is the number of subsets that the collided nodes are divided into. Second, the covert transmitter has to transmit data packets to convey its covert information. At most one packet can be transmitted in one CRP. Third, only when covert transmitter's data packet collides with other nodes' transmission can the covert transmitter embed one covert symbol into its splitting decision. The number of covert symbols sent in one CRP is equal to the number of collisions the covert transmitter meets before its successful transmission. So, the transmission rate also depends on the length of the CRP. Finally, not all of the slots are devoted to solving the collisions that involve the covert transmitter. For the rest of the time, the covert transmitter is either waiting for its turn to transmit, or waiting for the end of the current CRP. The covert throughput can be thus approximated by:

$$throughput \approx f_{CRP} \frac{c_{CRP}}{l_{CRP}} log_2(s) \tag{2.4}$$

where

- f_{CRP} is the frequency of the covert transmitter participating in CRPs;
- c_{CRP} is the number of collisions the covert transmitter encounters in a CRP;
- l_{CRP} is the length of the CRP;
- s is the number of subsets that collided nodes are divided into.

Based on (2.4), the following observations can be made. First, when the traffic

rate is low, f_{CRP} is small and the covert throughput is limited by covert transmitter's data packet transmission rate. Second, when the traffic rate is very high, almost all the users participate in each CRP. According to the results of Janssen and Jong in [34], for large number of users, m, (2.4) can be rewritten as (2.5) below, which indicates that the covert throughput is expected to decrease; namely, for high traffic rate and large number of users,

$$throughput \approx \frac{log_2(m-1)}{m} ln(s) \tag{2.5}$$

2.3.4 Detectability

Overall, it is very difficult to detect this covert channel. The covert receiver is guaranteed to be undetectable since it only passively monitors the channel to track the collision resolution procedure. The covert transmitter is also undetectable when it operates in the conservative mode. In fact, some splitting algorithms use different splitting criteria. The first-come-first-serve (FCFS) algorithm [33] uses the packet arrival times to decide which subset the packet should join in. Sagduyu and Ephremides [35] include the node residual battery energy into its decision factors to save the nodes' energy and lengthen the network lifetime. Still, none of this information is directly known to the other nodes except the owner itself. The transmission decision is made by a node locally. It is unclear what factors have influenced this decision. As a result, use of this covert channel is very difficult to detect especially when the covert source has similar distribution as the splitting decision does. When the covert transmitter runs under the aggressive or strategic mode, its ab-

normally active transmission could expose the use of the covert channel and its own identity. This trade-off can be made based on the particular circumstances of each application.

2.3.5 Performance Evaluation

To evaluate the performance of the covert channel, a packet-level discrete event simulator is developed. The covert channel performance is measured under a variety of conditions including variable number of nodes, traffic rate, covert operation modes, and various versions of the splitting algorithm. Again, the simulation results are only isolated data points which do not describe the covert channel completely; the purpose here is to demonstrate quantitatively some of the covert channel characteristics.

The collision resolution protocol is simulated with finite number of sources, denoted as N. Each source can have at most one packet waiting in the middle of a CRP and at most one other packet waiting to be transmitted in the next CRP. Packet arrivals at each user are i.i.d. Poisson processes, with mean λ/N packets per slot. The total average traffic rate is λ packets per slot. The performance metric of interest is covert throughput, which is defined as the average number of covert bits transmitted per slot.

The covert channel is implemented and evaluated based on the basic binary tree algorithm under all the three modes. Features of the first and second improvements are added into the binary tree algorithm separately. The unblocked

algorithm is implemented as in [32]. Only the conservative mode is implemented for the variations of the binary tree algorithm.

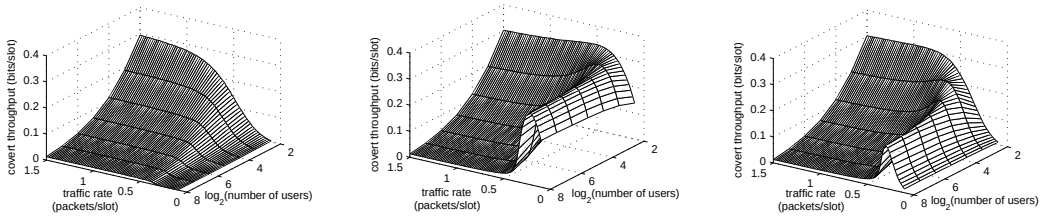
Extra care has to be taken to implement the covert channel in the unblocked type of algorithms, where there is no longer a clear definition of "start" and "end" of the collision resolution periods. To correctly track the splitting history of the covert transmitter, the covert receiver needs to know when the covert transmitter started to transmit the packet. A practical solution is to use a specific node's success as the synchronization signal. The covert transmitter always starts its transmission right after it observes a packet is successfully sent by that node. There is an advantage in using the covert receiver's own success, which allows the covert receiver to control the covert transmission rate by adjusting its own transmission rate. The covert receiver transmits valid packets upon their arrivals, and the covert transmitter joins the collision resolution procedure right after covert receiver's success. Dummy packets are created when necessary.

The simulation warms up with 20 CRPs. It lasts at least 1,000,000 slots and terminates at the end of the first CRP after the $1,000,000^{th}$ slot, except for the unblocked algorithm where the simulation warms up with 100 time slots and terminates at the $1,000,000^{th}$ slot.

Fig. 2.5 presents the covert channel performance in the binary tree algorithm under all three described operation modes. It is consistent with the observations made in Section 2.3.3). Light traffic implies rare collisions, and thus it holds back covert transmission rate, especially for the conservative mode in Fig. 2.5(a). On the other hand, with high traffic rate and large network, collisions happen frequently.

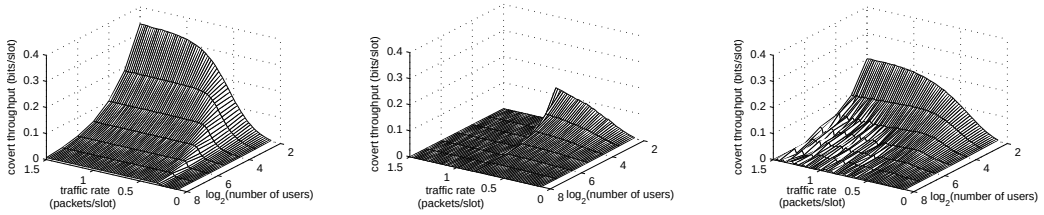
However, large portion of time is used to resolve the collisions that do not involve the covert transmitter. As a result, the covert throughput is not good either. The aggressive operation mode, shown in Fig. 2.5(b), effectively improves the throughput when the traffic is light. Meanwhile, the strategic mode does not exhibit as significant throughput improvement in Fig. 2.5(c). This is because in the aggressive mode, the covert transmitter not only makes use of every CRP with collisions, but also creates collisions through its aggressive transmission. Overall, the best throughput is obtained at high traffic rate and small number of users in the network. The best covert throughput is about 0.3 bits/slot.

Fig. 2.6 presents the covert channel performance under different variations of the splitting algorithms. The first improvement to the basic binary tree algorithm does not affect the splitting tree very much except that it reduces the CRP length by saving some slots of collisions and idleness . As a result, the covert throughput increases. The second improvement splits the root of tree depending on how many nodes are expected to be in the root. This improvement takes effect when the traffic rate is high and many new arrivals occur during the last CRP. By splitting the new arrivals immediately, extra collisions are avoided and covert transmission is held back. In Fig. 2.6(b), the covert throughput drops steeply as the traffic rate increases above a certain value. Fig. 2.6(c) illustrates the case of the unblocked algorithm. It shows similar features as it does with the blocked algorithm. But the covert throughput is not as good, especially at high traffic rate. This is the cost of synchronization between the covert transmitter and receiver. Our synchronization scheme requires the covert transmitter to wait until it observes a success from the



(a) Conservative Mode (b) Aggressive Mode (c) Strategic Mode

Figure 2.5: Covert Throughput under the Basic Binary Tree Algorithm



(a) Improvement 1 (b) Improvement 2 (c) Unblocked Algorithm

Figure 2.6: Covert Throughput under Various Splitting Algorithms

covert receiver before each transmission.

2.3.6 Covert Channels in Other MAC Protocols

The splitting algorithms resolve the collision by dividing collided nodes into smaller subsets. There are other contention based MAC protocols that resolve the collision using different methods [36, 37, 38, 39, 40]. In the basic ALOHA protocol [36], nodes can transmit anytime they want. If collision happens, they each wait a random waiting period before the next attempt. Slotted ALOHA [37] improves the channel efficiency by using slotted channel and nodes can access the channel only at the start of a slot. The CSMA (Carrier Sense Multiple Access) technique allows a node to avoid some of the collisions by sensing the channel for on-going transmission [38]. If the channel is already in use, the node will back off its transmission for a random period of time. The MACA (Medium Access Collision Avoidance) protocol [39] proposes a virtual sensing technology by using the RTS and CTS (Request-to-Send and Clear-to-Send) control packets. When the RTS packets from two or more nodes collide, each collided node adopts a random exponential back-off scheme. The IEEE 802.11 MAC protocol uses both physical sensing and virtual sensing technologies [40]. Upon collision, the random back-off time at each node is determined by the DCF (Distributed Coordination Function).

One common idea behind these protocols is that the retransmission decisions are made randomly by each individual node involved in the collision. Similar covert operations can be implemented by manipulating the retransmission decisions. Reception of the covert information is possible by observing the channel and the transmissions made by the covert transmitter. For example, covert information can be

conveyed through manipulating the channel idle period preceding the covert transmitter's transmission. The covert transmitter transmits a data packet only when the channel has been idle for a particular period of time which is determined by the covert symbol to transmit. Or, the covert information can be encoded in the number of collisions observed between two successful transmissions of the covert transmitter. The covert receiver in both examples can retrieve the covert information by passively monitoring the channel. Variations of these MAC protocols have been proposed to improve the back-off strategies. As long as the back-off scheme remains distributed and random, similar covert ideas may be applied in some modified form. The resulting covert channels may have different throughput and suffer different degrees of detectability.

2.4 Conclusion

It has been clearly demonstrated that covert communication can occur via controlling ad-hoc network protocols. Performance of the covert channels depends on various network parameters. Although the channel throughput is very poor comparing to normal data communication, use of these channels are very difficult to detect. Future investigation is necessary which includes, but is not limited to, a complete evaluation of the proposed covert channels, including theoretical analysis to decide the bounds on the covert channel throughput, and the design of countermeasures against the covert attacks. We believe there is rich potential for discovering and then exploring new covert channel attacks as well as defending against them.

Chapter 3

An Anonymous Trapdoor for Anonymous Communication in Ad Hoc Networks

3.1 Motivation

Privacy has always been an important real world societal issue. User privacy in the cyberspace is needed for various reasons from enabling the e-commerce applications to supporting the freedom of speech ([3, 4, 5, 6]). Preserving privacy under the inherently open wireless communication networks ([7, 8, 9, 10, 11, 12, 13, 14]) has demonstrated even more challenging problems. Wireless communication is vulnerable to both eavesdropping and jamming due to the open wireless medium.

Specifically, traffic pattern information and/or changes in traffic pattern information can be inferred by observing when and where a packet, encrypted or not, is transmitted between which users. Protection of such traffic information is critical for many sensitive applications. For example, in military applications, a sequence of packets transmitted by the commander may indicate a forthcoming action. Or, in a civil application, a psychiatric patient may not want to be noticed that he/she frequently visits some medical assistance webpage. Or, two collaborating companies may want to hide the fact that they are communicating. It is the objective of anonymous communication to conceal such traffic information from adversaries.

Referring to the terminology introduced by Pfitzmann and Kohntopp [41], there are three types of anonymous communication properties that can be provided: 1) Sender-anonymity, which means it is impossible to identify the sender of a particular message; 2) Receiver-anonymity, which similarly means it is impossible to identify the receiver; 3) Sender-receiver-anonymity, which means it is impossible to determine whether any two users in the network are communicating. For multi-hop ad hoc networks, sender and receiver also stand for source and destination. It is worth noting here that sender-receiver-anonymity is provided when either of the first two kinds of anonymity is assured.

None of the routing protocols mentioned in the previous chapter support anonymity. Routing information is carried in plain-text in both routing control packets and data packets, because intermediate nodes need to know who is the destination before they can decide where to forward the packets. An anonymous routing protocol is needed which can establish and maintain the connection between the source and destination without disclosing their identities to any other nodes, not even to those on the relaying path.

A novel concept of *anonymous trapdoor* has been utilized in most of the current wireless anonymous communication schemes ([11, 7, 13]) to hide the receiver identities. An anonymous trapdoor is a special token generated by a trapdoor function. The function is difficult to invert unless you are the designated receiver who has some secret information related to the trapdoor. In other words, only the designated receiver can open the trapdoor. Being anonymous further requires that the other users not only cannot open the trapdoor, but neither can they recognize who

does have the capability. By substituting the receiver ID in a packet with an anonymous trapdorr that only the receiver can open, a packet can be broadcast locally without specifying in clear-text who the receiver is. Similarly, the destination ID in the route request can be replaced with an anonymous trapdoor for the destination. Note that when a user receives a route request, it does not have to know exactly who is the requested destination, but only need to decide whether it is the destination itself. Any user except the destination will realize that it is not the destination when it fails to open the trapdoor, but cannot decide who is the destination. Such anonymous trapdoor can be implemented with cryptographic functions. The cost of constructing and opening the trapdoors depends on the type of cryptographic functions available at the application layer.

A novel trapdoor construction is proposed in this chapter. Construction of the trapdoor is based on a simple adaptation of the secret handshake scheme introduced in [42], which is reviewed next.

3.2 Cryptographic Primitives

This section introduces those cryptographic primitives utilized for construction of the proposed anonymous trapdoors. For more information, refer to [43] and references within.

3.2.1 Hash Function

A hash function H(x) transforms an input string x, which can have any length, to a fixed-size output string y. The output y is referred to as the *hash value* of x. The following properties are desired for a well designed cryptographic hash function: *1) Easy to compute. 2) One way.* Given an output y, it is computationally infeasible to find some input x such that H(x)=y. *3) Collision resistant.* For strong collision resistance, it is computationally infeasible to find any two inputs that have the same hash value. Weak collision resistance requires that given an input x_1, it is computationally infeasible to find another different input x_2 such that $H(x_1) = H(x_2)$. As implied in the collision resistance requirements, a hash function does not guarantee a one-to-one mapping between the input and output.

One major application of cryptographic hash functions is to generate message digests. The hash function transforms a long message of arbitrary length into a much shorter and fixed-length string, referred to as the *message digest*. Then expensive cryptographic operations can be performed on the short message digest rather than the original message to reduce computational cost. For example, computing the digital signature for a long message of arbitrary length is a very expensive operation. Instead of calculating the digital signature over the original message directly, it is much more efficient to hash the message first and then calculate the digital signature based on the short, fixed-length message digest.

3.2.2 Trapdoor Function

Similar to hash function, trapdoor function is also a one way function that is hard to invert unless (which makes it different from hash function) you have some special secret information. A trapdoor function does provide one-to-one mapping between the input and output space although inverting the mapping requires knowledge of the secret information.

An anonymous trapdoor further ensures that without that special secret information, not only you cannot open the trapdoor, but also you cannot tell who can. Based on this property, a packet can be protected with a trapdoor that only the intended receiver can open. Then the packet is broadcast to the network. While the protected packet reaches many users, only the intended receiver can open the trapdoor and retrieve the content of the packet. However, nobody can tell which user is the receiver.

3.2.3 Pairing Function

Our trapdoor construction is based on pairing functions, which is a bilinear, non-degenerate map $\hat{e} : G \times G \rightarrow G'$, where G and G' are two groups of large prime order q. The following properties are satisfied:

1. Bilinearity:

$$\forall P, Q \in G, \forall a, b \in Z_q^*$$

$$\hat{e}(aP, bQ) = \hat{e}^{ab}(P, Q)$$

2. Non-degeneracy: $P \neq 0 \Rightarrow \hat{e}(P, P) \neq 1$

3. Computability: $\hat{e}(P, Q)$ is efficiently computable.

In particular, modified Weil [44] pairing and Tate [45] pairing are two such bilinear maps, for which it is also assumed that the Bilinear Diffie-Hellman problem (BDHP) is hard, i.e., given P, aP, bP, cP in G, it is hard to compute $\hat{e}^{abc}(P, P)$.

Based on the bilinearity property and the BDHP assumption, pairing function has found increasing applications in cryptography, ranging from key agreement [46], to identity based encryption [44], to secret handshakes [42]. In this work, pairing function is used to implement anonymous trapdoors [47].

3.3 Pairing-based Trapdoor

3.3.1 The Bootstrapping Phase

Construction the anonymous trapdoor uses a bilinear, non-degenerate, and computable pairing function $\hat{e} : G \times G \rightarrow G'$, for which the BDHP is assumed to be hard. G and G' are two groups of large prime order q. Assume there are two collision resistance hash functions H_1 and H_2: $H_1 : \{0,1\}^* \rightarrow G$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^\beta$ (such as SHA-1). β is a fixed integer that represents the length of the output hash value. The 6-tuple $< q, G, G', \hat{e}, H_1, H_2 >$ is public known to every user.

The system administrator picks a random number $t \in Z_q^*$ and declares it to be the system secret. Then, the administrator equips each user i with a set of paired values: $\{< PsdNym_{i,j}, tH_1(PsdNym_{i,j}) >, for j = 1, 2, \ldots m\}$, where the

$PsdNym_{i,j}$'s are one-time-use pseudonyms of user i, and $tH_1(PsdNym_{i,j})$ is the secret point corresponding to $PsdNym_{i,j}$. The set is of size m. In addition to the pseudonyms, the administrator also equips each user i with a unique identity and the corresponding secret point: $< ID_i, tH_1(ID_i) >$.

Before entering the network, each user i is initiated with the following public information $< q, G, G', \hat{e}, H_1, H_2 >$, and the following private information $< ID_i$, $tH_1(ID_i) >$ and $\{< PsdNym_{i,j}, tH_1(PsdNym_{i,j}) >, for j = 1, 2, \ldots m\}$. However, user does not know the value of system secret t.

Denote $x_1 \parallel x_2$ as concatenation of string x_1 and x_2. Encryption of x using the key K is represented as $K(x)$, and decryption is represented as $K^{-1}(x)$. Construction of the proposed pairing-based trapdoor is described in the following section.

3.3.2 Pairing-based Trapdoor Construction

Now let us consider Alice as the source who wants to communicate with Bob. Alice needs to find a path to Bob and may have to rely on other users on the path to relay her packets. On one hand, Alice does not want them know who she is and who she is communicating with. On the other hand, the other users need to know where to forward Alice's packet. Alice establishes such a path by using a pseudonym for herself and hiding Bob's identity in the anonymous trapdoor.

Alice first picks an unused pseudonym $PsdNym_{Alice}$ and constructs the trap-

door as follows:

$$< PsdNym_{Alice}, CNym_{Alice,Bob}, K_{Alice,Bob}(ID_{Bob}) >, \tag{3.1}$$

where

$$\begin{cases} CNym_{Alice,Bob} &= H_2(\hat{e}(tH_1(PsdNym_{Alice}), H_1(ID_{Bob})) \parallel 0) \\ K_{Alice,Bob} &= H_2(\hat{e}(tH_1(PsdNym_{Alice}), H_1(ID_{Bob})) \parallel 1). \end{cases} \tag{3.2}$$

The second part of the trapdoor,$CNym_{Alice,Bob}$, is the pseudonym for the connection between Alice and Bob. The connection pseudonym and the secret key $K_{Alice,Bob}$ bind the source pseudonym with the destination ID through the pairing function. Such a binding can only be recognized by the destination. Both the connection pseudonym and the secret key have to be calculated with either Alice's secret point related to her pseudonym or Bob's secret point related to his identity. A third party cannot guess the value of $K_{Alice,Bob}$. In addition, although the connection pseudonym $CNym_{Alice,Bob}$ is public, nobody can tell that it is related to Bob's identity. For a user other than the source and destination, he can at most determine that the trapdoor cannot be opened by himself but cannot obtain any information about who can.

Recalling the on-demand routing mechanism described in chapter 2, upon receiving a route request, a user checks the destination ID entry. If the user is the destination or knows of a path to the destination, the user constructs and sends back reply. Otherwise, he rebroadcasts the request. Both of the destination ID and source ID are carried in plain-text. To anonymize the route request packet, these

entries can be replaced by an anonymous trapdoor like Eq. 3.1. Then the trapdoor-protected route request is broadcast throughout the network. An intermediate user who received the request cannot open the trapdoor, thus will continue to forward the route request. When the route request arrives at the destination Bob, Bob will be able to open the trapdoor and respond as the destination. Since intermediate users do not know who is inquired in the route request, they can no longer generate replies even if they do have routes to the destination. In the next subsection, the procedure of opening an trapdoor is described.

3.3.3 Open Trapdoor

When a user X receives a route request, it first checks whether it has seen the request before, for example by checking whether it has seen the source pseudonym before. If so, this request is ignored and discarded. Otherwise, user X takes as inputs the source pseudonym and its own secret point, and calculates:

$$CNym_{PsdNym_{Alice},X} = H_2(\hat{e}(H_1(PsdNym_{Alice}), tH_1(ID_X)) \parallel 0) \tag{3.3}$$

If user X is not the requested destination, the connection pseudonym calculated using X's secret point will not equal to the connection pseudonym carried in the route request:

$$\begin{aligned} CNym_{PsdNym_{Alice},X} &= H_2(\hat{e}(H_1(PsdNym_{Alice}), tH_1(ID_X)) \parallel 0) \\ &= H_2(\hat{e}(tH_1(PsdNym_{Alice}), H_1(ID_X)) \parallel 0) \\ &\neq \underbrace{H_2(\hat{e}(tH_1(PsdNym_{Alice}), H_1(ID_{Bob})) \parallel 0)}_{CNym_{Alice,Bob}}. \end{aligned} \tag{3.4}$$

When destination Bob receives the route request, he will try to open the trapdoor just like any other users:

$$\begin{aligned} CNym_{PsdNym_{Alice},Bob} &= H_2(\hat{e}(H_1(PsdNym_{Alice}), tH_1(ID_{Bpb})) \parallel 0) \\ &= \underbrace{H_2(\hat{e}(tH_1(PsdNym_{Alice}), H_1(ID_{Bob}))}_{CNym_{Alice,Bob}} \end{aligned} \tag{3.5}$$

Bob finds out that the connection pseudonym he calculates is equal to the one carried in the route request. Bob further confirms that he is the requested destination by calculating the secret key:

$$\begin{aligned} K_{PsdNym_{Alice},Bob} &= H_2(\hat{e}(H_1(PsdNym_{Alice}), tH_1(ID_{Bob})) \parallel 1) \\ &= \underbrace{H_2(\hat{e}(tH_1(PsdNym_{Alice}), tH_1(ID_{Bob})) \parallel 1)}_{K_{Alice,Bob}} \end{aligned} \tag{3.6}$$

Bob gets the same secret key that was used to encrypt the third part of the trapdoor. Bob can successfully decrypt it can find its own ID in the decrypted content. So Bob confirms that he is the destination and should send back a route reply. For any user X who is not Bob, given

$$< PsdNym_{Alice}, CNym_{Alice,Bob}, K_{Alice,Bob}(ID_{Bob}) >,$$

user X cannot learn any information about ID_{Bob} except that it is not his own identity ID_X.

3.3.4 Proof of Opening Trapdoor

When secure routing is desired, a proof of opening the trapdoor shall be provided in the route reply, such that the source can verify the route reply is indeed

created by the destination. A simple proof can be calculated as the follows:

$$Proof = H_2(\hat{e}(H_1(PsdNym_{Alice}), tH_1(ID_{Bob})) \parallel 2) \tag{3.7}$$

However, this proof can be verified by only the source. Forged route replies can still be injected into the network. Although the source will finally reject the reply, propagation of the fake reply still consumes communication and computation resources. It is more desirable if a proof can be verified by any intermediate user on the route.

A globally verifiable proof is provided by slightly modifying the trapdoor. In fact, another layer of hash is applied over $Nym_{Alice,Bob}$. The new trapdoor becomes

$$< PsdNym_{Alice}, CNym'_{Alice,Bob}, K_{Alice,Bob}(ID_{Bob}) >,$$

where $CNym'_{Alice,Bob} = H_2(CNym_{Alice,Bob})$. Accordingly, the same hash operation is performed when a user tries to decide if it is the destination. Because of the one-way property of the hash function, it is difficult to find another input that produces $CNym'_{Alice,Bob}$. However, the destination user with its own secret point can easily calculate $CNym_{Alice,Bob}$ just as the source did. So, the destination uses $CNym_{Alice,Bob}$ as its proof of opening the trapdoor, i.e. $Proof' = CNym_{Alice,Bob}$. Any forwarding user who has recorded the trapdoor can verify the proof by checking if $H_2(Proof') = CNym'_{Alice,Bob}$.

The proof of opening the trapdoor does not carry the destination identity in plain-text, so it reserves the property of destination anonymity.

3.4 Perfect Anonymity vs. Computational Anonymity

Recall that the anonymous trapdoor consists of the 3-tuple of <source pseudonym, connection pseudonym, encrypted destination identity>. The anonymity property requires that given a trapdoor, no third party beside the source and destination is able to tell which user's identity, out of a set of known user identities, was used to create the trapdoor.

Anonymity of a given trapdoor construction scheme can be assessed under an information-theoretical framework similar to the one proposed by Shannon in 1949 [48] to evaluate the *secrecy* of a cryptosystem. In Shannon's framework, the secrecy of a cryptosystem is evaluated as the amount of information about a randomly chosen message an attacker can derive from the cipher text. A cryptosystem is said to achieve *perfect secrecy* if the attack gains no information from the cipher text. Following the same framework, anonymity of a given trapdoor construction scheme can be assessed as the amount of information about a randomly chosen identity (drawn out of a set of known identities with some probability distribution) an attacker obtains after being given a trapdoor constructed using that identity. *Perfect anonymity* is achieved if the no information about the destination identity can be derived from the trapdoor.

However, perfect anonymity cannot be achieved using our trapdoor construction scheme. Theoretically, the trapdoor contains all the necessary information to calculate the destination identity. (For example, the destination identity can be found by performing a brute force search through all possible combinations of user

identities and integers $s \in Z_q^*$ and finding the pair which, combined with the source pseudonym, generates the same trapdoor.) A more practical goal is to make sure it is computationally infeasible to do so.

The same problem exists for the notion of perfect secrecy. In fact, achieving perfect secrecy under this information theoretical model has been proved impractical[48]. It requires that for every bit of information to be exchanged secretly, one bit of shared secret information is already established between the communication parties. No practical method has been found which can generate, exchange, and store such large amount of secret information. Insteads, security of a practical encryption scheme usually relies on the hardness of some well-known mathematical problems: for which no polynomial time solution is known (yet). By showing that the problem of breaking the cryptosystem can be reduced to solving the hard mathematical problem, it is proved to be *computationally infeasible* to recover the key and the original message.

3.5 Anonymous Authentication and Key Establishment

Two notable side benefits of our trapdoor construction are automatic anonymous authentication and key establishment. Alice and Bob agree on the same secret key $K_{Alice,Bob}$ (Bob calculates $K_{PsdNym_{Alice},Bob}$ which is equal to $K_{Alice,Bob}$) if and only if both Alice and Bob are legitimate users who have obtained their secret points from the system administrator. Intractability of the BDHP ensures that given a collection of pseudonyms and the corresponding secret points, the system secret t can not be deduced with non-negligible probability. Without knowing the system

secret t, it is hard to find pseudonym and the corresponding secret point that can the confirmation. In the mean time, Alice and Bob have established a shared secret key, $K_{Alice,Bob}$, for future transactions.

3.6 Related Work

3.6.1 Secret Handshakes

The proposed anonymous trapdoor is based on a simple adaptation of the secret handshake protocols [42]. The pairing function together with one-time user pseudonyms are used to perform secret handshakes. The secret handshake allows two users to authenticate each other as valid group members and set up a shared secret key after exchanging their pseudonyms. User pseudonyms are randomly generated independent of the user real identities and each pseudonym is used only once so that it is impossible to link a pseudonym to the user's real identity or to relate two pseudonyms to the same user. An authenticated and secure communication channel is established between two users without disclosing their real identities to anyone, not even to each other.

However, the secret handshake requires exchange of messages in both directions before the secret key can be calculated by both users. The two users have to exchange their pseudonyms before the shared secret key can be established. As a result, although the secret handshake scheme provides powerful anonymity properties, it cannot be applied directly for the problem of anonymous multi-hop routing. On one hand, the source needs to find a path and establish the connection to the

destination before any message can be exchanged. On the other hand, at least two, the source and destination pseudonyms, have to be exchanged before the connection can be established. This results in the chicken and egg problem. The proposed new anonymous trapdoor breaks this loop by replacing the destination pseudonym used during key calculation with its real identities so that messages only need to be passed in one direction: from the source to the destination, which is done through broadcast. The one-time source pseudonym is kept to protect both source anonymity.

Chapter 4

Anonymous Authentication with Distributed Anonymity Revocation

4.1 Motivation

Chapter 3 has described a new anonymous trapdoor construction scheme based on pairing functions. With the new trapdoor, it is possible to establish an anonymous communication connection between a pair of users in the wireless network.

Unfortunately, anonymity can be misused. A compromised user can abuse anonymity and launch malicious attacks without being detected. The same is true for a selfish user. Even if the misbehavior is detected, since it is conducted under pseudonyms, the attacker or selfish user can dodge detection and continue its action by simply switching to new pseudonyms. On the other hand, most misbehavior detection and responding schemes ([49, 50, 51, 52]) depend on accountability. Accountability ensures that events of interest can be connected to specific users such that responsibility can be assigned if something goes wrong. Without accountability, it would be impossible to know who caused the observed or suspected malfunction and what counteractions should be taken against whom in order to contain the damage.

Obviously, anonymity and accountability are two conflicting properties by definition. Existing anonymous communication schemes combat this problem by revoking anonymity when misbehavior is detected. Revocation of anonymity depends

on an assumed existence of centralized authorities(CA), who maintain the mapping between user identity and user pseudonyms. When evidence of misbehavior is presented, the real identity behind the pseudonyms is recovered and all the related pseudonyms are revoked. Availability of an on-line CA is critical for instant intruder identification.

However, in ad hoc networks, it cannot always be assumed that such a centralized control is constantly available. Users may obtain their anonymous credentials from the CA before joining the ad hoc network. It is inappropriate to assume that users can keep constant access to the CA. Current anonymous communication protocols for ad hoc networks, ANODR [11] and MASK [14] for example, either do not support anonymity revocation at all or rely on some centralized control to do so. For example, ANODR users generate their own uncertified pseudonyms which are absolutely unlinkable to their real identities. Better accountability is provided in MASK, as neighboring users authenticate each other under pseudonyms generated by a trusted authority. The trusted authority maintains a list of pseudonyms associated with each individual user. Later if a pseudonym is detected misbehaving, the trusted authority can link it to the particular user. The trusted authority revokes the anonymity of a user by broadcasting the complete list of its pseudonyms to the entire network. However, it cannot always be assumed that such a trusted authority is constantly available, in which case it is crucial for the network nodes to be able to identify the misbehaving nodes by themselves and take proper counteractions. Furthermore, maintaining the long list of revoked pseudonyms is expensive both communication-wise and storage-wise.

In this chapter, I describe an anonymous authentication architecture with distributed anonymity revocation. The anonymity revocation protocol does not depend on the existence of any on-line trusted third parties. Instead, given enough information about a user's pseudonyms, anybody can revoke the anonymity of that user. Assume that each user has a unique identity. Based on the cryptographic concept of *threshold secret sharing*, user pseudonyms are generated by decomposing the user identity using the threshold secret sharing scheme([53, 54]). Anybody collecting enough pseudonyms of a given user can recover that user's identity, and thus revoke the user's anonymity.

Once the identity of a user is recovered and distributed, all the pseudonyms of the user, used or unused, are automatically revoked due to the fact that they contribute to the same identity. This is an obvious advantage over the traditional solutions, where a long revocation list has to be maintained for all the arbitrarily-unlinkable pseudonyms.

This chapter is organized as follows. In Section 4.2, an existing anonymous authentication scheme, called secret handshakes, is introduced. The secret handshake scheme was already mentioned briefly in the previous chapter as related works. This section provides a more detailed description of the secret handshake protocol, based on which a new anonymous authentication protocol with distributed anonymity revocation is proposed. The basic idea behind the design is explained in Section 4.3. A new cryptographic primitive, the threshold secret sharing scheme, is also introduced in Section 4.3. It is described in detail in Section 4.4 the new anonymous

authentication with distributed anonymity revocation protocol. Its properties with regard to anonymity and accountability is analyzed in Section 4.5. Related work is reviewed in Section 4.6. Section 4.7 concludes this chapter with a summary and some pointers to future work.

4.2 Secret Handshakes

Our anonymous authentication is based on an existing anonymous authentication protocol, referred to as the secret handshake scheme [42]. Here, we briefly review the secret handshake scheme.

The secret handshake schemes in [42] are realized based on the same pairing-based cryptography that is already introduced in Chapter 3.2. The system is set up very similarly to that for the trapdoor constructions. The system administrator first generates a set of public parameters $< q, G, G', \hat{e}, H_1, H_2 >$, which is known by every user. The system administrator picks a random number $t \in Z_q^*$ and declares it to be the system secret. Then, the administrator equips each user i with a set of $\{< PsdNym_{i,j}, tH_1(PsdNym_{i,j}) >, for j = 1, 2, \ldots m\}$, where $PsdNym_{i,j}$'s are again one-time-use pseudonyms of user i, and $tH_1(PsdNym_{i,j})$ is the corresponding secret point. The difference is that for mutual anonymous authentication, users do not necessarily have to have the secret point related to their real identities.

Let Alice and Bob be two users who wish to authenticate each other. The secret handshake proceeds as follows. Alice and Bob each pick an unused pseudonymous credentials, denoted as $< PsdNym_{Alice}, tH_1(PsdNym_{Alice}) >$ and

$< PsdNym_{Bob}, tH_1(PsdNym_{Bob}) >$.

1. Alice sends Bob her pseudonym with a random nonce $nounce_{Alice}$.

$$Alice \xrightarrow{PsdNym_{Alice},\, nounce_{Alice}} Bob$$

2. Bob recieves Alice's pseudonym and calculates the secret value

$$S_{Bob} = \hat{e}(H_1(PsdNym_{Alice}), tH_1 PsdNym_{Bob}). \tag{4.1}$$

Bob picks another random nonce $nounce_{Bob}$ and calculates

$$V_0 = H_2(S_{Bob}||PsdNym_{Alice}||PsdNym_{Bob}||nounce_{Alice}||nounce_{Bob}||0). \tag{4.2}$$

Bob sends Alice his pseudonym, the random $nounce_{Bob}$, and the value V_0.

$$Alice \xleftarrow{PsdNym_{Bob},\, nounce_{Bob},\, V_0} Bob$$

3. Alice receives Bob's pseudonym and calculates

$$S_{Alice} = \hat{e}(tH_1 PsdNym_{Alice}, H_1(PsdNym_{Bob})). \tag{4.3}$$

Alice confirm that

$$V_0 = H_2(S_{Alice}||PsdNym_{Alice}||PsdNym_{Bob}||nounce_{Alice}||nounce_{Bob}||0). \tag{4.4}$$

Alice calculates

$$V_1 = H_2(S_{Alice}||PsdNym_{Alice}||PsdNym_{Bob}||nounce_{Alice}||nounce_{Bob}||1). \tag{4.5}$$

Alice sends Bob the value V_1.

$$Alice \xrightarrow{V_1} Bob$$

4. Bob confirms that

$$V_1 = H_2(S_{Bob}||PsdNym_{Alice}||PsdNym_{Bob}||nounce_{Alice}||nounce_{Bob}||1). \tag{4.6}$$

Notice that the secret value S_{Alice} is equal to S_{Bob}, if and only if both Alice and Bob are legitimate users who have obtained their secret points from the system administrator. Denote this common secret value as $S_{Alice,Bob}$. Various security properties of the secret handshakes have been proved in [42]. For more details, please refer to the original paper. Now, Alice and Bob can set up a shared secret key for future transactions as

$$K_{Alice,Bob} = H_2(S_{Alice,Bob}||PsdNym_{Alice}||PsdNym_{Bob}||nounce_{Alice}||nounce_{Bob}||2). \tag{4.7}$$

4.3 Basic Idea and the Threshold Secret Sharing

The secret handshake scheme described earlier allows two users to authenticate each other secretly. An eavesdropper observing the authentication process cannot learn anything, including the identities of the two parties. Many proposals have utilized this property of secret handshakes for anonymous communications, such as MASK [14]. The strong anonymity property was achieved through the use of arbitrarily-unlinkable pseudonyms during the secret handshakes.

Instead, I propose to construct user pseudonyms with certain embedded relationship. Given enough information, this relationship can be rediscovered and used to link all the related pseudonyms together. I will show that such pseudonyms can

be constructed using the threshold secret sharing scheme, which is introduced in the next section.

The new anonymous authentication architecture still allows users to authenticate each other under the pseudonyms. Upon joining the network, each user obtains a set of secret points corresponding to their pseudonyms. The secret points are calculated according to the secret handshake scheme[42]. Following the secret handshake procedure in [42], two users can successfully authenticate each other and establish a shared secret key, if and only if both of them have obtained their pseudonyms and the secret points from the system administrator.

The new anonymous authentication architecture provides strong anonymity against eavesdroppers. Two rounds of secret handshakes are performed during the authentication procedure. The second secret handshake is performed only if the first handshake is successful. Exchange of messages during the second secret handshake is protected with the key established during the first handshake. This ensures that anybody observing the authentication procedure can obtain only part of the user pseudonyms, which does not help to recover the user identity.

The new anonymous authentication architecture ensures that even if a user's identity is already discovered by the adversaries, its transactions with other benign users remain anonymous. In other words, although the user may be recognized when directly interacting with an adversary, its transactions with other benign users remain unrecognizable to the adversary.

We introduce the threshold secret sharing scheme next.

4.3.1 Threshold secret sharing

In order to be able to recover the user identity from its pseudonyms, there must be certain relationship embedded in these pseudonyms. Given enough information, this relationship can be rediscovered and used to link all the related pseudonyms together. Such pseudonyms can be constructed based on the threshold secret sharing scheme proposed by Shamir[53] in 1979. The original application was robust key management for crypto-systems. The basic idea is that given a piece of secret information, such as the user identity, construct n related pieces, generally referred to as secret shares, such that:

1. any k out of the n pieces will reveal the secret, but
2. any $k-1$ or fewer pieces are not enough for reconstructing the secret.

Shamir's scheme is perfect, in the sense that any fewer than the threshold number of shares reveals absolutely no information about the secret [54]. Now we review the scheme in more details.

Shamir's secret sharing scheme is based on interpolation of a polynomial defined over a finite field, $GF(p)$, where p is a large prime number. Given k points in the two dimensional plane, (x_i, y_i) for $i = 1, 2, ..., k$, there is a unique polynomial $F^{k-1}(x_i)$ of degree $k-1$ such that $F^{k-1}(x_i) = y_i$ for all i. Without loss of generality, we can assume that the secret D is an element in $GF(p)$. A (k, n) threshold secret sharing scheme works as follows:

1. pick a random $k-1^{st}$ degree polynomial $F^{k-1}(x)$:

$$F^{k-1}(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + ... + a_1x^1 + a_0 \tag{4.8}$$

where a_0 is the secret D. So, $F^{k-1}(0) = a_0 = D$.

2. the n shares $D_i = (x_i, y_i)$ are calculated by evaluating $F^{k-1}(x)$ at n distinct points x_i, $x_i \neq 0$:

$$D_i = (x_i, y_i) = (x_i, F^{k-1}(x_i)) \tag{4.9}$$

Given any subset of k of these D_i pairs, it is computationally easy to solve for $a_0 = D$ in Eq. (4.8). Knowledge of $k-1$ pairs, however, does not suffice in order to calculate D. In fact, no subset of fewer than k shares can determine any partial information about the secret.

One observation to be made here is that a secret share is actually composed of a pair of values, i.e. the input value x and the polynomial evaluated at x . Without the second part, the value of x alone does not carry any information about the secret. With some careful design, this fact is utilized in our authentication protocol to ensure anonymity. More details will be provided with the formal introduction of the authentication protocol in section 4.4.

4.4 The AADAR Protocol

4.4.1 Pseudonym generation

In the bootstrapping phase, the SA first determines a pair of public and private keys $< PubK, PrvK >$, two groups G and G' of the same prime order q, a Weil/Tate pairing mapping function $\hat{e}$, and two collision resistant hash functions $H_1 : \{0,1\}^* \rightarrow G$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{\beta}$. Then, SA picks a system secret t. In the end, each node has the knowledge of $< PubK, q, G, G', \hat{e}, H_1, H_2 >$, but does not know the

value of the private key $PrvK$ or the system secret t.

For each user, say Alice as an example, the SA generates a certificate for her identity ID_{Alice} : $cert(ID_{Alice}) = \{ID_{Alice}\}_{PrvK}$, where $\{ID_{Alice}\}_{PrvK}$ denotes ID_{Alice} signed by key $PrvK$. Then, the TA picks a random polynomial of order $k-1$: $F^{k-1}_{Alice}(x) = a_{Alice,k-1}x^{k-1} + a_{Alice,k-2}x^{k-2} + ... + a_{Alice,1}x + a_{Alice,0}$ and $a_{Alice,0} = cert(ID_{Alice})$.

A set of pseudonymous credentials of the form $\{< PsdNym^{(1)}_{Alice,i}, ScrPt^{(1)}_{Alice,i}, PsdNym^{(2)}_{Alice,i}, ScrPt^{(2)}_{Alice,i} >, \ldots\}$ are constructed as follows:

- $PsdNym^{(1)}_{Alice,i}$'s are distinctive non-zero random numbers in G;
- $ScrPt^{(1)}_{Alice,i} = tH_1(PsdNym^{(1)}_{Alice,i})$ is the secret point with regard to $PsdNym^{(1)}_{Alice,i}$;
- $PsdNym^{(2)}_{Alice,i} = F^{k-1}_{Alice}(PsdNym^{(1)}_{Alice,i})$ is the polynomial evaluated at $PsdNym^{(1)}_{Alice,i}$;
- $ScrPt^{(2)}_{Alice,i} = tH_1(PsdNym^{(1)}_{Alice,i}||PsdNym^{(2)}_{Alice,i})$ is the secret point corresponding to the concatenation of $PsdNym^{(1)}_{Alice,i}$ and $PsdNym^{(2)}_{Alice,i}$.

Alice presents only $PsdNym^{(1)}_{Alice,i}$ during the first secret handshake. If and only if the first secret handshake was successful, Alice transmits $PsdNym^{(2)}_{Alice,i}$ encrypted with the key established during the first handshake. The binding between $PsdNym^{(2)}_{Alice,i}$ and $PsdNym^{(1)}_{Alice,i}$ is verified if the second secret handshake is successful. The random value $PsdNym^{(1)}_{*,*}$ of two users shall not collide. Otherwise,

anonymity may be tampered since two pseudonyms with the same $PsdNym^{(1)}_{*,*}$ but different $PsdNym^{(2)}_{*,*}$ obviously belong to different users. In the next subsection, it is described how anonymous authentication is realized under the pseudonyms.

4.4.2 Anonymous Authentication

The anonymous authentication protocol consists of two rounds of secret handshakes. Two users authenticate each other by exchanging pseudonyms. It is important to have two rounds of secret handshakes in order to ensure anonymity. Notice that anybody having enough shares of the pseudonyms can recover the user identity. Exchange of the pseudonyms must be protected from arbitrary eavesdroppers. With the two-round mechanism, only part of the pseudonym is subject to eavesdropping during the first handshake. Once two users establish the shared secret key, exchange of the remaining part of the pseudonym can be protected by encryption. Assume Alice and Bob are the two users who want to authenticate each other. Alice and Bob each pulls out one unused pseudonym, say
$< PsdNym^{(1)}_{Alice,i}, ScrPt^{(1)}_{Alice,i}, PsdNym^{(2)}_{Alice,i}, ScrPt^{(2)}_{Alice,i} >$ and
$< PsdNym^{(1)}_{Bob,j}, ScrPt^{(1)}_{Bob,j}, PsdNym^{(2)}_{Bob,j}, ScrPt^{(2)}_{Bob,j} >$.
The superscripts '(1)' and '(2)' are used to distinguish messages exchanged in the first and second secret handshakes.

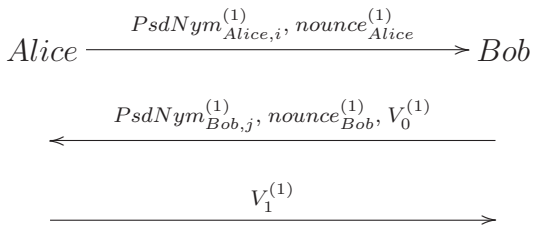


Figure 4.1: AADAR: The First Round of Secret Handshake

The first secret handshake involves only $< PsdNym^{(1)}_{Alice,i}, ScrPt^{(1)}_{Alice,i} >$ and $< PsdNym^{(1)}_{Bob,j}, ScrPt^{(1)}_{Bob,j} >$ and follows the same procedure as the original scheme of [42]. As demonstrated in Fig. 4.1, the random nonces $nounce^{(1)}_{Alice}$ and $nounce^{(1)}_{Bob}$ are picked by Alice and Bob individually. $V^{(1)}_0$ and $V^{(1)}_1$ are values calculated by Bob and Alice according to their secret points, each to be verified by the other party. During the handshake procedure, Alice and Bob calculate a shared secret value:

$$\begin{aligned} S^{(1)}_{Alice,Bob} &= \hat{e}(ScrPt^{(1)}_{Alice,i}, H_1(PsdNym^{(1)}_{Bob,j})) \\ &= \hat{e}(H_1(PsdNym^{(1)}_{Alice,i}), ScrPt^{(1)}_{Bob,j}). \end{aligned} \tag{4.10}$$

Then, $V^{(1)}_0$ and $V^{(1)}_1$ are calculated and confirmed as:

$$\begin{cases} V^{(1)}_0 = H_2(S^{(1)}_{Alice,Bob}||PsdNym^{(1)}_{Alice,i}||PsdNym^{(1)}_{Bob,j}||nounce^{(1)}_{Alice}||nounce^{(1)}_{Bob}||0) \\ V^{(1)}_1 = H_2(S^{(1)}_{Alice,Bob}||PsdNym^{(1)}_{Alice,i}||PsdNym^{(1)}_{Bob,j}||nounce^{(1)}_{Alice}||nounce^{(1)}_{Bob}||1). \end{cases} \tag{4.11}$$

Based on $S^{(1)}_{Alice,Bob}$, Alice and Bob also calculate a shared secret key as:

$$K^{(1)}_{Alice,Bob} = H_2(S^{(1)}_{Alice,Bob}||PsdNym^{(1)}_{Alice,i}||PsdNym^{(1)}_{Bob,j}||nounce^{(1)}_{Alice}||nounce^{(1)}_{Bob}||2) \tag{4.12}$$

Once $V_0^{(1)}$ and $V_1^{(1)}$ are successfully confirmed, Alice and Bob know that they possess the same secret key $K_{Alice,Bob}^{(1)}$. This key is used during the second secret handshake to protect the pseudonyms from eavesdroppers.

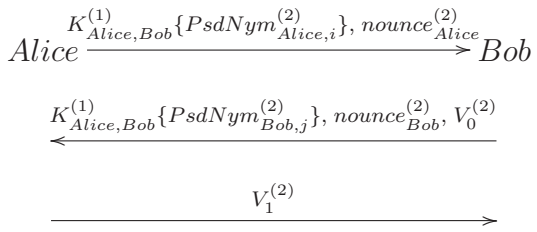


Figure 4.2: AADAR: The Second Round of Secret Handshake

The second handshake proceeds as demonstrated in Fig. 4.2 $K_{Alice,Bob}^{(1)}\{\cdot\}$ denotes $\cdot$ encrypted with the shared secret key $K_{Alice,Bob}^{(1)}$. $nounce_{Alice}^{(2)}$, $nounce_{Bob}^{(2)}$, $V_0^{(2)}$, and $V_1^{(2)}$ have similar meaning as above. Another shared secret value $S_{Alice,Bob}^{(2)}$ can be calculated by Alice and Bob as:

$$\begin{aligned} S_{Alice,Bob}^{(2)} &= \hat{e}(ScrPt_{Alice,i}^{(2)}, H_1(PsdNym_{Bob,j}^{(1)}||PsdNym_{Bob,j}^{(2)})) \\ &= \hat{e}(H_1(PsdNym_{Alice,i}^{(1)}||PsdNym_{Alice,i}^{(2)}), ScrPt_{Bob,j}^{(2)}) \end{aligned} \quad (4.13)$$

Similarly, $V_0^{(2)}$ and $V_1^{(2)}$ are calculated and confirmed as:

$$\begin{cases} V_0^{(2)} = H_2(S_{Alice,Bob}^{(2)}||PsdNym_{Alice,i}^{(2)}||PsdNym_{Bob,j}^{(2)}||nounce_{Alice}^{(2)}||nounce_{Bob}^{(2)}||0) \\ V_1^{(2)} = H_2(S_{Alice,Bob}^{(2)}||PsdNym_{Alice,i}^{(2)}||PsdNym_{Bob,j}^{(2)}||nounce_{Alice}^{(2)}||nounce_{Bob}^{(2)}||1) \end{cases} \quad (4.14)$$

Validity of the whole pseudonyms is confirmed if and only if both $V_0^{(2)}$ and $V_1(2)$ are successfully verified. Just like the first secret handshake, Alice and Bob

can calculate another shared key based on $S^{(2)}_{Alice,Bob}$:

$$K^{(2)}_{Alice,Bob} = H_2(S^{(2)}_{Alice,Bob}||PsdNym^{(2)}_{Alice,i}||PsdNym^{(2)}_{Bob,j}||nounce^{(2)}_{Alice}||nounce^{(2)}_{Bob}||2) \tag{4.15}$$

4.4.3 Anonymity Revocation

As discussed earlier, anonymity can be misused. Some traditional security solutions, such as ([49, 50, 51, 52]), no longer work when anonymity is enforced. This is because these solutions rely on accountability, which is incompatible with anonymity by definition. To combat this problem, we propose a scheme that offers distributed anonymity revocation when misbehavior is detected. In our scheme, identity of a misbehaving node can be recovered once it is caught misbehaving more than a certain number of times (in this case, the secret sharing threshold k).

In our scheme, each user in the system maintains two *blacklists*: one list of misbehaving pseudonyms and another list of revoked identities. Each entry in the list of revoked identities is also associated with the corresponding polynomial $F^{(k-1)}(\cdot)$ used to decomposed that identity. Without loss of generality, assume Alice detects that Bob is misbehaving. Alice has one of Bob's pseudonyms:$<$ $PsdNym^{(1)}_{Bob,j}, PsdNym^{(2)}_{Bob,j}$ $>$, which is also a share of Bob's real identity. Alice adds this share to the list of misbehaving pseudonyms she has maintained. When this list is of size at least k, Alice can try to recover the real identity of misbehaving nodes by running the polynomial interpolation with every combination of k pseudonyms. If the reconstructed secret does not form a valid certificate, i.e. not in

the form of $cert(ID_X)$, Alice decides that these k pseudonyms do not belong to the same user. Otherwise, Alice determines that a misbehaving user X with identity ID_X is recovered. The corresponding polynomial $F_X^{k-1}(\cdot)$ is also reconstructed from the pseudonyms. Alice adds the ID_X and the polynomial $F_X^{k-1}(\cdot)$ into the list of revoked identities.

Denote the size of the list of misbehaving pseudonyms as z. For $z \geq k$, Alice has to run $\binom{z}{k}$ number of polynomial interpolations. Obviously, as z grows, the number of polynomial interpolations to be performed is $O(z^k)$. However, we expect z to be bounded by $(k-1) \cdot N_{adversary} + N_{innocent}$, where $N_{adversary}$ is the number of adversaries in the network and $N_{innocent}$ is the number of innocent pseudonyms mistakenly observed as misbehaving. We also observe that when one identity is recovered, all the related pseudonyms can be removed from the list. Thus the list of misbehaving pseudonyms contains only the suspected pseudonyms of those users whose identity has not been recovered yet.

To prevent the identified adversaries from further participating in the network operation, Alice has to verify that a pseudonym $< PsdNym^{(1)}, PsdNym^{(2)} >$ does not belong to one of the identified adversaries. This can be realized by ensuring:

1. $< PsdNym^{(1)}, PsdNym^{(2)} >$ is not already in the list of misbehaving pseudonyms
2. $\forall ID$ in the list of identified adversaries,

$$PsdNym^{(2)} \neq F_{ID}^{k-1}(PsdNym^{(1)}). \tag{4.16}$$

Our anonymity revocation scheme ensures that once the identity of a node is recovered, all of its pseudonyms, used or unused, can be linked together. This offers

an efficient system for anonymity revocation in comparison with other solutions where a long revocation list has to be maintained for all the arbitrarily-unlinkable pseudonyms.

4.4.4 Blacklist Exchange

Ad-hoc network users are mobile users. So are the adversaries. A single honest user may have not collected enough pseudonyms of a given adversary before it moves away. A smart adversary may dodge identification by carefully "distributing" its attacks across the whole network, while never leaving enough evidence to one single honest user.

A smart adversary can also prevent identification by reusing its pseudonyms to limit the disclosure of its pseudonyms. Our two-level authentication scheme has been designed to prevent an adversary from forging pseudonyms. However, used pseudonyms are valid ones that can be reused to authenticate adversaries without any trouble. A given user can only refuse to authenticate a pseudonym that is already on its own blacklist, while the adversary can cheat different users under the same pseudonym. The adversary's identity remains undiscovered even if every user in the network has detected it misbehaving under that single same pseudonym.

Both of the problems can be solved through sharing the lists of misbehaving pseudonyms and revoked identities. This allows more effective adversary identification and prevent pseudonym reuse by the adversary. Each time a user detects a misbehaving pseudonym, it can inform other users by broadcasting a revoke mes-

sage.

It is important to ensure authenticity of the revoke messages. Distribution of bogus revoke messages with invalid pseudonyms consumes not only communication but also computation resources. Any identity reconstruction effort involving invalid pseudonyms will be unsuccessful. We propose to employ Paterson's ID-based signature scheme [55] as a simple extension of the original protocol. The following additional operations are performed by the SA during the bootstrapping phase:

1. Picks an element $P \in G$ and calculates $Ppub = tP$.
2. Chooses another collision-resistant hash function

$$H_3 : G \rightarrow \{0,1\}^{\beta}.$$

3. Makes $< P, Ppub, H_3 >$ known to every user.

The revoke message in our scheme has the format of

$$< REVOKE, [PsdNym_{reporter}, PsdNym_{misbehaver}]_{PsdNym_{reporter}} >$$

$PsdNym_{misbehaver}$ is the reported misbehaving pseudonym. $PsdNym_{reporter}$ is the pseudonym of the reporter. $[M]_{ID}$ stands for signing message M using the identity ID. The signature is a pair of value (R, S). To generate the signature, user ID picks a random number $k \in Z_q^*$ and calculates:

$$\begin{cases} R &= k \cdot P \\ S &= k^{-1}(H_2(M) \cdot P + H_3(R) \cdot tH_1(ID)) \end{cases} \tag{4.17}$$

To verify that (R, S) is a valid signature generated by user ID for M, the

verifier confirms that

$$\begin{aligned}\hat{e}(R,S) &= \hat{e}(k\cdot P, k^{-1}(H_2(M)\cdot P + H_3(R)\cdot tH_1(ID))) \\ &= \hat{e}(P, H_2(M)\cdot P + H_3(R)\cdot tH_1(ID)) \\ &= \hat{e}(P,P)^{H_2(M)}\cdot \hat{e}(Ppub, H_1(ID)^{H_3(R)} \end{aligned} \tag{4.18}$$

Furthermore, to prevent the reporter from injecting fake misbehaving pseudonyms into the blacklist. Authenticity of the $PsdNym_{misbehavior}$ also needs to be verified. This can be enabled by attaching a signature on $PsdNym_{misbehavior}$ generated by $PsdNym_{misbehavior}$ itself. Such a signature can be obtained during the anonymous authentication procedure. Two users may exchange their self-generated signatures on their own pseudonyms right after the second secret handshake is successfully performed.

4.4.5 Packet-based Pseudonyms

After the two secret handshakes, Alice and Bob can securely communicate with each other using their pseudonyms and the secret key. However, the same pseudonyms will link together all the communications and transactions between Alice and Bob. The same approach of MASK[14] can be adopted to ensure unlinkability between two packets or two transactions. Alice and Bob calculate a sequence of shared keys and pseudonymous identifiers as:

$$\begin{cases} PsdNym^{(\gamma)}_{Alice,Bob} &= H_2(S^{(2)}_{Alice,Bob}||nounce^{(2)}_{Alice}||nounce^{(2)}_{Bob}||1*\gamma+1) \\ K^{(\gamma)}_{Alice,Bob} &= H_2(S^{(2)}_{Alice,Bob}||nounce^{(2)}_{Alice}||nounce^{(2)}_{Bob}||1*\gamma) \end{cases} \tag{4.19}$$

where γ, and $PsdNym^{(\gamma)}_{Alice,Bob}$ and $K^{(\gamma)}_{Alice,Bob}$ denote the γ^{th} pseudonymous identifier and the corresponding shared secret key. A new pair of $< PsdNym^{(\gamma)}_{Alice,Bob}, K^{(\gamma)}_{Alice,Bob} >$ is used for every packet transmitted or transaction performed between Alice and Bob. The maximum number of such pairs should be small enough such that the probability of collision is negligible.

4.5 Discussion

In this subsection, I will discuss some properties of the protocol, design choices, and several enhancements.

4.5.1 Two Rounds of Secret Handshakes

It can be immediately observed that during the first handshake, Alice and Bob used only parts of their pseudonyms, which are absolutely-unlinkable random numbers in G. An eavesdropper who overhears these cannot obtain any information about user identities. For the second handshake, exchange of the remaining parts of the pseudonyms is secured using the newly established secret key $K^{(1)}_{Alice,Bob}$. Since the key $K^{(1)}_{Alice,Bob}$ is particularly linked to Alice and Bob, nobody else can obtain the key or the pseudonyms. The authentication procedure is anonymous against eavesdroppers who can be either outsiders or compromised insiders.

It is more complicated when one of Alice and Bob is compromised, say Bob. Bob can recover Alice's identity by repeatedly performing authentication with Alice

for at least k different pseudonyms of Alice. From then on, whenever Alice authenticates with Bob, Bob can link the pseudonyms to Alice. As a result, Bob can link all his own authentication experiences with Alice. However, Bob also discloses his own pseudonyms and risks his own privacy.

On the other hand, our scheme is more robust than it first appears. In fact, communications and transactions between benign users remain anonymous even if one or both of their identities have been recovered by the adversaries. Again, this is because of the two-round secret handshake mechanism. For example, the secret handshake procedure between Alice and Carl remains unrecognizable to Bob, even if he has recovered the identities of both Alice and Carl. Our scheme ensures robust anonymity of communications and transactions within the group of benign users.

4.5.2 Distributed Adversary Identification

As recognized above, the fact that an internal spy may recover user's identities posts threats against the system anonymity. This problem can be alleviated by further dividing the capability of re-identification across the network users. First, every pseudonym is encrypted before loaded to users. Secret points are calculated from the encrypted pseudonyms correspondingly. The encrypted pseudonyms have to be decrypted before being used to reconstruct the user identity. Then, according to the schemes of [56] or [57], the decryption capability can be shared by the network users such that it requires the cooperation of a minimum number of users to decrypt the pseudonyms. Assume that the minimum number is D. Now, a user has to

convince at least another $D-1$ users to help it decrypt a misbehaving pseudonym. The user can do so by presenting evidence of misbehaviors. Honest users refuse to help the decryption unless proper evidence is provided. As a result, it requires that at least D users to be compromised in order to decrypt the pseudonyms and recover other users' identities.

However, extra care has to be taken to ensure the distributed decryption procedure does not introduce new breaches of anonymity. We suggest a separate set of identities, unlinkable to the ones for data communication, to be used for the purpose of distributed decryption.

4.5.3 Threshold

One important property of our scheme is the flexible trade-off between anonymity and accountability through adjusting the design parameter k. On one hand, a given adversary may be identified if it has been caught misbehaving at least k time. The smaller k is, the sooner an adversary can be identified. On the other hand, spies can recover the real identity of honest users, once they obtain at least k pseudonyms of a given user. The smaller k is, the easier for spies to breach the user anonymity. Moreover, benign users may be mistaken as misbehavers, for reasons such as dynamic channel status. The number of unwanted identification of benign users increases as k decreases, and vice versa. The proper value of k should be determined according to the application requirements. Larger k is preferred for privacy sensitive applications, and smaller k is preferred when fast misbehavior detection and misbehaver

identification are critical.

4.5.4 Pseudonym Reloading

As users are preloaded with limited number of pseudonyms, they either have to reuse their pseudonyms, or there will be a time when a user has to go back to the trusted authority to be reloaded with a fresh set of pseudonyms. Note that although each user having access to its own pseudonyms may reconstruct the polynomial function and generate pseudonyms for themselves, they much obtain the corresponding secret points from the SA. Without the corresponding secret points, users can no longer be authenticated using the pseudonyms. This prevents a compromised user from introducing arbitrary pseudonyms into the system.

There are two options when constructing the new set of pseudonyms:

1. A new random polynomial of order $k-1$ is picked for the same secret, and the new set of pseudonyms is calculated using this new polynomial.
2. The same polynomial is used, but it is evaluated at different random points.

The major difference between these two options is whether the new pseudonyms can be combined with the old ones when reconstructing the user identity. Option 1) suggests that a different polynomial is used, so the new set of pseudonyms cannot be combined with the old ones. Option 2) is the opposite case. A disadvantage of option 1) is that an adversary can dodge detection by reloading its pseudonyms whenever $k-1$ pseudonyms from the same set are used. Option 2) prevents this problem, but provides less confidence in users' anonymity. A spy who has collected

any k pseudonyms of the same user can recover the user's real identity. On the other hand, option 1) requires the spy to collect at least k pseudonyms from the same set. And more importantly, with option 1), an exposed user regains anonymity once it gets reloaded with the new set of pseudonyms.

Which option to take depends on the interests of the actual applications. In applications where protecting anonymity is more important than detection of misbehaving users, option 1) is preferred, while the opposite is true for option 2).

4.5.5 An example application of Secure and anonymous routing

In this section, we discuss an example application our anonymous authentication architecture to provide secure and anonymous routing in ad-hoc wireless networks. We consider the case when compromised or selfish users agree to forward the data packets but fail to so. It is critical to detect such misbehaviors under all the constraints of anonymous communication.

Without anonymity in consideration, a group of secure routing protocols ([49, 51, 52]) have been designed to defend agaisnt such misbehaviors. The so-called *Watchdog* mechanism is initially proposed in [51], and later extended in ([49, 58]. The basic idea is based on the use of *passive acknowledgment (PACK)*: a node can confirm that its neighbor has received a packet by overhearing it forwarding that packet. However, this PACK mechanism cannot be applied to anonymous communications. This is because to ensure anonymity, hop-by-hop encryptions are generally applied to the packets to prevent transmission of the packet from being

traced according to its payloads. This prevents a node from being able to monitor its neighbors' transmissions.

Solutions proposed in ([52, 59]) makes use of active acknowledgments and reports. HADOF[52] is based on the source routing protocol DSR [22]. Source in HADOF collects traffic statistic reports from the intermediate users in order to detect misbehaving users on the route. On the other hand, the secure data forwarding (SDF) scheme [59] is based on the distance-vector routing protocol AODV [21]. In SDF, the destination generates ACKs that can be verified by the source and every intermediate user. For every packet, if the source or an intermediate user receives neither the destination ACK nor a misbehaving report from its downstream within a certain amount of time, it will generate a misbehaving report about its downlink and send it upstream to the source. When the source detects its own downlink misbehaving or receives a misbehaving report from the downstream, the source can issue a new secure route request avoiding the misbehaving users. Because SDF does not rely on source routing, which is the case for most anonymous routing protocols, we suggest employing SDF as the misbehavior detection mechanism to implement secure and anonymous routing protocols.

4.6 Conclusion

In this chapter, we observe that existing anonymous communication protocols designed for ad hoc networks have not been able to accommodate accountability, they are seriously vulnerable to other forms of threats, such as the Denial-of-Service

attack. Based on this observation, we presented an anonymous authentication architecture with distributed anonymity revocation. While it is impossible to provide perfect anonymity and accountability at the same time, our scheme provides a framework that allows flexible trade-off between these two security requirements. Anonymity is provided, and can be revoked, through using pseudonyms, which are specially constructed according to the threshold secret sharing schemes. Based on the secret handshake approach, our authentication protocol ensures anonymity of benign users against eavesdroppers. A special property of our scheme is that transactions between benign users are strongly protected in terms of anonymity, even if both users' identities are already recovered by adversaries.

Future work includes designing and implementing communication protocols under the same framework. In particular, we plan to investigate how anonymous and yet secure routing protocols can be implemented through this approach. Another closely related topic is intrusion detection, especially distributed intrusion detection. As accountability is provided, more efforts are needed to adapt current (or design new) intrusion detection schemes to work under the same framework.

Chapter 5

Conclusion

It has been clearly demonstrated that covert communication can occur via controlling ad-hoc network protocols. Performance of the covert channels depends on various network parameters. In case of the covert channel based on AODV, network mobility turns out to be beneficial to the covert channel performance. Larger network population allows more information conveyed in each covert transmission, but it also increases the probability of loss. Very high and very low maximum transmission powers both suppress the covert transmission. Data traffic generates the need for routes, thus high traffic rate helps the covert communication. For the covert channel based on the splitting tree algorithm, the covert transmission is error free. Better throughput is obtained under smaller network size and higher traffic rate. At low traffic rate, the covert transmitter can improve the throughput by aggressively transmits dummy packets, at the cost of being more easily detectably. The cover transmitter can also make strategic moves according to its own eagerness to transmit and willingness of being exposed. Various improvements to the basic splitting algorithms do not eliminate the covert channel, although they have different affects on it.

To support anonymous communication in wireless ad hoc networks, a novel construction of anonymous trapdoor has been presented. The new trapdoor con-

struction scheme requires simple key management, provides strong anonymity, supports anonymous authentication and key establishment, and thus is most compatible with other secure routing schemes.

Finally, it is observed that existing anonymous communication protocols designed for ad hoc networks have not been able to accommodate accountability. Based on this observation, an anonymous authentication architecture with distributed anonymity revocation is proposed. While it is impossible to provide perfect anonymity and accountability at the same time, our scheme provides a framework that allows flexible trade-off between these two security requirements. Anonymity is provided, and can be revoked, through using pseudonyms, which are specially constructed according to the threshold secret sharing schemes. Based on the secret handshake approach, our authentication protocol ensures anonymity of benign users against eavesdroppers. A special property of our scheme is that transactions between benign users are strongly protected in terms of anonymity, even if both users' identities are already recovered by adversaries.

Future work includes incorporating the new trapdoor into a complete anonymous routing protocol, and evaluating the routing performance. Given the proposed distributed anonymity revocation architecture, current intrusion detection schemes can be adapted (or new schemes are to be designed) to work under the same framework. A joint investigation is needed that incorporates all different secure properties using various secure mechanisms across multiple protocol layers of the network.

Bibliography

[1] B. W. Lampson. A note on the confinement problem. *ACM Comm.*, 16:613–615, October 1973.

[2] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, 2003.

[3] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, 1981.

[4] M. Freedman and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In *ACM Conference on Computer and Communications Security(CCS'02)*, 2002.

[5] M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–48, 1999.

[6] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.

[7] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. Sdar: A secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *LCN*, pages 618–624, 2004.

[8] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, 2003.

[9] Y-C. Hu and H. J. Wang. Location privacy in wireless networks. In *Proceedings of the ACM SIGCOMM Asia Workshop 2005*, 2005.

[10] W.S.Juang, C.L.Lei, and C.Y.Chang. Anonymous channel and authentication in wireless communications. In *Computer Communications 22*, 1999.

[11] J. Kong and X. Hong. Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Mobihoc'03*, Annapolis, MD, June 2003.

[12] J.J.Kong, D.P.Wu, X.Y.Hong, and M.Gerla. Mobile traffic sensor network versus motion-mix: Tracing and protecting mobile wireless nodes. In *ACM Security of Ad-hoc and Sensor Networks(SASN'05)*, 2005.

[13] S. Seys and B. Preneel. Arm: Anonymous routing protocol for mobile ad hoc networks. In *AINA 2006: Proceedings of the 20th IEEE International Conference on Advanced Information Networking and Applications - Workshops*, 2006.

[14] Yanchao Zhang, Wei Liu, and Wenjing Lou. Anonymous communications in mobile ad hoc networks. In *INFOCOM 2005: 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.

[15] V. Gligor. A guide to understanding covert channel analysis of trusted systems. Technical Report NCSC-TG-030, National Computer Security Center, Ft. George G. Meade, MD, U.S.A., November 1993.

[16] *Department of Defense Trusted Computer System Evaluation Criteria*. Number DoD 5200.28-STD. National Computer Security Center, December 1985.

[17] S. Li and A. Ephremides. A network layer covert channel in ad-hoc wireless networks. 2004.

[18] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM'94*, October 1994.

[19] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile radio networks. In *Proc. 1989 IEEE Military Communications Conf.*, Boston, MA, 1989.

[20] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *ACM/Baltzer J. Wireless Networks*, 1(1):61–81, February 1995.

[21] C.E. Perkins, E. M. Royer, and S. R. Das. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[22] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[23] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, Japan, April 1997.

[24] Z.J. Haas, M.R. Perlman, and P. Samar. Determining the optimal configuration for the zone routing protocol. In *IEEE Journal on Selected Areas in Communications, special issue on Ad-Hoc Networks*, August 1999.

[25] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *IEEE INFOCOM*, Israel, March 2000.

[26] M. Guerrero. Secure ad hoc on-demand distance vector (saodv) routing. In *ACM SIGMOBILE Mobile Computing and Communication Review*, July 2002.

[27] Y-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *MobiCom'02*, Atlanta, GA, September 2002.

[28] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conf. (CNDS 2002)*, San Antonio, TX, January 2002.

[29] N. B. Lucena, D. F. Calvert, J. Pease, and S. J. Chapin. Semantics-preserving application-layer protocol steganography. ProtoStego.pdf.

[30] L. Buttyan and J.P. Hubaux. Enforcing service availability in mobile ad hoc wans. In *Mobicom'00*, Boston, MA, August 2000.

[31] S. Li and A. Ephremides. A covert channel in mac protocols based on splitting algorithms. 2005.

[32] J. I. Capetanakis. *The Multiple Access Broadcast Channel: Protocol and Capacity Considerations*. PhD thesis, Massachusetts Institute of Technology, 1977.

[33] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, Inc, 1992.

[34] Augustus J. E. M. Janssen and Mare J. M. de Jong. Analysis of contention tree algorithms. 46(6), September 2000.

[35] Y. E. Sagduyu and A. Ephremides. Energy-efficient collision resolution in wireless ad-hoc networks. In *Proc. IEEE INFOCOM'2003*, San Francisco, April 2003.

[36] N. Abramson. The aloha system - another alternative for computer communications. In *Fall Joint Comput. Conf., AFIPS Press*, 1970.

[37] N. Abramson. Packet switching with satellites. In *Fall Joint Comput. Conf., AFIPS Press*, 1973.

[38] L. Kleinrock and F. A. Tobagi. Packet switching in radio channels: Part i - carrier sense multiple-access modes and their throughput-delay characteristics. In *IEEE Transactions on Communications*, December 1975.

[39] P. Karn. Maca - a new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th computer Networking Conference*, 1990.

[40] IEEE Standards Department. Wireless lan medium access control (mac) and physical layer (phy) specifications. In *IEEE Standard 802.11-1999*.

[41] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudeonymity — a proposal for terminology. In *International workshop on Designing privacy enhancing technologies*, pages 1–9, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[42] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H-C. Wong. Secret handshakes from pairing-based key agreements. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 180, 2003.

[43] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: PRIVATE Communication in a PUBLIC World*. Prentice Hall, Inc, 2002.

[44] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[45] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[46] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *SCIS 2000: Symposiums on Cryptography and Information Security*, 2000.

[47] S. Li and A. Ephremides. Anonymous routing: A cross-layer coupling between application and network layer. 2006.

[48] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1981.

[49] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol. In *Mobihoc*, pages 226–236, 2002.

[50] P. Kyasanur and N. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, 2003.

[51] S. Marti, T.J. Guili, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobicom'00*, Boston, MA, August 2000.

[52] W. Yu, Y. Sun, and K. J. R. Liu. Defense against routing disruptions in mobile ad hoc networks. In *INFOCOM 2005: 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.

[53] A. Shamir. How to share a secret. In *Communications of the ACM 22*, 1979.

[54] G. J. Simmons. An introduction to shared secret and/or shared control schemes and their application. In *Contemporary Cryptology, The Science of Information Integrity, IEEE Press*, 1992.

[55] K. G. Paterson. Id-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[56] C. Cachin. An asynchronous protocol for distributed computation of rsa inverses and its applications. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, 2003.

[57] D.Catalano, R.Gennaro, and S.Halevi. Computing inverses over a shared secret modulus. In *Eurocrypt'00, LNCS 1807*, 2000.

[58] P. Michiardi and R. Molva. core: a collaborative reputatio mechanism to enforce node cooperation in mobile ad hoc networks. In *IFIP - Communications and Multimedia Security Conference*, 2002.

[59] Q. Huang, I. Avramopoulos, B. Liu, and H. Kobayashi. Secure data forwarding in wireless ad hoc networks. In *IEEE International Conference on Communications (ICC 2005)*, 2005.