URSA: Ubiquitous and Robust Access Control for Mobile Ad-Hoc Networks

Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, Lixia Zhang
UCLA Computer Science Department, Los Angeles, CA 90095-1596
{hluo,jkong,pzerfos,slu,lixia}@cs.ucla.edu

Abstract

Restricting network access of routing and packet forwarding to well-behaving nodes, and denying access from misbehaving nodes are critical for the proper functioning of a mobile ad-hoc network where cooperation among *all* networking nodes is usually assumed. However, the lack of a network infrastructure, the dynamics of the network topology and node membership, and the potential attacks from inside the network by malicious and/or non-cooperative selfish nodes make the conventional network access control mechanisms not applicable. We present *URSA*, a ubiquitous and robust access control solution for mobile ad-hoc networks. URSA implements ticket certification services through *multiple-node consensus* and *fully localized instantiation*, and uses tickets to identify and grant network access to well-behaving nodes. In URSA, no single node monopolizes the access decision or is completely trusted, and multiple nodes jointly monitor a local node and certify/revoke its ticket. Furthermore, URSA ticket certification services are fully localized into each node's neighborhood to ensure service ubiquity and resilience. Through analysis, simulations and experiments, we show that our design effectively enforces access control in the highly dynamic, mobile ad-hoc network.

1 Introduction

The emerging mobile ad-hoc networking technology seeks to provide users "anytime" and "anywhere" services in a potentially large infrastructureless wireless network, based on the collaboration among individual network nodes. The growing civilian and military interest in these networks has made the access control service increasingly important.

This paper addresses the problem of access control for a mobile ad-hoc network. Our specific interest is on the access to the network-layer functionalities of routing and packet forwarding. We seek to grant access to well-behaving nodes and deny access from misbehaving nodes. A misbehaving node can be either a *selfish* or a *malicious* node. A selfish node may enjoy network services, e.g., receiving packets destined for itself, but refuse to route or forward packets for others, therefore invalidating the basic collaboration premise in almost all current routing algorithms for mobile ad-hoc networks. A malicious node may seek to damage or disrupt normal network operations. Moreover, a misbehaving node may act as a good network citizen for certain *time period* or in certain *place*s, but then starts to act selfishly or maliciously in other time or locations.

Access control for mobile ad-hoc networks is challenging for several reasons. First, unlike a wired or wireless cellular network where access control mechanisms can be deployed at the access routers or base stations, an ad-hoc network is infrastructureless and does not possess a well-defined, clear line of defense. Access control in an ad-hoc environment is a distributed problem by its nature. Second, mobile users may roam freely in a potentially large network, and they demand "anytime, anywhere" ubiquitous services. It is desirable that any services for access control be available at each networking node's locality, in order to avoid communication over highly unreliable, multihop wireless channels [24]. Third, the solution has to handle misbehaviors by selfish and malicious nodes from inside the network. These misbehaving insiders may already possess certain information on the access control. Finally, the solution has to function with dynamic node membership, i.e., nodes join, leave and fail.

In this paper, we propose *URSA*, a fully localized design paradigm to provide ubiquitous and robust access control for mobile ad-hoc networks. The proposed solution takes a ticket-based approach. Each well-behaving node uses a certified ticket to participate in routing and packet forwarding. Nodes without valid tickets are classified as being misbehaving. They will be denied from any network access, even though they move to other locations. Thus, misbehaving nodes are "isolated" and their damage to the mobile ad-hoc network is confined to their locality.

The operation of URSA access control emphasizes *multiple-node consensus* and *fully localized instantiation*. Since any individual node is subject to misbehaviors, we do not rely on any single node. Instead, we leverage the nature of cooperative computing in an ad-hoc network and depend on the collective behaviors of multiple local nodes. In URSA, multiple nodes in a local network neighborhood, typically one or two-hop away, collaborate to monitor a node's behavior and determine whether it is well-behaving or misbehaving using certain detection mechanism of their choice. The expiring ticket of a well-behaving node will be renewed collectively by these local monitoring neighbors, while a misbehaving node will be denied from ticket renewal or be revoked of its ticket. In this way, the functionality of a conventional access control authority, which is typically centralized, is fully distributed into each node's locality. Every node contributes to the access control system through its local efforts and all nodes collectively secure the network.

We also use soft states to enhance the system robustness against potential attacks. The ticket not only is certified, but also has to be renewed regularly to preserve the bearer's good network citizenship. To further resist conspiracy of attacks by multiple misbehaving nodes, we use soft states to periodically refresh the certification function itself. An additional benefit of the soft-state based design is that it handles well the issue of dynamic node join and leave.

The implementation of URSA is based on refined threshold cryptography algorithms, which operations are fully localized. The communication protocol also confines message exchanges within a local network neighborhood.

We demonstrate the effectiveness of URSA through both simulations and real experiments. We show that URSA provides ubiquitous and robust services to restrict network access to well-behaving mobile nodes, while achieving communication efficiency that fits well in the ad-hoc networking context. The results also reveal that there is a fundamental tradeoff between security strength and mobility support. Roaming users cannot move too fast in order for the network monitoring and ticket services to be effective.

The rest of this paper is organized as follows. Section 2 reviews and compares with the related work. Section 3 specifies system models including the network model, the localized trust model, and the attack model. Section 4 describes URSA design. Section 5 describes our cryptographic algorithms and communication protocols. The performance is evaluated in Section 6 via our implementation in UNIX systems and the *ns-2* network simulator. We discuss several important issues in Section 7 and conclude this paper in Section 8.

2 Related Work

Driven by the demand for ubiquitous connectivity of mobile users and the consequent deployment of public networks, a number of network access control systems have been recently developed. In SPINACH [1], users are authenticated through Kerberos-enabled telnet. Based on MAC addresses, SPINACH routers only allow the DHCP traffic, SPINACH server traffic and authorized user traffic to go through the network. NetBar [29] separates public LANs for configuration and authentication. Traffic coming from a public port is confined with limited connectivity to the authentication and DHCP servers. Full connectivity is granted only after proper authentication. NetBar is also adopted in the InSite [19] system. In [38], access control is implemented through the collaboration of intelligent hubs that are capable of disabling and enabling specific ports and an authentication-enhanced DHCP server, so that only authenticated clients are properly configured. Although they are effective in restricting access by unauthorized mobile users to a fixed, wired or wireless network infrastructure, these designs do not apply in the context of mobile ad-hoc networks where no reliable network infrastructure exists. There is no switch, router, gateway, or dedicated servers where the client traffic converges to be regulated or audited. As the network itself is composed of autonomous mobile clients, it is subject to impact due to the potential misbehaviors of any individual entity.

Network firewalls [8] can effectively regulate inbound and outbound traffic according to the address/port filtering patterns and other security policies. Remote-access Virtual Private Network (VPN) [15] can provide protected connection for authorized users to access private networks via public, "unauthorized" networks. In the telecommunication cellular networks such as GSM, each subscriber is identified by a SIM smart-card and must be authenticated by its home switch center before the call request is accepted. In all these systems, there is a clear boundary between service-provisioning infrastructure and the mobile users that access the network services, where the service provider's access policy can be implemented by a variety of techniques such as access control lists and capabilities. If such a supporting infrastructure is not available, e.g., in a mobile ad-hoc network, new mechanisms have to be devised for network access control.

COCA [46] and its application in ad-hoc networks [45] propose the deployment of a group of servers for the purpose of certificate management. At a first glance, this approach is plausible for access control service too. For example, every mobile node can be issued a ticket by these servers to participate in networking activities. However, in order to effectively identify and isolate malicious nodes and selfish nodes, these tickets have to be periodically renewed or certain ticket revocation service has to be maintained at these servers. In either case, heavy communication with these servers is involved, which may stress the mobile ad-hoc network due to the bandwidth limit of the wireless link and the multihop routing failures caused by node mobility [4]. We further examine this problem through simulations in Section 6.2. This problem is expected to be more severe as the network size increases, because the non-localized traffic, introduced by communication with one or a few centralized servers, will cause the per-node available bandwidth to diminish to zero [17, 26].

A significant amount of effort has been invested in the IEEE 802.11 standard family [23], the dominant technology for wireless data networks, to protect the network from unauthorized access. An intruder has to be able to monitor and understand the IEEE 802.11 physical layer, and transmit according to the specific setting. However, necessary hardware that is capable of eavesdropping and injecting packets in an IEEE 802.11 network is already available off-the-shelf, e.g., Lucent Wi-Fi Orinoco PC Cards. At the link layer, IEEE 802.11 has the option to drop all the packets that are not properly encrypted with WEP (Wired Equivalent Privacy). However, Borisov et al. [3] pointed out that the WEP-based design is subject to a number of attacks and cannot achieve its claimed goal of access control. More importantly, the WEP-based access control again is for the infrastructure mode where mobile nodes access the network through access points (APs) attached to a backbone distribution system. It does not handle selfish or malicious nodes if applied in the alternative infrastructureless (ad-hoc) mode. There are a few recent studies of secure communication [7] and cooperation stimulation [5] for mobile ad-hoc networks. However, they assume each node be equipped with either secure side channel [7] or tamper-resistant hardware [5], whose availability cannot be guaranteed in general.

3 System Models

In this section, we specify the network model, the trust model, and the attack model for the URSA design.

3.1 Network Model

We consider a wireless, mobile, ad-hoc network. Nodes communicate with one another via bandwidth-constrained, error-prone, and insecure wireless links. Reliable transmission over multihop wireless paths is not assumed.

The network is composed of n nodes, and n may be dynamically changing as nodes join, leave, or fail over time. Besides, n is not constrained; there may be a large number of networking nodes. Each node has a unique, non-zero ID or name. It can be a MAC-layer address or an IP address. A node is capable of discovering its one-hop neighbors by running some neighborhood discovery protocol. Such protocols are part of most existing ad-hoc routing protocols, therefore can be reused.

We also assume that each node is equipped with a local, one-hop monitoring mechanism to detect misbehaving nodes among its direct neighbors. One of the examples is "watchdog" [28] that implements at a promiscuous wireless interface and monitors the one-hop broadcast wireless channel. More sophisticated mechanisms are shown in [40]. We further discuss this issue in Section 7. Furthermore, we assume that the network is dense enough, such that a typical node has at least k one-hop well-behaving neighbors. We will discuss the issue of insufficient neighbors later in Section 4.3.2.

3.2 Localized Group Trust Model

The notion of trust is fundamental in any security design. There are two major trust models in the literature. In the TTP (trusted third party) trust model [32], a node is trusted if it is trusted by a central authority. While the implementations of the TTP model feature the efficiency and manageability of centralized systems, they suffer from scalability and robustness problems if applied in a large ad-hoc network. In the PGP "web-of-trust" model [47], a node manages its own trust based on recommendations. The drawback is that the trust relation between any two nodes is not well structured or well-defined. Besides, it does not handle dynamic node memberships well.

We define a localized *group trust* model for mobile ad-hoc networks as the foundation of the URSA design. That is, a node is trusted if it is trusted by any k trusted nodes. A locally trusted node is accepted network-wide, while a locally distrusted node is regarded as untrustworthy everywhere in the network. The trust relation is also soft-state and is defined within a certain time period T_{cert}. In this model, k and T_{cert} are two important parameters, where k characterizes the collective decisions by multiple nodes and T_{cert} characterizes the time-evolving nature of trust.

There are two possible options to set k. The first is to set it as a globally fixed parameter. In this case, k acts as a system-wide trust threshold. The second option is to set k as a location-dependent parameter. For instance, k may be the majority of a node's neighboring nodes. This second option provides maximum flexibility to fit in specific network topologies. However, there is no clear system-wide trust criterion. Since the number of neighbors used in the model may vary and is not known *a priori*, it can be exploited by misbehaving nodes. Therefore, we choose the first option with a network-wide fixed k. The specific choice of k is tuned according to the network density and desired system robustness.

In our localized group trust model, trust management and maintenance are distributed in both spatial (k) and temporal (T_{cert}) domains. This property is particularly appropriate for a large, *dynamic*, ad-hoc wireless network, where centralized trust management would be difficult or costly. Besides, a node indeed cares most about the trustworthiness of its immediate neighbors, since it relies on them to connect with the rest of the network.

3.3 Attack Model

Our focus in this paper is to deal with potential misbehaviors at the network layer, whose main functionality is routing and packet forwarding. We do not explicitly consider attacks at other layers, e.g., physical-layer jamming and MAC-layer misbehaviors.

The attack can be initiated by a *single* selfish or malicious node. A selfish node poses threats to nearly all the existing ad-hoc network routing algorithms and protocols where co-operation among all participating nodes is a prerequisite. We refer to recent publications [28, 22] for the discussions on how to detect such protocol disruptions with bounded reaction time. The attacks by a malicious node can be directed towards the wireless link or other well-behaving nodes. For the wireless link, a malicious node can eavesdrop, record, inject, re-order, and re-send (altered) packets. It may also blast dummy messages in a brute-force form of network-layer denial-of-service attack. Most such misbehaviors can be detected via local mechanisms of overhearing the broadcast channel [28, 40]. Efficient message authentication protocols (e.g., TESLA [33]) can be employed to get rid of unauthenticated and out-of-date packets. A malicious node can also target at other nodes. It may occasionally compromise and control one or a few well-behaving nodes through software bugs or system backdoors.

Multiple misbehaving nodes may conspire to initiate *collaborative* attacks such as joint accusation against a well-behaving node. We limit the scope and capability of such malicious collaborations in our attack model. Specifically, we consider two cases in this paper:

- Model I: During the entire lifetime of the network, the number of collaborative malicious nodes is less than k.
- Model II: The lifetime of the network is divided into time intervals of length T each. During any time interval T, there is less than k collaborative malicious nodes.

Note that the above attack model is different from models in the literature of secure routing for mobile ad-hoc networks in two aspects. First, the related work mainly focuses on the possible attacks against a bare-bone network that does not have any security protection. Attackers are mainly outsiders of the system, and many of such attacks can be resisted through authentication protocols assuming key management in place [35, 42, 2, 31, 20, 21]. Our concern is more towards the insider attacks by selfish or malicious nodes. These nodes are already in the system and they have limited knowledge of protocol states and security settings. Second, we consider not only attacks by a single node, but also collaborative attacks by multiple nodes. We limit the capability and scope of such attacks, since attackers given unbounded capability can overwhelm any practical system.

Finally, mobility also introduces new vulnerability to large ad-hoc networks. A malicious node may roam from one location to another to extend the impact of its attacks.

4 URSA Design

In this section, we describe URSA design. We first provide an overview of URSA system in Section 4.1, and then present URSA tickets in Section 4.2. URSA ticket services are presented in Section 4.3, and bootstrapping for URSA system is presented in Section 4.4. We analyze how URSA handles potential attacks in Section 4.5. Finally soft-state is employed to further enhance the resilience of the URSA ticket services against attacks in Section 4.6.

4.1 Overview

In a mobile ad-hoc network that is protected by URSA, each networking node is required to carry a valid ticket in order to participate in network activities. A ticket is considered valid if it is certified and unexpired. When an existing node moves to a new location, or a new node joins the network, it exchanges tickets with its one-hop neighboring nodes to establish mutual trust relationship. Misbehaving nodes without valid tickets will be denied from all networking activities, therefore isolated from the mobile ad-hoc network.

URSA ticket services ensure that ideally only well-behaving nodes receive tickets. The implementation of ticket renewal and revocation services is fully distributed into each well-behaving node through an *initialization* process during the bootstrapping phase of the network. For nodes that join or rejoin the network, they can be initialized by a certain number of neighbors in order to serve other nodes for ticket renewal and revocation. Neighboring nodes also monitor each other during the normal operations with certain misbehavior detection mechanisms of their choice. When its ticket is about to expire, a node solicits its neighboring nodes to collectively renew its ticket. A neighboring node responds to such request only if the requesting node is considered as being well-behaving during the last monitoring period. Furthermore, an accusation message will be flooded locally once a misbehaving node is detected by any of its neighboring nodes. Misbehaving nodes, once detected and convicted (to be elaborated in Section 4.3.3), will not be able to have their tickets renewed.

Note that URSA ticket services do not issue brand-new tickets: a node needs a still-valid ticket when it requests ticket renewal from its neighborhood. There are several options to issue an initial ticket and admit a new node into the network. A new node may obtain its initial ticket from a coalition of existing nodes, after its authenticity being verified through external means (e.g., in-person ID). Alternatively, a new node can be “tentatively” admitted to the network and be closely monitored during the trial period. Nodes holding such trial tickets can forward packets for others but cannot have their own packets delivered. Finally, a new node can be offered the option to access the network by paying certain amount of money to the existing legacy nodes. This financial charge can effectively discourage a malicious node from frequently re-entering the network, and compensate the potential damage to the mobile ad-hoc network due to the admission of misbehaving nodes.

4.2 Ticket

An URSA ticket serves as a *passport* for a networking node. It provides a simple, yet effective mechanism for controlling the access from well-behaving and misbehaving nodes.

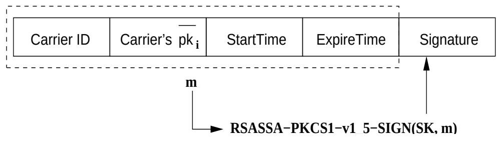

Figure 1: Ticket Format

A typical URSA ticket[1] is shown above in Figure 1. It includes the `ID` of the ticket carrier, the ticket carrier's personal $\overline{pk_i}$, the `StartTime` and `ExpirationTime`, and a `Signature` on the message digest of the ticket body. The node `ID` can be its IP address, or its MAC address in the context of an ad-hoc networking environment. The ticket binds the `Carrier's ID` and its personal public key $\overline{pk_i}$ that is usually used to establish secure communication channels. `StartTime` and `ExpirationTime` define the ticket validity period T_{cert}. The `Signature` verifies the message integrity, and is generated using a system RSA secret key (SK, N)[2]. A ticket can be verified by applying the well-known, system public key (PK, N) on the ticket `Signature`.

No single node in the network has complete information of the exponent SK of the system secret key (SK, N) that is used to sign tickets. Instead, each node with ID v_i holds a share P_{v_i} as partial information of SK. This share is used to sign the partial tickets, as we elaborate in Section 4.3.1.

Tickets identify *well-behaving* nodes. When a mobile node moves to a new location, it exchanges tickets with its new neighbors, as the first step to cross-verify each other. After receiving the ticket from its neighbor, the node verifies the ticket signature with the system public key (PK, N) and uses a standard challenge/response protocol to confirm the ticket holder's knowledge of its personal secret key $\overline{sk_i}$, corresponding to the advertised public key $\overline{pk_i}$. After these procedures, certified neighboring nodes help each other in routing and forwarding packets, while nodes without valid tickets are treated as misbehaving nodes and are denied from participating in any network-layer activities. Neighboring nodes also monitor each other to detect possible misbehaviors. The choice of specific detection mechanism is left to individual nodes.

[1]For simplicity we do not consider multi-level access control, in which nodes may be granted different capabilities in the network system.

[2]Note that the RSA secret key is also denoted as $SK : (d, n)$ and the public key denoted as $PK : (e, n)$ in the literature. We use SK and PK to represent the exponents of RSA keys in this paper.

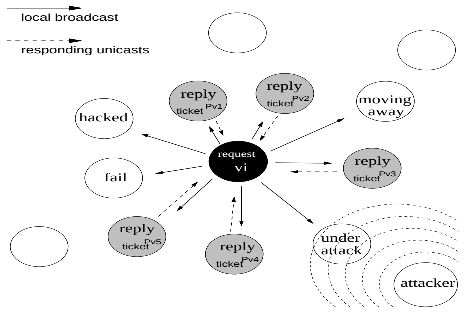

Figure 2: Localized Ticket Renewal

4.3 Ticket Services via Local Collaboration

URSA ticket services include ticket renewal and ticket revocation. In this section, we describe their localized implementation and the impact of mobility on URSA design.

4.3.1 Ticket Renewal

Tickets are stamped with expiration time. Nodes have to be issued a new ticket upon the expiration of its old ticket. Our goal is to localize this service into each node's neighborhood in order to maximize service availability and resilience against potential attacks, while conserving network resources and system scalability by localizing the communication traffic. A straightforward approach is to select a local node to renew the ticket when it is about to expire. However, it suffers from single point of failure or compromise, and inaccuracy of single-node detection mechanisms.

Our solution relies on a group of k well-behaving, neighboring nodes, and works as follows (shown in Figure 2). When the ticket of a node v_i is about to expire, it sends a ticket renewal request to its one-hop neighbors. Upon receiving such a ticket renewal request, a neighboring node v_j checks its records, generated by its chosen neighborhood monitoring mechanism during the latest monitoring period of time T_{mon}. The monitoring period T_{mon} is typically about the same order of magnitude of the average time that a node remains within the one-hop communication range. v_j returns a "partial" ticket only if its record shows that v_i is well-behaving. After collecting k partial tickets in a certain timeout, node v_i combines them together and constructs a renewed, complete ticket. The timeout value is set to allow k neighbors to process and transmit k partial tickets. It can be the sum of the processing delay and transmission time of k partial tickets, plus the estimate of the channel access time.

In the above ticket-renewal procedure, as long as k well-behaving nodes respond, the ticket renewal service is available. The service is resilient to the attacks from malicious nodes, as improper partial tickets will be dropped. On the other hand, the misbehaving node who sends out improper partial tickets will be detected by the node requesting ticket renewal, and will be denied of its own ticket renewal.

A valid ticket represents the collective trust of a coalition of k nodes. This realizes the localized group trust model presented in Section 3.2. Nodes without a valid ticket will be denied network access anywhere, thus effectively isolating "roaming" misbehaving nodes. Because the ticket renewal service is localized to each networking node's one-hop neighborhood, we realize service ubiquity for mobile nodes and eliminate any single-point of failures in the system.

From the data communication's perspective, because only one-hop packet exchange is involved in the ticket renewal service, the above *localized* mechanism improves not only efficiency and load-balancing, but also robustness in the presence of wireless channel error and node mobility-induced routing failure. Our implementation does not rely on the performance of the underlying transport or routing protocols. More importantly, the localized scheme complies with the scalability requirement, in term of per-node available bandwidth in an ad-hoc network. The analysis in [26] shows that for an ad-hoc network to scale, its traffic pattern should be localized and must not traverse long hops in the network. Conventional approaches that involve non-localized traffic patterns, e.g., communications with one or a few centralized certification servers, will cause the available per-node bandwidth to diminish in the order of $O(\frac{1}{\sqrt{n}})$ as the number of networking nodes n increases [17].

4.3.2 Mobility Impact

Node mobility brings both good and bad news into the access control system. On one hand, mobility can bring benefits. The above ticket renewal solution assumes that each node has at least k neighbors. However, if a node is located in a sparse neighborhood, it may not be able to find k neighbors. In this case, node mobility helps in two ways. First, the node which requests ticket service can simply move around to "accumulate" k partial tickets. Second, the node may anticipate other nodes to move into the neighborhood within the immediate future. In both cases, Node mobility becomes a blessing instead of a curse. We evaluate this property by simulation in Section 6.2.1.

On the other hand, mobility may also empower the misbehaving nodes with new attacks that do not exist in a stationary system. In the ticket renewal process, a node v_j serves the ticket renewal request from another node v_i only if v_j's monitoring records demonstrate that the requesting node v_i is well-behaving. Suppose the ticket validity period is T_{cert}, and node v_j's misbehavior detection mechanism requires a monitoring period of at least T_{mon}. Usually T_{cert} is set to be greater than T_{mon} to allow certain time margin for mobile nodes. That is, a mobile node can move during the first period of $T_{cert} - T_{mon}$, but still has enough time to establish its records in its new neighborhood, even if none of its new neighbors has the knowledge of its past behaviors. However, a misbehaving node can take advantage of the $T_{cert} > T_{mon}$ setting and launch an "intermittent moving attack". It can misbehave initially for an interval of $T_{cert} - T_{mon}$. If it is detected, the misbehaving node can simply move to a new location, and behave well during the rest of the period T_{mon}. This way, it is still able to establish good records and renew its ticket in the new location. The misbehaving node can repeat the above sequence and attack the network intermittently.

The above discussion indicates that, if we want to defend against intermittent moving attacks, we have to constrain well-behaving node's mobility. The more we relax the mobility constraint, i.e., allowing a longer $T_{cert} - T_{mon}$, the worse the intermittent moving attack can be. The localized ticket renewal by itself cannot completely eliminate the intermittent roaming attack. We present the ticket revocation service to deal with such attacks in the next section.

4.3.3 Ticket Revocation

In order to fight against intermittent moving attacks, we propose URSA ticket revocation. The records that a node v_j maintains consist of two parts. One is its direct monitoring records on its neighboring nodes, and the other is a ticket revocation list (TRL). Each entry of the TRL is composed of a node ID and a list of the node's accusation from others. If a node's accusation list contains less than k accusers, the node is marked as "suspect". Otherwise, the node is determined by v_j to be misbehaving and marked as "convicted". We choose the threshold that convicts a node as k to ensure that a well-behaving node is not convicted by malicious accusations.

A node is marked "convicted" in two scenarios. One is when node v_j determines by direct monitoring that one of its neighboring nodes is misbehaving. v_j puts the node into its TRL and directly marks the node "convicted". At the same time v_j also floods a signed accusation. The range of the flooding is studied below. The other scenario is when v_j receives an accusation against another node. It firstly checks whether the accuser is a convicted node in its TRL. If it is, the accusation is determined to be malicious and dropped. If not, node v_j updates its TRL entry of the accused node by adding the accuser into the node's accuser list. The accused node will be marked "convicted" if the number of accusers reaches k. Besides, v_j removes the newly convicted node from all other accuser lists. If the length of any of those accuser lists drops below k, v_j re-marks the corresponding node to "suspect".

The range of the accusation propagation is an important design parameter. A large range causes excessive communication overhead, while a small range may not be enough to isolate a mobile misbehaving node. The accusations should be propagated in a range so that before its current ticket expires, the misbehaving node cannot move out of the area where it is convicted by the accusations. One practical scheme for controlled flooding is by setting the TTL[3] field in the IP headers of the accusation packets. One way to set TTL is based on the ticket validity period T_{cert}, the maximum one-hop wireless transmission range D, and the maximum node moving speed S_{max}. In a uniformly distributed network, to ensure a misbehaving node cannot escape the area of accusation before the expiration of its current ticket, the TTL should be set at least:

$$TTL \geq \left\lceil \frac{T_{cert} \cdot 2S_{max}}{D} \right\rceil$$

If the TTL of the accusation messages is set to m, the nodes whose accusations arrive at v_j must be at most m hops away. Therefore v_j's TRL contains nodes at most $m+1$ hops away. To further decrease the TRL complexity, v_j holds each TRL entry for T_{cert} only so that it will not serve a convicted node that carries still-valid ticket. v_j removes an entry from its TRL T_{cert} after the entry's last update. The reason is that after a period of T_{cert}, a convicted node should have its ticket expired, and thus be cut off from the network.

In our design, TRL is constrained both spatially and temporally. It is built and maintained *on-demand*, and stored *locally*. These features again comply with the scalability requirement and work well with the ad-hoc networking characteristics.

4.4 Self-organized Bootstrapping

During the bootstrapping phase of the network, an authority has to send each node, which has an ID v_i, privately its share of the exponent SK of the ticket-signing key (SK, N). We denote this process the "initialization" of these networking nodes. A large ad-hoc wireless network may contain hundreds or even thousands of networking nodes. Relying on the authority for the initialization, as most early work [16, 13, 39, 12, 14, 11] does, is not scalable and may not be feasible. Moreover, new nodes may join anytime after the bootstrapping phase, how to initialize these nodes poses an additional challenge.

We address above issues by a self-organized initialization mechanism called "self-initialization". In the scheme, the authority is only responsible to initialize the very first k or slightly more nodes (our simulations used $2k$), selected out of a two-hop local neighborhood. After that, the initialized nodes collaboratively initialize other nodes, typically their neighboring nodes. Repeating this procedure that is analogous to a diffusion wave, the network progressively "self-initializes" itself. A node only needs to contact its neighbors in its locality in order to obtain its share of the system key SK. Each of its neighbors returns a "partial" share of SK. By collecting k such partial shares and combining them together, the node is initialized with its complete share of SK. This procedure of self-initialization is similar to ticket renewal as shown in Figure 2, thus exhibiting the same features of efficiency and robustness. It also applies when a well-tested, well-behaving node needs to obtain its share of SK in order to participate in URSA ticket services.

[3]TTL is defined as "time to live": the maximal number of hops that a packet can traverse in the network.

4.5 Resisting Attacks

The URSA ticket services can resist various attacks. In general, attacks can be categorized as single node attack and conspired attack by multiple attackers.

A single misbehaving node may falsely accuse other well-behaving nodes. However, false accusations can only place a well-behaving node in TRL tentatively as a suspect, but cannot convict a well-behaving node. The false accusation will also be pruned from the TRL after T_{cert}. Conventional approaches that rely on single node's decision are vulnerable to such attacks.

With both ticket renewal and revocation services in place, a roaming attacker will be promptly revoked of its valid ticket and isolated from the network. The intermittent attack can also be defeated, as discussed above.

For other attacks on the routing and forwarding functionality, we rely on the local monitoring mechanisms implemented at each node for detection. Once the alarm on a misbehaving node is raised, URSA ticket services isolate the misbehaving nodes immediately. Local cooperation among multiple nodes also helps to reduce the inaccuracy of individual node's detection. For a simple analysis, interested readers are referred to [40].

The design can also handle, to a certain extent, the conspired attacks from multiple attackers. One type of such attacks is false accusation against other well-behaving nodes. An URSA protected network can resist false accusations from up to $k-1$ attackers during an interval T_{cert}. This is again achieved by the local collaboration among k good nodes and the soft-state entry in the TRL. The URSA design also limits the chance for potential collaboration among attackers. For example, attackers may exploit the TRL to find potential attack partners. But its success chance is limited by several factors. Each accusation is only flooded locally but not globally, and each entry is purged after T_{cert}. Once a period of T_{cert} elapses, an attacker cannot receive any packet forwarding services and it is hard for the node to find other attackers without accessing the network. The only possibility for k or more attackers to collaborate is that they find out each other in T_{cert} and within a network proximity (i.e., the flooding scope of Section 4.2.3). If we set a small T_{cert} and consequently a small accusation propagation range, the success chance for attackers to discover each other will be negligible.

There are some attacks that the current system cannot resist. In a sparse network neighborhood, an attacker can isolate well-behaving nodes. Besides, k or more conspired attackers can successfully accuse good nodes. We can reduce the renewal threshold T_{cert} to make such a scenario unrealistic, but at the cost of more frequent ticket renewals.

4.6 Soft States to Improve Robustness

In order to strengthen the security of the distributed ticket-signing key (SK, N), we further apply soft states in the periodical refreshing of each node's secret share. The goal is to defeat long-term attacks by malicious nodes, where they can accumulate k or more victim nodes and their secret shares. The challenge here is to come up with a scalable and communication-efficient solution that works well in the ad-hoc network setting. Early solutions, which usually assume globally accessible broadcast channel and authenticated broadcast channel [6], are not applicable in this context.

The approach we take is a simple sequential process based on self-initialization (see Section 4.4). First, a coalition of k nodes update their shares by applying the existing protocols as proposed in [18]. Then, the neighbors of such nodes update their secret shares. This process repeats until all networking nodes refresh their secret shares of the exponent of the ticket signing key. We scratch the algorithms in Section 5.1.2 and leave the details in a technical report [27].

5 Implementation

In this section, we present our algorithms and protocols that implement the localized ticket certification services and the self-initialization of the mobile ad-hoc network. We modify existing threshold cryptography to fit in mobile ad-hoc networks. In particular, we propose a technique named *k-bounded coalition offsetting* for the multi-signature generation to scale up with the dynamic network size n. We implement our communication protocols in *ns-2* [30] simulator to evaluate the communication performance.

5.1 Cryptographic Implementation

5.1.1 Ticket Renewal

Distributed ticket renewal consists of two parts: the *distribution mechanism of the exponent SK of the ticket signing key*, and the *multi-signature generation mechanism* based on the SK distribution. There are mainly two secret sharing mechanisms in the literature: polynomial secret sharing [36] and additive secret sharing [13]. Applications of additive secret sharing in multi-signature have appeared in [39, 13, 14]. We take the polynomial secret sharing mechanism due to its ability to handle dynamic grouping.

The challenge of applying polynomial secret sharing in the context of mobile ad-hoc network is as follows. In polynomial secret sharing, k polynomial shares can be converted into k additive shares by the technique of Lagrange interpolation. These additive shares are then applied in the RSA multi-signature algorithms [11]. Let N, a product of two large random primes $N = pq$, denote the RSA modulo of the signing key (SK, N). The polynomial shares and the corresponding additive shares are usually calculated over the ring $Z_{\phi(N)}$ or $Z_{\lambda(N)}$[4]. However, the release of either $\phi(N)$ or $\lambda(N)$ makes the factorization of the RSA modulo N trivial. The solutions that are presented in [12, 11, 37] cannot be applied in the context of an ad-hoc network due to its dynamic node membership and large node ID pool.

In order to address the scalability issue to adopt the polynomial secret sharing in distributing SK among the networking nodes, we modify the multi-signature algorithms with our k-bounded coalition offsetting technique. In our algorithms, node v_i's polynomial share P_{v_i} and its additive share SK_{v_i} in term of a specific coalition, are defined over the ring Z_N, instead of $Z_{\phi(N)}$ or $Z_{\lambda(N)}$. After the initialization of the network (see Section 5.1.2), each node with its ID $v_i \neq 0$ holds a polynomial share $P_{v_i} = f(v_i) \bmod N$, where $f(x) = SK + f_1 x + \cdots + f_{k-1} x^{k-1}$ is a uniformly distributed random polynomial.

In order to renew its ticket, node v_i first broadcasts its request among a coalition $\mathcal{B}$ of k neighboring nodes. Without loss of generality, let the coalition be $\mathcal{B} = \{v_1, v_2, \cdots, v_k\}$. A neighboring node $v_j \in \mathcal{B}$ that receives the request and decides to serve the request will return a partial ticket $TICKET_{v_j}$ by applying its polynomial shares on the ticket with new expiration time:

$$TICKET_{v_j} = (ticket)^{(P_{v_j} \cdot l_{v_j}(0) \bmod N)} \mod N$$

where $l_{v_j}(0) = \prod_{r=1, r\neq j}^{k} \frac{v_r}{v_r - v_j} \bmod N$.

Upon receiving k such partial tickets $\{TICKET_{v_1}, TICKET_{v_2}, \cdots, TICKET_{v_k}\}$, node v_i combines them together using multiplication to generate a "candidate ticket" $TICKET'$:

$$TICKET' = \prod_{r=1}^{k} TICKET_{v_r} \mod N$$

[4]$\phi(N)$ denotes Euler Tortient number and $\lambda(N)$ denotes Carmichael number.

Note that:

$$\begin{aligned} TICKET' &= \prod_{r=1}^{k} TICKET_{v_r} \\ &= (ticket)^{\sum_{r=1}^{k}(P_{v_r} l_{v_r}(0) \bmod N)} = (ticket)^{t \cdot N + SK} \\ &= TICKET \cdot (ticket)^{t \cdot N} \bmod N \end{aligned}$$

where t is an integer bounded by k: $0 \leq t < k$.

Finally node v_i employs the k-bounded coalition offsetting to recover its new ticket $TICKET$.

```
Inputs:  $TICKET'$: the candidate ticket
         $ticket$: statement of the ticket, to be signed
Output:  $TICKET$: ticket

1: $Z := (ticket)^{-N} \bmod N$
2: $r := 0, Y := TICKET'$
3: while $r < k$ do
4:     $Y := Y \cdot Z \bmod N, r := r + 1$
5:     if $(ticket = Y^{PK} \bmod N)$ then
6:         break while
7:     end if
8: end while
9: output $Y = TICKET$
```

If we denote an RSA signing or verification as a unit computation, the computation complexity for each participating node is $O(1)$. For the node that requests the ticket renewal, the computation complexity is $O(k)$, independent of the network size n. The communication consists of one broadcast and k unicasts, as illustrated in Figure 2.

5.1.2 Self-initialization and Share Update

After initializing the first k nodes with their polynomial shares, the trusted authority destroys $f(x)$ and quits. The initialized nodes then collaboratively initialize their neighboring nodes. Each node returns a "partial" share to the requesting node. By collecting k such partial shares and combining them together, the node is initialized with its valid share. Furthermore, we update the secret shares using our self-initialization mechanism in order to achieve scalability and efficiency. A coalition of k nodes first updates their shares by applying the existing protocols as proposed in [18]. The self-initialization protocols then follow to update the shares of the rest of the network. This mechanism strengthens the robustness of our system against model II conspired attack, as defined in Section 3.3. Detailed algorithms and security proof are available in the technical report [27].

5.2 Protocol Implementation

We implement the communication protocols in *ns-2*. The ticket renewal involves one-hop wireless communications between the node that requests the service and its neighbors. The request message is locally broadcast, and its neighbors respond with unicast replies. To reduce potential collisions in the reply messages, each neighboring node implements a simple backoff mechanism. The node generates a random backoff value in the range of $[0, m]$, measured in the time unit Δt that includes the transmission time of a reply message and propagation delay. m is the number of its neighbors. This way, the reply messages are spread out to avoid collisions. The node does not send explicit acknowledgment back to the reply message; the current IEEE 802.11 MAC already has it.

A node may not receive enough replies for several reasons. For example, the node may not have sufficient number of neighbors, or the replies are lost due to collisions or corruptions. To handle such cases, each node initiates its ticket renewal request starting at $T_{cert} - 3T_{det}$. The node accumulates all the valid reply messages during this triple time interval until its ticket expires. If the node does not receive enough replies during the initial $1.5T_{det}$, it roams to another area in anticipation of more neighbors.

The current misbehavior detection algorithm is mainly through overhearing the channel during the detection period T_{det}. A potential drawback is that it may consume much energy. A simple fix to this problem is via *statistical sampling*. Each node randomly samples the behavior of its neighboring node and sleeps during other time. Because the random sample pattern is not known to other nodes, malicious neighbors cannot avoid being sampled through the scheduled attacks.

6 Performance Evaluation

6.1 Computation Cost

Our cryptographic implementation on UNIX is written in C and currently consists of about 10,000 lines of code. It implements cryptographic building blocks for ticket services, self-initialization, and secret share update services, along with other supporting modules.

We use RSASSA-PSS signature scheme [34] to certify URSA tickets. We set the public exponent e as 65537 in all measurements[5]. Tables 1, 2 and 3 compare the computation costs of our ticket services with the standard RSA operations on three popular platforms with different computation power: (1) a Compaq iPAQ3670 Pocket PC with 206MHz Intel StrongARM CPU, 16M flash ROM, and 64M RAM, (2) a laptop with 300MHz Pentium II CPU and 256M RAM, and

[5]Low public exponent $e < 2^{16}$ is valid for RSA signature schemes but not encryption schemes [9].

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.15	0.29	0.39
1280	0.01	0.26	0.50	0.57
1536	0.01	0.41	0.79	0.88
1792	0.01	0.61	1.18	1.29
2048	0.01	0.85	1.71	1.79

Table 1: RSA and URSA ticket certification ($k = 5$, Pocket PC iPAQ3670, StrongARM 206MHz CPU)

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.07	0.16	0.18
1280	0.01	0.13	0.27	0.31
1536	0.01	0.21	0.46	0.51
1792	0.01	0.31	0.67	0.74
2048	0.01	0.44	1.01	1.08

Table 2: RSA and URSA ticket certification ($k = 5$, Laptop, PentiumII 300MHz CPU)

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.02	0.05	0.06
1280	0.01	0.04	0.10	0.11
1536	0.01	0.08	0.15	0.18
1792	0.01	0.11	0.24	0.26
2048	0.01	0.16	0.36	0.38

Table 3: RSA and URSA ticket certification performance ($k = 5$, Laptop, PentiumIII 850MHz CPU)

(3) a laptop with 850MHz Pentium III CPU and 256M RAM. All processing latency is measured at the granularity of 0.01 second. key denotes ticket-signing RSA key length in bits; RSA-PK denotes standard RSA's PK-verification; RSA-SK denotes standard RSA's SK-signature; URSA-PTC denotes partial ticket computation, that is, using secret shares to sign tickets; URSA-Combine denotes the delay caused by combining k partial tickets; URSA-PSS denotes partial secret share computation, i.e., Lagrange interpolation; URSA-Sum denotes obtaining a secret share by summing together all partial secret shares.

Our measurements show that computation power is a critical factor that affects the performance. The Pentium III laptop performs well in all test cases, while the Pocket PC consumes more time. However, in terms of the most heavy computation URSA-Combine, our Pocket PC takes less than 0.4 second when the RSA key length is set to 1K bits, and less than 1.8 seconds when 2K RSA key length is applied. If IEEE 802.11 wireless interface is adopted with around 200 meter single-hop transmission range, this delay is acceptable even for a mobile node moving at freeway speed, i.e., around 20 m/sec. We also observe from these tables that the standard RSA signature scheme is considerably faster. The reason is that a major optimization technique using the two prime factors of the RSA key is applied [34]. However, the URSA signature scheme still achieves comparable delay performance without such optimization.

In Table 4 we measure the computation cost for different values of the coalition size k. We find that parameter k does not affect the system performance significantly because (i) partial tickets are computed in parallel by the coalition members; and (ii) incremental of value k does not significantly increase the overhead in combining k partial tickets.

k	iPAQ3670, ARM 206MHz		Laptop, PIII 850MHz	
	URSA-PTC	URSA-Combine	URSA-PTC	URSA-Combine
2	0.290	0.397	0.059	0.063
3	0.291	0.391	0.057	0.062
5	0.293	0.394	0.057	0.062
7	0.291	0.393	0.056	0.062
10	0.292	0.392	0.058	0.061
20	0.291	0.393	0.059	0.060
30	0.291	0.396	0.056	0.063

Table 4: URSA ticket service performance vs. k (Average value on 10 runs, RSA key length 1024bit, time unit: second)

key (bit)	iPAQ3670,ARM 206MHz		Laptop, PIII 850MHz	
	URSA-PSS	URSA-Sum	URSA-PSS	URSA-Sum
1024	0.09	0.10	0.01	0.01
1280	0.10	0.11	0.01	0.01
1536	0.10	0.11	0.01	0.01
2048	0.10	0.11	0.01	0.01

Table 5: URSA self-initialization performance ($k = 5$, time unit: second)

The operations used in self-initialization and share update, i.e., multiplicative inverse and Lagrange interpolation, are relatively inexpensive to compute compared with URSA multi-signature generation. As shown in Table 5, processing latency incurred by self-initialization is not significantly affected by key length, and is much smaller than that of RSA and URSA ticket certification.

6.2 Communication Performance

We use simulations to evaluate the performance of URSA design in terms of the communication cost. We implement all URSA communication protocols on the application layer in the network simulator *ns-2* [30]. A UDP-like transport agent is developed for delivery of actual application data units (ADUs) and one-hop IP broadcast.

We experiment with networks of sizes ranging from 50 to 100 nodes. The average node moving speed varies from 1 to 15 m/sec and the routing protocol is DSR [24]. We use an improved mobility model [41] based on the random way-point model in *ns-2* distribution to emulate node mobility patterns, which guarantee the convergence of the average moving speed within the 1200 seconds of the total simulation time. Among those pairs of $[s_{min}, s_{max}]$[6] values given in [41] that converge, we choose the maximum range, i.e., $s_{max} - s_{min}$, so that it allows more randomness in speed setting. We set the expiration time of tickets at 300 seconds, so that the first round of ticket renewal happens after about

[6] s_{min} and s_{max} denote the minimum and the maximum node moving speeds respectively.

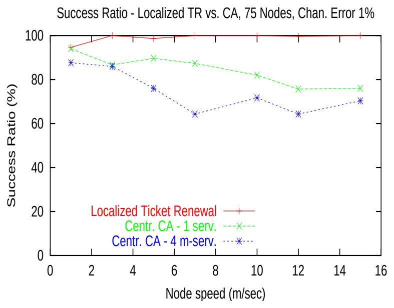

Figure 3: Success ratio vs. node speed, ticket renewal with 1% channel error rate

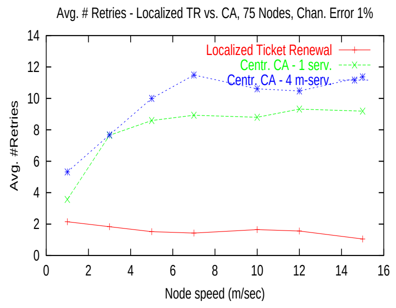

Figure 4: Avg. # of retries vs. node speed, ticket renewal with 1% channel error rate

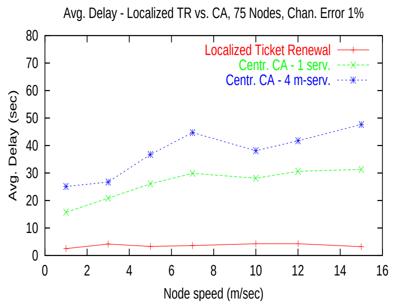

Figure 5: Avg. delay vs. node speed, ticket renewal with 1% channel error rate

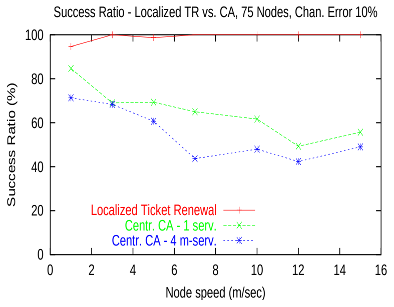

Figure 6: Success ratio vs. node speed, ticket renewal with 10% channel error rate

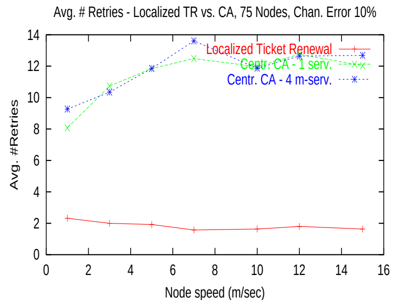

Figure 7: Avg. # of retries vs. node speed, ticket renewal with 10% channel error rate

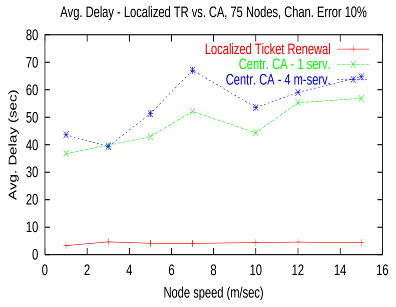

Figure 8: Avg. delay vs. node speed, ticket renewal with 10% channel error rate

300 seconds to give enough time for the node mobility pattern to stabilize. The coalition size is set as $k = 5$, based on typical node density of mobile ad-hoc networks.

We use the following metrics to evaluate the performance: *Success ratio* measures the ratio of the number of successful ticket renewals performed by all nodes, over the total number of renewals that should take place during the simulation time (4 per node). *Average number of retries* measures the number of retries before a node successfully receives the ticket service. *Average delay* measures the average latency to successfully renew a ticket. Finally, *Normalized overhead* measures the aggregate, over all nodes of the topology, communication overhead over the success ratio. It includes the total amount of traffic (in bytes) due to packets transmitted as part of the communication protocol that provides the localized ticket renewal service. We normalize the overhead in order to compensate for the fact that a node which fails in ticket renewal will be isolated and not able to issue ticket renewal requests for the remaining time of the simulation.

6.2.1 Ticket services

We first examine the performance of our localized ticket renewal service in a network of 75 mobile nodes. We study the scenarios of low (1%) and high (10%) wireless channel error rates and the average node moving speed ranging from 1 to 15 m/sec. We compare with the cases in which the network is served by one centralized authority (CA) server or multiple distributed CA servers. In the one CA case, the CA server is placed statically in the middle of the network topology and renews tickets for all the networking nodes. In the distributed CA case, four mobile CA servers are deployed in the network, following the same mobility model. In the latter scenario, a client node selects the servers in a round-robin fashion, since we do not assume an on-line location service.

Comparing the low wireless channel error scenarios (Figures 3,4,5) with high wireless channel error scenarios (Figures 6,7,8), the performance of both single CA server and distributed CA servers significantly degrades as the wireless channel error rate increases. In contrast, due to the localized communication pattern, our ticket renewal service performs almost the same with respect to all three measurement metrics, as channel error rate increases from 1% to 10%.

Figures 3 and 6 show that the centralized solutions have lower success ratio than our localized ticket renewal, which is almost 100%. As the node speed increases from 1 m/sec to 15 m/sec, we observe that the success ratio almost remains unchanged for URSA at speeds above 5 m/sec. Higher mobility speed increases the likelihood that a node locates k neighboring nodes for ticket services. This effect offsets the performance impact of routing dynamics under high mobility.

Figures 4,7 and Figures 5,8 further demonstrate our service availability in terms of the average number of retries and average delay, respectively. The average number of retries decreases and the average delay of our ticket renewal remains nearly constant as the node mobility speed increases. For comparison, we show the average number of retries and average delay of the ticket renewal that are implemented by a single centralized CA server and four CA servers. They both incur much higher average number of retries, and average delays under both channel error rates tend to grow as the node mobility speed increases.

An interesting observation from the above comparison is that the centralized approach with the single static CA server outperforms that of the multiple mobile CA servers. This shows the undesirable effect of mobility in the centralized approaches. However, it is worth noting that placing a server optimally in a mobile network would be difficult as the centroid of the network moves over time.

6.2.2 Scalability and Communication Overhead

In Figures 9, 10, and 11, we show the success ratio, average delay and normalized overhead, respectively, as the network size increases from 50 to 100 nodes. The average moving speed is set to 3 m/sec in all scenarios. The localized ticket renewal service exhibits both higher success ratio and lower average delay than the centralized approaches. In these figures we also show the case of four CA servers being static and optimally placed in the square topology ("Centr. CA - 4 serv."), which clearly outperforms both the single server and the multiple mobile servers ("Centr. CA - 4 m-serv.") approaches. However, even though the multiple static servers have a success ratio comparable to our localized approach, their delay is considerably higher.

The normalized communication overhead for the localized ticket service is higher than that of the centralized CA approaches, as shown in Figure 11, at the relatively low moving speed of 3m/sec. This is due to the fact that for every ticket renewal request, k unicast partial tickets have to be sent back to the requester, in order for the latter to reconstruct the complete ticket. There is a fundamental trade-off between communication overhead and resilience of the ticket renewal service against attacks, and design parameter k can be used to tune the system towards either communication efficiency or service resilience.

However, the normalized communication overhead becomes comparable for the localized ticket renewal solution and the CA approaches, at higher mobility speeds (see Figure 12). At high moving speeds, it is much more difficult for the centralized CA server to establish communication with the mobile clients. Consequently, more attempts are needed in order for the CA server to update the tickets and successfully send them back to the clients, which contributes to higher communication overhead. This fact is also denoted by the "jump" in the results of Figure 12, for the centralized CA cases of 1 and 4 servers. On the contrary, communication overhead of our localized ticket renewal service slightly decreases, since it becomes more likely for the requesting node to find k neighbors that will renew its ticket.

Moreover, as the channel error increases to 10%, as shown in Figure 13, the normalized communication overhead of the localized ticket service becomes even lower than that of the centralized approaches. Because only localized communication is involved, our ticket service results in fewer attempts for a successful renewal (shown in Figure 7), and consequently less communication overhead. This explains the gap in the results of the localized proposal and the centralized approaches in Figure 13. As a concluding remark, from Figures 12 and 13, we note that communication overhead of our localized ticket renewal service remains constant as mobility or channel error increases.

6.2.3 Resilience to Attacks

In order to demonstrate resilience in the presence of attackers, we place a simple "DoS" attacker in the simulation scenario. The attacking node continuously blasts CBR traffic of 1.2Mbps over the wireless channel with 2Mbps raw bandwidth. In the case of the centralized approaches, it is placed in the vicinity of the CA server within one hop, while in simulating our localized ticket services it freely roams as the other legitimate nodes do. The average moving speed is set to 5 m/sec.

Figure 14 plots the success ratio as the network size increases from 50 to 110 nodes. It is obvious from the figure that, while our localized ticket approach still achieves high success ratio close to 100%, the centralized CA approach has a sharp decrease in its success ratio as the network size grows beyond 80 nodes. This is due to the fact that, more networking nodes result in higher aggregate ticket renewal traffic. Together with the attack traffic the total offered traffic volume goes beyond the threshold that the underlying IEEE 802.11 MAC can coordinate in the vicinity of centralized CA servers. From Figure 15 we can also see how the overhead of both approaches grows as the number of nodes in the network increases. The aggregate overhead of the localized ticket service increases linearly with the network size, while the centralized approach has super-linear growth: it is almost doubled as the network size increases from 90 to 110 nodes.

6.2.4 Self-initialization and share update

The performance of self-initialization process and the share update is studied in this section. Figure 16 presents the average latency for each node in the self-initialization process. In each experiment that lasts for 900 seconds, we assume that the first $2k$ nodes of the topology are initialized by a dealer,

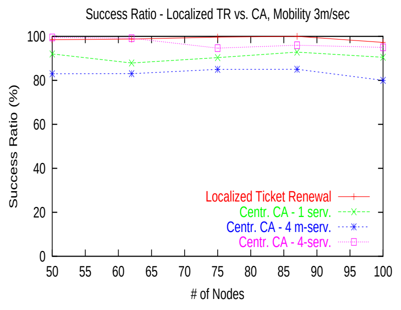

Figure 9: Success ratio vs. # of nodes

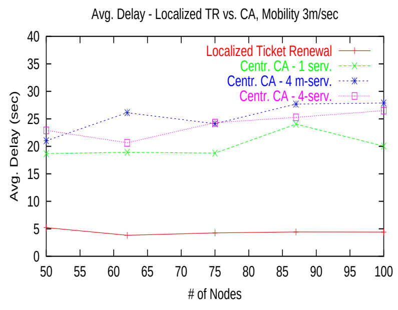

Figure 10: Avg. delay vs. # of nodes

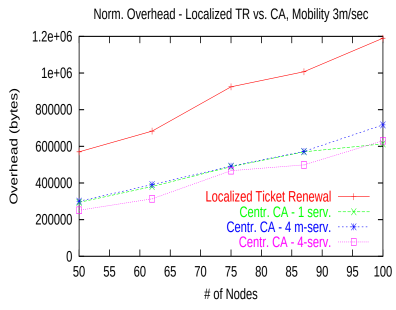

Figure 11: Normalized overhead vs. # of nodes

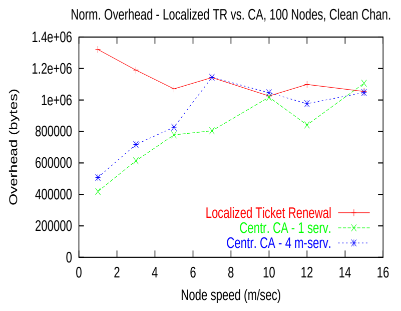

Figure 12: Normalized overhead vs. node speed, clean channel

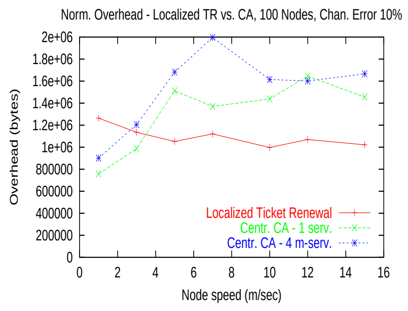

Figure 13: Normalized overhead vs. node speed, 10% channel error rate

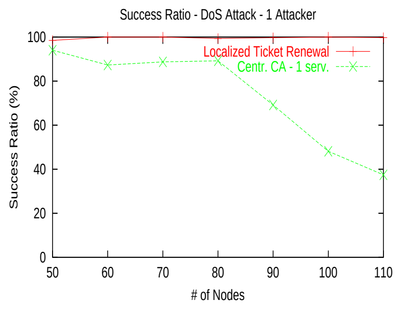

Figure 14: Success ratio vs. # of nodes with 1 attacker. Avg. mobility speed 5 m/sec

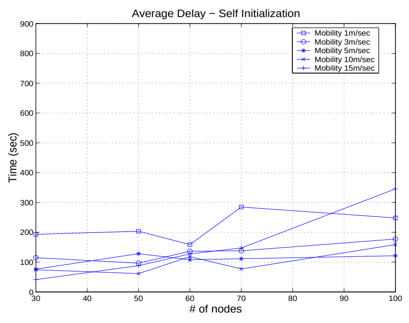

Figure 16: Self-initialization: avg. delay vs. node speed

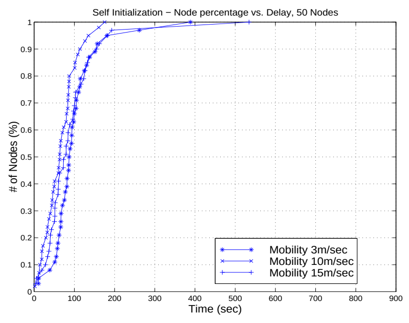

Figure 17: Share update: node percentage vs. delay, 50 nodes

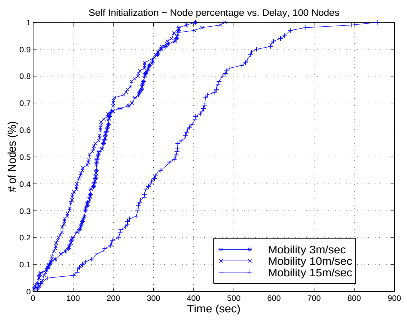

Figure 18: Share update: node percentage vs. delay, 100 nodes

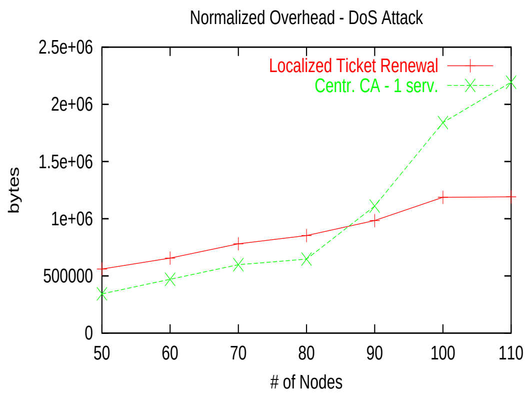

Figure 15: Normalized overhead vs. # of nodes with 1 attacker. Avg. mobility speed 5 m/sec

and other nodes are initialized through self-initialization. From Figure 16 we see that the self-initialization design scales well to network size and node mobility; even for the largest topology of 100 nodes and node speed at 20 m/sec, all nodes manage to be initialized in less than 500 seconds.

In Figures 17 and 18, we present the progress of the secret share update in network topologies with 50 and 100 nodes, respectively, and for three speeds (3, 10 and 15 m/sec, corresponding to low, medium and high mobility). For the case of 50 nodes, it takes almost 50 seconds for the first 20% of the network to update. However, as more and more nodes receive their shares, the convergence is accelerated: 80% of the network is initialized in another 50 seconds. We also note that the evolution of the algorithm is similar at all mobility speeds; this shows that our design is robust to mobility. In the scenario of 100 nodes, the curves are shifted to the right, since the network is larger and it takes more time for the "initialization wave" to be propagated to the whole network.

7 Discussion

Misbehavior Detection in Mobile Ad-Hoc Networks URSA works with other (local) misbehavior detection mechanisms. Due to the lack of an infrastructure and a clear line of defense, network-oriented, global intrusion detection techniques are not applicable in a mobile ad-hoc environment. URSA only requires that a networking node monitor and detect its neighboring nodes' misbehaviors.

Some recent research efforts demonstrate the feasibility and effectiveness of a number of local detection and monitoring techniques. In Watchdog [28], each node on the data forwarding path turns its wireless interface into promiscuous mode, and monitors its downstream node's transmission by passively listening to the channel. If a node does not forward the packet, it will be detected by its upstream node and classified as being misbehaving. In [43, 44], each node matches its own movement and the corresponding changes in its own routing table against a trained classifier to detect anomaly. Kyasanur et al. [25] devised a mechanism to protect the IEEE 802.11 MAC contention backoff algorithm. A receiver measures the sender's backoff period between RTS attempts. At the same time, other nodes within the transmission range observe the behaviors of both the sender and the receiver to identify potential misbehaviors. Hu et al. studied the defense against misbehaviors on proactive DSDV and on-demand DSR routing protocols in [20, 21]. Per-hop hashing is applied so that corrupted routing messages can be immediately detected. Similar idea is adopted in [42] where digital signature is used.

Due to the diversity of potential misbehaviors and the diversity of the detection techniques, we do not advocate any specific local detection algorithm. The detection algorithm serves as a plug-in component; a better local detection algorithm can be readily inserted and work with URSA. Note that an autonomous node has enough incentive to pay the cost to monitor its neighbors, since it relies on their proper collaboration to relay packets to and from the rest of the network.

In fact, our collaborative consensus can also help to reduce the detection inaccuracy. For example, if the false positive probability (i.e., the case when a well-behaving node is detected as misbehaving) of node i is P_i where $0 < P_i \ll 1$, then it is easy to see that the misdetection probability by k collaborative nodes reduces to be $\Pi_{i=1}^{k} P_i$. On the other hand, the false negative probability (i.e., the case when a misbehaving node is undetected) slightly increases but is comparatively the same, since the joint detection probability is roughly $1 - \Pi_{i=1}^{k}(1 - P_i^{'}) \approx \sum_{i=1}^{k} P_i^{'}$ when each node's false negative detection probability $P_i^{'}$ is small[7]. This simple analysis shows that our collaborative design can alleviate the effect of misdetection by an individual node.

Soft states revisited We have used soft states extensively in URSA design. The first benefit is that it handles dynamic membership well. The management overhead associated with node join and leave is greatly reduced. Moreover, it is also used to resist attacks. As discussed in Section 4.5, the proper choice of T_{cert} can limit the possible collaboration among misbehaving nodes, as they cannot quickly move together and start conspired attacks. Regularly refreshing the secret shares also reduces the possibility of exposing such sensitive information to malicious nodes. However, frequent refreshing also incurs much communication overhead. Therefore, the soft states used in this paper provide possibly flexible tradeoff between security strength and communication overhead.

Setting Parameter k Our design so far assumes that each node has at least k trusted neighbors. For simplicity, k also

[7]The fact is that, the chance of a misbehaving node being undetected is much smaller than the case a well-behaving node is misdetected.

defines the power of the attackers that our design is able to handle. However, in the case when $m < k$ attackers exist in the local neighborhood, our design requires that the neighborhood should have at least $m + k$ nodes to ensure the functioning of the URSA certification service. If the number of attackers $m > k$, as long as they do not exist in the same local neighborhood and do not conspire with one another, our design still works. In some sense, we have to limit the scope of collaboration among attackers to make the problem still solvable.

8 Conclusions

The lack of a clear line of defense and the lack of infrastructure support pose the major challenges for access control in a mobile ad-hoc network that is composed of autonomous nodes. The wireless bandwidth constraint, the dynamic node membership and/or failures, and the multihop routing unreliability due to node mobility, further increase the design difficulty. In this paper, we present URSA, a novel solution to ubiquitous and robust access control in mobile ad-hoc networks. In URSA, only well-behaving nodes are granted access to routing and packet forwarding via valid tickets issued collectively by multiple local nodes. Our design has been motivated by the principle that the access control decision has to be fully *distributed* and *localized* in order to operate in a large-scale, dynamic mobile ad-hoc network. We seek to maximize the service availability in each network locality, which is also crucial to supporting mobile users. Our experiences in both implementation and simulations have shown the effectiveness of our design.

9 Acknowledgment

We greatly appreciate all the constructive comments by our shepard, Dr. Klara Nahrstedt and the reviewers.

References

[1] G. Appenzeller, M. Roussopoulos, and M. Baker. User-friendly Access Control for Public Network Ports. In *IEEE INFOCOM*, 1999.

[2] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure Pebblenets. In *ACM MOBIHOC*, pages 156–163, 2001.

[3] N. Borisov, I. Goldberg, and D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *ACM MOBICOM*, 2001.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *ACM MOBICOM*, pages 85–97, 1998.

[5] L. Buttyan and J. P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[6] R. Canetti, S. Halevi, and A. Herzberg. Maintaining Authenticated Communication in the Presence of Break-Ins. *Journal of Cryptology*, 13(1):61–105, 2000.

[7] S. Capkun, J. P. Hubaux, and L. Buttyan. Mobility Helps Security in Ad Hoc Networks. In *ACM MOBIHOC*, 2003.

[8] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.

[9] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[10] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Symposium on Foundations of Computer Science (FOCS)*, pages 427–437, 1987.

[11] Y. Frankel and Y. Desmedt. Parallel Reliable Threshold Multi-signature. Technical Report TR-92-04-02, Dept. of EECS, University of Wisconsin-Milwaukee, 1992.

[12] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In *Symposium on Foundations of Computer Science (FOCS)*, pages 384–393, 1997.

[13] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. In *CRYPTO*, pages 440–454, 1997.

[14] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In *CRYPTO*, pages 157–172, 1996.

[15] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks. `http://www.ietf.org/rfc/rfc2764.txt`, 2000.

[16] L. Gong. Increasing Availability and Security of an Authentication Service. *IEEE Journal of Selected Areas on Communications*, 11(5), 1993.

[17] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, 2000.

[18] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing or: How to Cope with Perpetual Leakage. extended abstract, IBM T.J. Watson Research Center, November 1995.

[19] P. Honeyman. Workstation Authorization. `http://www.citi.umich.edu/u/honey/talks/insite/`, 1997.

[20] Y.-C. Hu, D. B. Johnson, and A. Perrig. Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, 2002.

[21] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *ACM MOBICOM*, 2002.

[22] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *IEEE INFOCOM*, 2003.

[23] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE standard 802.11, 1997.

[24] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[25] P. Kyasanur and N. H. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. Technical report, University of Illinois, August 2002.

[26] J. Li, C. Blake, D. D. Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *ACM MOBICOM*, pages 61–69, 2001.

[27] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000.

[28] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *ACM MOBICOM*, 2000.

[29] E. A. Napjus. NetBar - Carnegie Mellon's Solution to Authenticated Access for Mobile Machines. `http://www.net.cmu.edu/docs/arch/netbar.html`.

[30] ns-2 (The Network Simulator). `http://www.isi.edu/nsnam/ns/`.

[31] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad Hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 2002.

[32] R. Perlman. An Overview of PKI Trust Models. *IEEE Network*, 13(6):38–43, 1999.

[33] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, 5(2):2–13, 2002.

[34] RSA Security Inc. PKCS #1 - RSA Cryptography Standard. `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/`.

[35] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *10th International Conference on Network Protocols (ICNP'02)*, 2002.

[36] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[37] V. Shoup. Practical Threshold Signatures. In *EUROCRYPT*, 2000.

[38] D. L. Wasley. Authenticating Aperiodic Connections to the Campus Network. `http://www.ucop.edu/irc/wp/wp_Reports/wpr005/wpr005_Wasley.html`, 1996.

[39] T. Wu, M. Malkin, and D. Boneh. Building Intrusion Tolerant Applications. In *Eighth USENIX Security Symposium (Security '99)*, pages 79–91, 1999.

[40] H. Yang, X. Meng, and S. Lu. Self-Organized Network Layer Security in Mobile Ad Hoc Networks. In *First ACM Workshop on Wireless Security (WiSe)*, 2002.

[41] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *IEEE INFOCOM*, 2003.

[42] M. G. Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *First ACM Workshop on Wireless Security (WiSe)*, 2002.

[43] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *ACM MOBICOM*, 2000.

[44] Y. Zhang, W. Lee, and Y.-A. Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *ACM Mobile Networks and Applications (MONET) Journal*, 2002.

[45] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, 1999.

[46] L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A Secure Distributed On-line Certification Authority. *ACM Transactions on Computer Systems*, 20(4):329–368, November 2002.

[47] P. Zimmermann. *The Official PGP User's Guide*. The MIT Press, 1995.