An Authentication Framework for Hierarchical Ad Hoc Sensor Networks

Mathias Bohge *
Wireless Information Network Laboratory
(WINLAB)
Rutgers, The State University of New Jersey
73 Brett Rd.
Piscataway, NJ 08854
Mathias@Bohge.de

Wade Trappe
Wireless Information Network Laboratory
(WINLAB)
Rutgers, The State University of New Jersey
73 Brett Rd.
Piscataway, NJ 08854
trappe@winlab.rutgers.edu

ABSTRACT

Recent results indicate scalability problems for flat ad hoc networks. To address the issue of scalability, self-organizing hierarchical ad hoc architectures are being investigated. In this paper, we explore the task of providing data and entity authentication for hierarchical ad hoc sensor networks. Our sensor network consists of three tiers of devices with varying levels of computational and communication capabilities. Our lowest tier consists of compute-constrained sensors that are unable to perform public key cryptography. To address this resource constraint, we present a new type of certificate, called a TESLA certificate, that can be used by low-powered nodes to perform entity authentication. Our framework authenticates incoming nodes, maintains trust relationships during topology changes through an efficient handoff scheme, and provides data origin authentication for sensor data. Further, our framework assigns authentication tasks to nodes according to their computational resources, with resource-abundant access points performing digital signatures and maintaining most of the security parameters. We conclude by providing an initial performance evaluation and security analysis for our framework.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: [distributed networks, network communications]

General Terms

Security

*The author is currently with the Telecommunication Networks Group at the Technical University of Berlin, Sekr FT5-2, Einsteinufer 25, 10587 Berlin, Germany.

WiSE'03, September 19, 2003, San Diego, California, USA.

Keywords

Authentication, Ad hoc networks, TESLA, Handoff

1. INTRODUCTION

Remote sensing applications are becoming an increasingly important area for research and development due to the critical need for applications that will perform environmental monitoring, provide security assurance, assist in healthcare services and facilitate factory automation. In remote sensing scenarios, one or more applications are connected to a sensor network through a communication network. The sensors in the sensor network make measurements, such as local temperature or barometric pressure, and communicate this data with the appropriate application via the network. Providing security mechanisms for sensor networks is of critical importance since sensors will ultimately be used to assist in our daily lives. The authentication of the data source as well as the data are critical concerns since adversaries might attempt to capture sensors and tamper with sensor data. Traditional authentication frameworks based on public key cryptography are not suitable for sensor networks since the sensor network will ultimately consist of small, low-powered devices that are mobile. The limited computational and storage resources available to sensors necessitates alternatives to authentication based on public key certificates.

Recently, a set of *security protocols for sensor networks*, known as SPINS, has been proposed [1]. SPINS addresses authentication on limited resource sensor networks by introducing two security protocols that rely on the presence of a more powerful basestation and an initial shared secret between the basestation and each participating sensor node: SNEP and μTESLA. SNEP is a simple protocol that provides data confidentiality, two-party data authentication, and evidence of data freshness using only symmetric keys and counters. μTESLA is a modified version of the TESLA protocol, which performs bootstrapping without using a public key infrastructure (PKI) and discloses one key each *epoch* independently of the packet rate to provide broadcast authentication. Another work that focused on authentication for ad hoc networks was presented in [2]. In this paper, a distributed light-weight model for authentication was presented that involves network nodes requesting trust references from neighboring nodes in order to establish the trust relationships needed for network authentication.

Each entity maintains a list of trusted entities, and using these lists trusted communication paths between two arbitrary entities can be derived. One drawback of this method, however, is its scalability. For large networks, the size of the trust tables can become prohibitive. Another work on authentication for ad hoc networks that addressed the issue of scalability was presented in [3], which introduced the use of cluster heads to reduce the amount of control packets needed. In this work, the network is divided into cluster regions, and cluster heads are elected from the regular network nodes within each cluster. Authentication is provided by using a public key infrastructure that, unfortunately, is not suitable for small sensor devices.

These methods focus on ad hoc networks employing a flat topology. However, ad hoc networks have been recently shown to have capacity limitations, and one approach to address this drawback is to employ a hierarchical ad hoc network. In this paper we will further explore the advantages of hierarchical ad hoc networks, particularly focusing on the advantages of the hierarchical ad hoc sensor network for performing authentication when compared with flat ad hoc networks. Authentication in hierarchical ad hoc networks has been essentially untouched, and we are aware of only one work in this direction [4, 5], which focused on a military environment. The security of their work is based largely on the assumption that the access points, which corresponded to unmanned aerial vehicles, are unable to be compromised. This is an assumption that does not hold in non-military applications, and therefore we consider a three-tier hierarchical ad hoc network that is suitable for more general remote sensing applications running on the Internet. We develop an authentication framework for our three-tier hierarchical sensor network that addresses the hardware resources of the three-tier network, and employs cryptographic primitives that are appropriate for each type of node.

1.1 Hierarchical Sensor Network

Mobile ad hoc networking is the ideal architecture for the wireless sensor network since ad hoc networking provides a ubiquitous communication infrastructure capable of growing and adjusting to sensor dynamics. However, despite the popularity of flat ad hoc networks for sensor applications, recent information theoretic studies have indicated the limitations of the flat topology [6].

Recently, hierarchical ad hoc networks have been proposed as an alternative topology to flat ad hoc topologies. Initial measurements indicate that the hierarchical approach has better performance than flat ad hoc network [7, 8]. In [8], a three-tier self-organizing hierarchical ad hoc network is proposed to improve the scalability of ad hoc wireless networks. A modified hierarchical dynamic source routing (DSR) protocol [9] is studied. In particular, they observed, when using the same amount of sensor nodes in a given coverage area for flat and hierarchical topologies, that the system throughput capacity increases, while system delay decreases. The main reason for these improvements is the reduced number of hops since most sensor data are destined for the Internet, which is reachable in a few hops in the hierarchical approach.

In this paper, we will use the three-tier ad hoc network topology of [8]. This architecture, depicted in Fig. 1, consists of three classes of wireless devices: (B) high-power access points that route packets received via radio links to the wired infrastructure, (C) mobile medium-powered forwarding nodes that relay information from sensor nodes to access points, and (D) low-powered mobile sensor nodes that have limited computing capability. We have depicted an Internet-based application (A) that is connected to the sensor network through the access points. This network is ideal for sensor-driven applications, where traffic flows from the sensors to Internet-based applications.

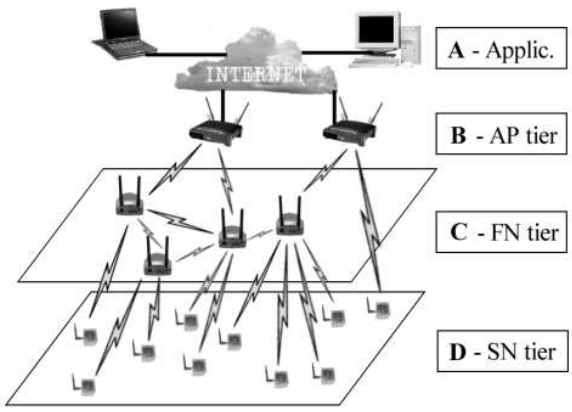

Figure 1: Three-tier hierarchical ad hoc sensor network.

There are several key points that differentiate the three-tier hierarchical sensor network from conventional ad hoc sensor networks:

1. *Varying levels of computational power within the sensor network*: Conventional ad hoc sensor networks assume all nodes are created equal. The presence of large numbers of unreliable and energy-constrained sensors makes the task of energy-efficient communication and security protocols essential to the operation of a flat sensor network. However, the three-tier sensor network consists of three types of devices with different degrees of computational capabilities.

2. *Sensors do not communicate with each other*: The purpose behind a remote sensor network is to feed data to the application, which will make decisions based on the observations it receives. Sensor nodes route their packets via higher-level nodes and it is therefore unnecessary for sensor nodes to communicate and authenticate each other.

3. *The forwarding node is a radio-relay*: The purpose of the forwarding node is to relay messages from the sensors to the access points. Forwarding nodes have two wireless interfaces, one that communicates with SNs, and one that communicates with APs. They do not necessarily perform measurements themselves.

2. TESLA AND TESLA CERTIFICATES

Today, the most widely used certification systems are PGP [10] and X.509 [11]. Both rely on public key cryptography, which makes them unsuitable for devices that are low-powered, or computationally-constrained. These devices should not have to verify an RSA-signature associated with a public key certificate. Therefore, if we wish to have a certificate-based authentication system for low-powered devices, we need a certificate structure that does not employ public key cryptography.

TESLA [12] is a broadcast authentication technique that

achieves asymmetric properties, despite using purely symmetric cryptographic functions (namely MACs [13]) and thus enables low-powered nodes to perform source authentication. We now briefly review TESLA. TESLA divides time into intervals of equal duration. Time slot n is assigned a corresponding key tK_n. For each packet generated during time interval n, the sender appends a MAC that is created using the secret key tK_n. Each receiver buffers the packets, without being able to authenticate them, until the sender discloses the key tK_n by broadcasting the corresponding key-seed s_n. Once s_n is disclosed, anyone with s_n can calculate tK_n and can pretend to be the sender by forging MACs. Thus, the use of tK_n for creating MACs is limited to time interval n, and future time intervals use future keys. Further, s_n isn't disclosed until d time slots later, where d is governed by an estimate of the maximum network delay for all recipients.

The keys tK_n are derived from s_n using a publicly available one-way function F'. The s_n are related to each other via a reverse-time chain of one-way functions. To create the chain of key-seeds, the sender chooses a terminal seed s_l, and generates s_{l-1} using a one-way function F. The remaining seeds $\{s_0, s_1, \cdots, s_l\}$ are derived via $s_l \xrightarrow{F} s_{l-1} \xrightarrow{F} s_{l-2} \xrightarrow{F} ... \xrightarrow{F} s_1 \xrightarrow{F} s_0$. The sender uses the seed-chain in the opposite direction (starting with seed s_0) to derive the TESLA keys by applying the one-way function F' via $s_n \xrightarrow{F'} tK_n$.

When a user receives a packet, he first checks whether the packet is *fresh* (i.e. it was sent in a timeslot whose TESLA-key hasn't been disclosed) or dated. The receiver discards all dated packets and buffers only the fresh ones. Once the user receives a TESLA-seed s_n, he checks $F(s_n) = s_{n-1}$ to be sure of s_n's authenticity. He derives tK_n by $tK_n = F'(s_n)$, and authenticates the packets that were sent in timeslot n.

The framework that we propose in this paper uses TESLA for authentication of data broadcast by the application. Additionally, TESLA is used to create TESLA certificates that support entity authentication in the sensor node handoff mechanism (cf. Section 7.2). In contrast to the original TESLA, our TESLA clients (i.e. all network nodes besides the forwarding nodes) are not bootstrapped using the public key infrastructure, but by the application sending the initial TESLA key to each network node encrypted with the appropriate shared key.

2.1 TESLA Certs

In Fig. 2, we present the entities involved in TESLA certificates, as well as the steps involved in using TESLA certificates. Much like conventional public key certificates, we have a certificate authority (CA), who is responsible for creating the certificates for entity B. A low-powered device, depicted by D, will contact B to use B's service.

The steps involved in TESLA certificates are:

1. The CA periodically issues TESLA certificates for B. During time slot n, the certificate authority (CA) doesn't sign the TESLA-certificate with its private key, but uses the non-disclosed TESLA key tK_{CA_n} to create a MAC that is included in the certificate. B's public key is replaced its authentication key aK_{B_n}, which is encrypted by the CA using the TESLA key tK_{CA_n}.

$$Cert_{CA_n}(B) = (ID_B, \{aK_{B_n}\}_{tK_{CA_n}}, TS_A, MAC_{tK_{CA_n}}(...))$$

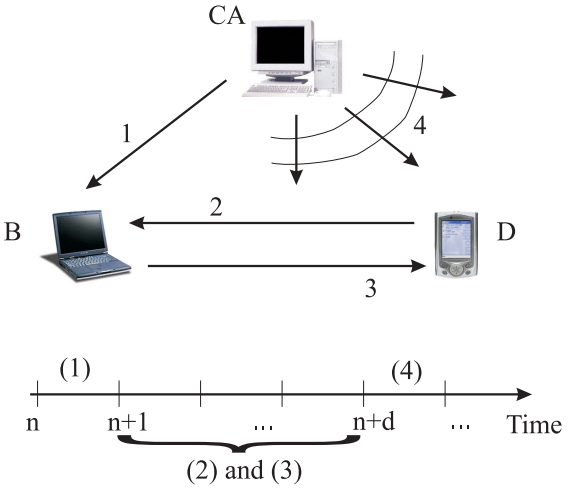

Figure 2: The steps involved in using TESLA certificates.

TS_A is a timestamp addressing the certificate's expiration date. The certificate is sent to B along with the matching authentication key aK_{B_n}:

$$CA \rightarrow B : (Cert_{CA_n}(B), \{aK_{B_n}\}_{K_{CA,B}}, MAC_{K_{CA,B}}(...)).$$

2. Sometime between time n and $n+d$, D contacts B requesting to use B's service, $D \rightarrow B : (request)$.

3. Following the request in step 2, B must prove its identity to node D. B sends an authentication packet, which consists of the TESLA certificate and a MAC that was created using B's authentication key aK_{B_n}:

$$B \rightarrow D : (Cert_{CA_n}(B), MAC_{aK_{B_n}}(request)).$$

Upon receiving the authentication packet, D measures the freshness of the certificate by checking the timestamp of $Cert_{CA_n}(B)$ to make sure that it arrived before time $n+d$, when the CA announces tK_{CA_n}. If $Cert_{CA_n}(B)$ is fresh, D buffers the authentication packet.

4. The CA discloses its TESLA key tK_{CA_n} at time $n+d$. Upon receiving tK_{CA_n}, D checks the authenticity of the TESLA certificate by checking $MAC_{tK_{CA_n}}$, then it decrypts B's authentication key aK_{B_n} and checks $MAC_{aK_{B_n}}$. User D is able to certify the identity of B as long as it receives the TESLA certificate $Cert_{CA_n}(B)$ before the CA revealed the TESLA key TK_{CA_n}.

The lifetime of a TESLA-certificate is short. It depends on the disclosure time of the TESLA key that the certificate authority used when creating the MAC and encrypting the subject's authentication key. Choosing a key that will be disclosed soon lowers the delay in the authentication process at node D, but results in increased overhead when issuing new certificates.

3. OVERVIEW OF THE AUTHENTICATION FRAMEWORK

There are two primary goals for the framework: first, to ensure that the data received by the application (A) is sent by an approved sensor node (D); second, to verify that the

data hasn't been modified on its way to the application. To achieve these goals, an authentication service has to be realized that authenticates incoming nodes, establishes shared secrets among them and with the application, keeps track of changes in the network topology and provides data origin authentication for sensor node data.

As in every authentication service, the proposed framework relies on the presence of initial trust. It is necessary to get some trustworthy information about an incoming node before it is allowed to join the network. Information that is provided by the node is not trustworthy if it is not confirmed by a trusted entity. Therefore, each node that wants to join the network must have a personal *initial certificate* ($iCert$) that is issued by the network's *trusted third party* (TTP). The TTP is nothing more than a reliable node that is able to perform RSA signatures, whose public key is known to all nodes of the network that are able to verify RSA-signatures.

When an access point B or a sensor node D wants to join the network, the node presents its $iCert$, which eventually will be checked by the application A (because of their role as a radio relay, we don't consider forwarding nodes here– their role in the network includes authentication upon request). If the $iCert$ is valid, A will establish a shared secret $K_{A,B}$ (with B) or $K_{A,D}$ (with D), which will enable the new node to access the network and communicate with the application. Once the node is part of the network, its $iCert$ becomes less important. As long as B or D doesn't switch applications, the trust relationship with the application will last. Instead, it needs a method to authenticate itself with other network entities that are not the application, e.g. an access point must be able to authenticate itself with a sensor node and vice versa. The topology of the ad hoc network may change frequently, and thus it is desirable for nodes to perform this inter-node authentication on their own, in a fast and flexible manner. The application A enables them to do so by periodically issuing *runtime certificates* ($cert$) for each access point and sensor node of the network. Since the computationally weak sensor nodes are included in this process, these *certs* can't rely on RSA signatures. As mentioned in Section 2.1 a new type of certificate will be used.

Having this network of authenticated nodes and shared secrets, we can use the trust relationships to provide a data origin authentication service. A sensor node D that wants to deliver data, creates a MAC using the secret key it shares with the application and appends it to the data. The application will use the MAC to verify the data's origin. D creates another MAC using the secret key it shares with its gateway access point B, which provides access to the Internet. B will use the MAC to perform access control, by making sure that D is a valid part of the network before forwarding D's data. In case one or more forwarding nodes are located between B and D, B will answer the challenge that D includes in the data-packet to assure that the data finally reaches B. While the challenge-response mechanism can guarantee that the data arrives at the access point, it is neither able to tell what happened to lost data nor to provide information about who dropped it. Since this method is unable to identify the entities involved in the data delivery, we refer to this mode of data delivery the *weak mode* of operation. However, if a sensor node wants to send sensitive data or, for any other reason, wants each node along the path to be authenticated before it actually starts sending the data, it can choose to use the *assured mode* of data delivery, which will provide authenticated information about those nodes at the cost of an overhead of message exchange and shared secrets.

In building our authentication framework, we assume each forwarding node and access point has an RSA-key-pair along with its certificate. We also assume that they know the TTP's public key $+K_{TTP}$. For reference, at the end of the paper, we provide a summary of the notation we use in our framework.

4. CERTIFICATES

Certificates are the major tool to build an entity authentication service since they enable entities that don't share a secret key to establish a trust relationship. Our framework distinguishes between initial and runtime certificates. While each node needs an initial certificate to join the network, the runtime certificates are periodically issued by the application to during the network's lifetime. The forwarding nodes are an exception. A forwarding node needs a general certificate that is only used once a sensor node requests *assured service* (cf. Algorithm 5). In this case, an access point B checks the forwarding node's certificate. Therefore, certificates based on RSA can be used for the forwarding node's certificate.

4.1 Initial Certs

The framework relies on certificates as a means of *initial* trust. Each access point or sensor node that wants to join the network must own a certificate ($iCert_{TTP}$) issued by the network's trusted third party.

4.1.1 Access Point

We assume that the access point B is a device of high computational power and battery resources. This enables B to validate and perform RSA-signatures. Therefore, each access point has an RSA public and private key pair $(+K_A, -K_A)$ and an $X.509$-certificate ($iCert_{TTP}$) issued by the Trusted Third Party TTP, binding this keypair to their identity.

4.1.2 Sensor Node

The sensor node D applies to the trusted third party for its initial certificate ($iCert_{TTP}$). This initial certificate is tied to a certain application A that the sensor node plans to connect to.

$$iCert_{TTP}(D) = \\ (ID_D, \{iK_D\}_{+K_A}, TS_{TTP}, SIGN_{-K_{TTP}}(...))$$

The TTP issues this $iCert$ to the sensor node D along with the unique key iK_D. D uses this key to authenticate itself against application A. Since D's initialization key is encrypted with A's public key, only A is able to obtain the key from the certificate and proof A's authenticity. If D plans to connect to several applications D, it can apply for more than one certificate.

4.2 Runtime Certs

The purpose of runtime certificates is to maintain authenticity between the initial authenticated nodes during the networks's lifetime. As a result of the forwarding node and sensor node mobility, shared keys become obsolete and new keys have to be established. The runtime certificates use the trust relationships between the application and the nodes of the network to create new trust relationships among them.

4.2.1 Access Points

An access point runtime certificate must be readable by each sensor node D. Therefore, RSA-based certificates cannot be used. Instead, we will fall back on TESLA certs:

$$Cert_{A_n}(B) = (ID_B, \{aK_{B_n}\}_{tK_{A_n}}, TS_A, MAC_{tK_{A_n}}(...)),$$

where aK_{B_n} is the access point's authentication key for slot n. B will use this certificate to authenticate itself with a sensor node during handoff. The $MAC_{tK_{A_n}}(...)$ proves that this certificate was issued by the application.

4.2.2 Sensor Nodes

The sensor node runtime certificates will be used by an access point B to check D's identity during handoff to establish a new shared secret. It contains D's ID, a timestamp TS_A and D's secret authentication key $aK_{AP,D}$ encrypted with an 'access point group key' gK_{AP}, that every access point of the network gets during its authentication with the application.

$$Cert_A(D) = (ID_D, \{aK_{AP,D}\}_{gK_{AP}}, TS_A, SIGN_{-K_A}(...))$$

The signature $SIGN_{-K_A}(...)$ proves that this certificate is issued by the application.

5. CERTIFICATE RENEWAL

During the lifetime of a network, trust relationships change. Misbehaving nodes have to be identified and must not be allowed to remain connected to the network. Therefore, we need a certificate renewal mechanism.

5.1 Access Point

The application issues a new certificate for each connected access point B after a certain period of time. In the beginning of time slot n, A sends to B the new certificate:

$$A \rightarrow B : (Cert_{A_n}(B), \{aK_{B_n}\}_{K_{A,B}}, MAC_{K_{A,B}}(...)),$$

where aK_{B_n} is B's authentication key, which will prove B's identity to a sensor node D during handoff. Checking $MAC_{K_{A,B}}$, B can verify that this certificate was issued by the application A. To decrease the number of certificates that have to be issued, a certificate could contain several authentication keys, each of them encrypted with the TESLA key of a different time-slot, and the matching MACs at the end. However, in this paper we concentrate on one key in each certificate.

5.2 Sensor Node

The application issues a new certificate for each connected sensor node D after a certain period of time. This period of time can be much greater than the TESLA time slots as the sensor node runtime certificates don't depend on the application's TESLA keys.

$$A \rightarrow D : (\{Cert_A(D), aK_{AP,D}\}_{K_{A,D}}, MAC_{K_{A,D}}(...)),$$

where $aK_{A,D}$ is the authentication key that D uses for sensor node handoff.

Algorithm: Access Point Authentication

Result: Authenticity and Shared Secret $K_{A,B}$ between Application A and Access Point B

1 $B \rightarrow A : (offer, SIGN_{-K_B}(offer), iCert_{TTP}(B))$

2 **if** *($A\_accepts\_offer$)* **then**
  if *($SIGN_{-K_B}(offer)$ valid)* **then**
    $A \rightarrow B :$
    $(ok, SIGN_{-K_A}(ok), \{K_{A,B}, gK_{AP}\}_{+K_B})$
  else
    $A \rightarrow B : (deny, SIGN_{-K_A}(deny))$
  end
else
  $A \rightarrow B : (LoI, SIGN_{-K_A}(LoI))$
end

Algorithm 1: Access Point Authentication Algorithm

6. ENTITY AUTHENTICATION

6.1 Access Point

An access point B isn't a proper mobile device, as it features a wired connection to the Internet and is installed at a certain place for a certain purpose. In the case of a sensor network the purpose is to provide Internet-access for sensor nodes and with that a connection to their application. There is a need for authentication of the access point because it will provide access control at the interface between the application and the sensors. Once an access point B is physically connected to the wired network, it will contact its application A and send a service-offer. We assume that B knows the address of A and A's public key $+K_A$. B will sign the offer with its private key $-K_B$ and append its certificate before sending the offer to the application A (cf. Algorithm 1). If A accepts the offer, it will check the signature with help of the certificate - otherwise it sends a signed $lack\_of\_interest(LoI)$-message. If the validation is successful it returns an accept-message including a shared secret key $K_{A,B}$ and the 'access point group key' gK_{AP} (that will be used in the sensor node handoff-scenario), encrypted with the access point's public key $+K_B$ and signed with its private key $-K_A$.

The *application* $\leftrightarrow$ *accesspoint* authenticity as well as the shared secret key $K_{A,B}$ are the basis for authenticity of the entire network.

6.2 Forwarding Nodes

Forwarding nodes are mobile devices. They don't feature wired connections and are free to roam between different access points or entire networks. In contrast to an access point, a mobile node needs a more flexible authentication mechanism to support its mobility. The forwarding node is the only kind of device in the sensor network with two wireless network interfaces. Its task is to forward data packets sent by sensor nodes, that can't reach the access point directly or that can save energy by using the forwarding node as an intermediate hop.

In the current approach, forwarding nodes only authenticate themselves if a sensor node wants to send its data in the *assured mode* (cf. Algorithm 5).

Algorithm: *Sensor Node-Authentication at FN*

Result: *Authenticity+Shared Secret $K_{B,D}$ between Access Point B and Sensor Node D; Authenticity+Shared Secret $K_{A,D}$ between Application A and Sensor Node D;*

1 $D \rightarrow C : (snReq, MAC_{iK_D}(snReq), iCert_{TTP}(D))$

2 $C \rightarrow B : (snReq, MAC_{iK_D}(snReq), iCert_{TTP}(D))$

3 $B \rightarrow A : (snReq, MAC_{iK_D}(snReq), iCert_{TTP}(D), ID_B, MAC_{K_{A,B}}(snReq))$

4 ***if*** ($MAC_{iK_D}(snReq)$ valid) ***then***

$A \rightarrow B : (ok, MAC_{K_{A,B}}(ok), \{K_{B,D}\}_{K_{A,B}}, MAC_{iK_D}(ok), \{K_{A,D}, K_{B,D}\}_{iK_D})$

$B \rightarrow C : (ok, MAC_{K_{B,D}}(ok), MAC_{iK_D}(ok), \{K_{A,D}, K_{B,D}\}_{iK_D})$

$C \rightarrow D : (ok, MAC_{K_{B,D}}(ok), MAC_{iK_D}(ok), \{K_{A,D}, K_{B,D}\}_{iK_D})$

else

$A \rightarrow B : (nok, MAC_{iK_D}(nok), MAC_{K_{A,B}}(nok))$

$B \rightarrow C : (nok, MAC_{iK_D}(nok))$

$C \rightarrow D : (nok, MAC_{iK_D}(nok))$

end

Algorithm 2: *Sensor Node Authentication Algorithm*

6.3 Sensor Nodes

Sensor nodes are devices of high mobility with restricted computational power. Our goal is to provide a flexible authentication scheme that supports their mobility and relieves the sensors from intensive computations.

Once a sensor node enters the network, it sends its request to the application (cf. Algorithm 2). An intermediate forwarding node C simply forwards the request. An access point B that receives an sensor node authentication request appends its ID_B and a MAC that it creates using the key it shares with the application A. Once A gets the request, it checks the certificate. If the certificate is valid, A establishes a shared secret between the access point B and the sensor node D by returning two instances of a new key $K_{B,D}$, one encrypted with D's initialization key iK_D and the other encrypted with the key it shares with the access point. A also establishes a shared secret with the sensor node by appending another new key $K_{A,D}$, which it also encrypts using iK_D. Therefore, after receiving A's answer, D shares a secret with its gateway access point B and the application A.

7. ROAMING AND HANDOFF

The goal of "roaming" is seamless connection switching. In the hierarchical sensor network scenario, sensor and forwarding nodes switch between access points while moving or as a result of load balancing between access points. Roaming is a challenge for authentication mechanisms as trust-relations can't be reused since the network topology is changing quickly. In this section, we address authenticity problems associated with a forwarding node or sensor node changing their access point.

Algorithm: *Sensor node handoff*

Result: *Authenticity and a new shared secret $K_{B',D}$ between the senor node D and the new access point B'*

1 $B' \rightarrow D : (apHO, Cert_{A_n}(B'), MAC_{aK_{B'_n}}(...))$

2 $D \rightarrow B' : (apHO, Cert_A(D), MAC_{aK_{AP,D}}(...))$

3 $B' \rightarrow D : (hoOK, \{K_{B',D}\}_{aK_{AP,D}}, MAC_{aK_{B'_n}}(...))$

Algorithm 3: *Sensor Node Handoff*

7.1 Forwarding Nodes

A forwarding node C doesn't connect to an application A or an access point B. Since C is not involved in any authentication processes, there are no shared secrets to update when C leaves the area near an access point. Thus, a forwarding node never has to perform handoff.

7.2 Sensor Nodes

During the lifetime of the network, the sensor node D will continually send data via the access point B to the application A. When the topology of the network changes in a way that D loses its connection to B and must connect to a new access point B', the data will no longer be delivered to A but will be blocked by B'. B' will then start the handoff-process by sending an $access\_point\_handoff\_request$ ($apHO$)-message to the sensor node (cf. Algorithm 3). Appended to the message is B''s TESLA certificate and a MAC that will be used by D to verify the identity of B' after A reveals the TESLA key. Next, D answers with its own certificate and a MAC that it creates using its authentication key $aK_{AP,D}$. B' will then send D the new key $K_{B',D}$ that is encrypted using $aK_{AP,D}$, which B' obtains from $Cert_{CA}(D)$ by decrypting it using the AP group key gK_{AP}. D can obtain the new key $K_{B',D}$.

However, before D can check B''s identity, it has to wait until A publishes the TESLA key tK_{A_n} during time slot $n+d$. Once D receives that key, D can check B''s certificate and confirm the identity of B'. Subsequently, D may resume sending its data to the application by securely sending its data to B' using $K_{B',D}$.

Switching between different forwarding nodes doesn't require a sensor node handoff, since a sensor node initially doesn't share a secret key with any forwarding node. However, if the the sensor node is sending data in *assured mode* while it is switches forwarding nodes, the assured service can no longer be provided. In this case the new forwarding node sends an error message to the sensor node. If the sensor node wants to continue sending in *assured service*, it has to reinitiate the service with the new forwarding node.

8. DATA ORIGIN AUTHENTICATION

The framework's data origin authentication service enables the application A to check whether the received data are sent by a valid sensor node. The sensor node D uses the secret key that it shares with the application to create a MAC that it appends to the data packet. Depending on the sensitivity of the data or the overall trust in the network, D can decide to use the *Weak Mode* or the more sophisticated *Assured Mode* to send the data. The application authen-

Algorithm: *Sending Sensor Data in* **weak mode**

***Result**: The application A can be sure that the received data comes from sensor node D*

1 $D \rightarrow C : (data, \{rn\}_{K_{B,D}}, MAC_{K_{B,D}}(data), MAC_{K_{A,D}}(data))$

2 $C \rightarrow B : (data, \{rn\}_{K_{B,D}}, MAC_{K_{B,D}}(data), MAC_{K_{A,D}}(data))$

3 ***if*** ($MAC_{K_{B,D}}(data)$ valid) ***then***
  $B \rightarrow A : (data, MAC_{K_{A,D}}(data), MAC_{K_{A,B}}(data))$
  $B \rightarrow C : (\{rn+1\}_{K_{B,D}})$
  $C \rightarrow D : (\{rn+1\}_{K_{B,D}})$
end

4 ***if*** ($MAC_{K_{A,B}}(data)$ valid) ***then***
  if ($MAC_{K_{A,D}}(data)$ valid) ***then***
    A processes data
  else
    $A \rightarrow B : (dRej, D, MAC_{K_{A,B}}(dRej, D), MAC_{K_{A,D}}(dRej, D))$
    $B \rightarrow C : (dRej, D, MAC_{K_{B,D}}(dRej, D), MAC_{K_{A,D}}(dRej, D))$
    $C \rightarrow D : (dRej, D, MAC_{K_{B,D}}(dRej, D), MAC_{K_{A,D}}(dRej, D))$
  end
else
  $A \rightarrow B : (dRej, B, MAC_{K_{A,B}}(dRej, B), MAC_{K_{A,D}}(dRej, B))$
  $B \rightarrow C : (dRej, B, MAC_{K_{B,D}}(dRej, B), MAC_{K_{A,D}}(dRej, B))$
  $C \rightarrow D : (dRej, B, MAC_{K_{B,D}}(dRej, B), MAC_{K_{A,D}}(dRej, B))$
end

Algorithm 4: *Sending Sensor Data in* **weak mode**

Algorithm: *Sending Sensor Data in* **assured mode**

***Result**: The application A can be sure that the received data comes from sensor node D*

1 $D \rightarrow C : (asdReq, \{K_{C,D}\}_{K_{B,D}})$
  $C \rightarrow B : (asdReq, \{K_{C,D}\}_{K_{B,D}}, SIGN_{-K_C}(asdReq), Cert_{TTP}(C))$

2 ***if*** ($SIGN_{-K_C}(asdReq)$ valid) ***then***
  $B \rightarrow C : (asdConf, \{K_{CD}\}_{+K_C})$
  $C \rightarrow D : (asdConf, MAC_{K_{C,D}}(asdConf))$
end

3 $D \rightarrow C : (data, \{\{rn\}_{K_{B,D}}\}_{K_{C,D}}, MAC_{K_{B,D}}(data), MAC_{K_{A,D}}(data))$
  $C \rightarrow B : (data, \{rn\}_{K_{B,D}}, MAC_{K_{B,D}}(data), MAC_{K_{A,D}}(data))$

4 ***if*** ($MAC_{K_{B,D}}(data)$ valid) ***then***
  $B \rightarrow A : (data, MAC_{K_{A,D}}(data), MAC_{K_{A,B}}(data))$
  $B \rightarrow C : (\{rn+1\}_{K_{B,D}})$
  $C \rightarrow D : (\{rn+1\}_{K_{B,D}})$
end

5 *... similar to weak mode.*

Algorithm 5: *Sending Sensor Data in assured mode*

ticates unicast data by appending a MAC created with the secret key it shares with the relevant node. It gains access to the wireless part of the network by appending another MAC that it creates using the key it shares with the relevant access point. Access points forward only authenticated application data to avoid Internet rendered attacks against the wireless devices. Broadcast application data is authenticated using the TESLA protocol.

8.1 Sending Sensor Data in Weak Mode

Since there are frequent changes in the network topology, the sensor node D doesn't know if its data will arrive directly at the gateway access point B or will first be received by a forwarding node C, that forwards it either to another forwarding node C' or to B. Therefore, the format of the data packet must depend only on the three entities that will always be involved in the data sending process (unless a handoff happens): the sensor node D itself, the gateway access point B and the application A. As shown in Algorithm 4, the packet contains three fields in addition to the actual data: two MAC-fields and one encrypted random-number-field. If the forwarding node C receives the packet, C forwards it without doing any modification to the packet. Once the gateway access point B gets it, B checks the first MAC. If it is valid, B removes the MAC and the random-number from the packet, adds a new MAC using the secret key it shares with the application and sends it on. After that, it decrypts the random number, adds one, encrypts the result and sends it back to D. Receiving the result, D can be sure that the data is on the right way to the application. However, this *challenge-response-mechanism* does not enable D to figure out who delivered the packet to the gateway access point. There also is no certainty that a misbehaving forwarding node copied the packet without being detected. A sensor node that wants all nodes on the path to the application to be authentcated has to choose the *assured mode*.

Upon the receipt of the data, the application A checks both MACs and, if both are valid, processes the data. If one of them is invalid, A returns a *data-reject*-message ($dRej$) that includes information about which MAC caused trouble. It appends two MACs that enable B and D to verify the application as the sender of the reject-message.

8.2 Sending Sensor Data in Assured Mode

The assured mode provides authenticity along the path of the packet at the cost of additional message exchange, higher computational overhead and less flexibility. A sensor node D that wants to send in assured mode first sends an *assured-data-request* ($asdReq$) that contains an encrypted secret key that will be used to install a shared secret between D and the forwarding nodes along the path. Algorithm 5 shows a case, in which a forwarding node C relays the packets from D to B. Once C receives the request, it will sign the packet and append its *cert* before forwarding it. The gateway access point B that gets the packet first checks the certificate, then the signature and, if both are valid, replies with an *assured-data-confirmation* ($asdConf$) that includes the secret key $K_{C,D}$ encrypted with the forwarding node's public key. The

forwarding node extracts $K_{C,D}$ from the confirmation packet and uses it to create and append a MAC to the confirmation message before sending it on to the sensor node. Once the packet reaches the sensor node, it will check the MAC. If the MAC is valid, D can be sure, that its gateway access point trusts the forwarding node.

To make sure that the data takes the authenticated path, the sensor node additionally encrypts the *challenge-response random number* with $K_{C,D}$. In the case that more than one forwarding node lies on the path between the sensor node and its gateway access point, each of them appends its signature to the request before forwarding it in the direction of the access point. The access point will establish the additional needed shared secrets between sensor and forwarding nodes. D will successively encrypt the random number using each of the keys in the appropriate sequence before sending data.

9. EVALUATION

In [14], L. Zhou and Z. Haas provide a general overview of security challenges and threats in ad hoc networks. In this section we provide basic security and performance analysis for the proposed framework on the basis of their security criteria. Throughout the evaluation we assume the TESLA protocol to be secure and that loose time synchronization exists in the network.

9.1 Security Analysis

The use of wireless links renders an ad hoc network susceptible link attacks. Secondly, because mobile nodes may be compromised, malicious attacks have to be considered from outside and inside the network.

9.1.1 Wireless Link Attacks

Since this paper addresses authentication in hierarchical ad hoc networks, neither application nor sensor data is protected against eavesdropping attacks on the wireless links. However, installing a confidentiality service on top of an existing authentication service is a relatively easy task and is one of the next issues that we will address. The use of message authentication codes in our framework protects all data against malicious modifications and information forgery. In our framework, we address the threat of authorization violation by providing an access control service at the access point level. While we can't prevent intruders from coming into the network and sending packets, we can make it uninteresting for them to do so. The most likely reason for an intruder to use the network's resources is to connect to the Internet, which is prevented by access control at the access point. This access control also restricts battery-consumption attacks by making it impossible to launch such attacks from outside the wireless part of the network. The deletion of packets in the wired part of the network is a threat that we haven't addressed yet.

9.1.2 Compromised Nodes

Since all initial authentication is done by the application that drives the network, compromised sensor nodes can't inflict any damage to the network other than feeding the application with wrong data. Because of the constrained battery resources, denial of service (DoS) attacks launched by compromised sensor nodes are unlikely to happen. Even in the case of such an attack, the application can easily find the origin of the DoS-packets and end the sensor node's trust relationships in the entire network. Compromised forwarding nodes don't have any means to threaten the authentication framework because they don't share a secret with the application. Since the sensor data packets include a challenge-response mechanism, false forwarding of packets or their deletion by a forwarding node will be detected by the sensor node. The framework doesn't provide the possibility to avoid or detect the malicious duplication and distribution of sensor node packets. However, any modification to the data will be detected by the access point or the application and in case the data was sent in *assured mode* the malicious forwarding node can be identified. If a compromised forwarding node stops forwarding data to or from the sensor node or continuously modifies sensor node data that then gets rejected at the access point, the sensor node must find a different network node to connect to (i.e. another forwarding node or its gateway access point). A compromised access point threatens the network's access control mechanism. It can stop forwarding packets in both directions. While the forwarding nodes won't detect the problem, the sensor nodes will notice the lack of certificate renewals and application TESLA keys. Since a sensor node is not able to distinguish between a compromised forwarding node and a compromised access point circumstance, it acts exactly the same in this case. However, for an attacker it is much more complicated to compromise an installed access point than a mobile node. Access points should feature a high degree of physical protection. Further, once an attacker manages to compromise the application, the authentication framework fails. However, if the application itself is compromised there is no use in protecting the sensor devices or data.

9.2 Performance Analysis

We now provide basic performance analysis for our authentication framework according to the major performance criteria of [14]. First, since ad hoc networks feature a frequently changing topology, security solutions must be highly adaptable, and secondly, because an ad hoc network may consist of thousands of sensor nodes, the security mechanisms should be scalable.

9.2.1 Adaptability

Our proposed architecture is capable of adapting to meet the authentication needs of the sensors resulting from topology changes. The handoff procedure described in Section 7 facilitates the establishment of new trust relationships as nodes move without requiring the participation of the application. This is desirable since it does not burden the application, and hence the application does not serve as a bottleneck. Further, our authentication framework does not require the explicit participation of the forwarding nodes in the authentication of data. We therefore do not have to update any authentication parameters due to the mobility of the forwarding nodes.

9.2.2 Scalability

Since our scenario is application-driven, the amount of resources required by sensors to store their authentication keys remains the same as the number of sensors increases. Although the amount of keys that must be maintained by access points and the application will increase, these entities have more resources to devote to security services.

We have conducted an initial evaluation the amount of time required to a 4096-bit message authentication using SHA-1, and 2048-bit RSA signing using the libtomcrypt library [15]. Using gprof on a Pentium-4 2GHz Linux machine, we measured that SHA-1 required an average of 46 milliseconds to perform, while RSA signing required an average of 2.26 seconds to perform. Although this platform is not the same as a typical sensor node, the timing measurements do allow us to estimate that the RSA operation requires roughly 4900 times more power than performing SHA-1. We are currently conducting a more thorough estimation of power consumption on Cerfcubes [16], which is the sensor node device planned for our testbed.

In our framework, we have sought to distribute the computational load according to each layer's capabilities. Examination of our protocols reveals that we have not burdened sensor nodes with public key operations. Instead, due to their use of TESLA certificates, the sensor nodes use computationally efficient MACs to perform authentication. However, we have placed more computational burden upon higher-powered access points by requiring them to perform public key cryptographic operations.

10. CONCLUSION

In this paper, we have presented an authentication framework for an application-driven hierarchical ad hoc sensor network. Our framework authenticates incoming nodes, maintains trust relationships during topology changes through a flexible handoff scheme, and provides data origin authentication for sensor data. Further, the presented framework treats nodes according to their resource limitations. In particular, weak sensor nodes are not involved with the creation or validation of public key signatures. Instead, sensor nodes perform runtime entity authentication by the means of TESLA certificates, a new kind of certificates that we propose as an alternative to the widely used PKI certificates.

Abbreviation	Explanation
ID_D	Identification Number of node D
TS_B	Time Stamp issued by node B
$K_{B,D}$	Symmetric Key shared by the nodes B and D
$+K_A$	RSA Public Key of application A
$-K_A$	RSA Private Key of application A
gK_{AP}	Symmetric Key shared by all access points and the application (group key)
tK_{A_n}	TESLA Key of application A, valid in timeslot n
$SIGN_{-K_B}(offer)$	Signature over $offer$ by node B using its private key $-K_B$
$SIGN_{-K_{TTP}}(...)$	Signature over the complete packet by the TTP using $-K_{TTP}$
$MAC_{K_{A,D}}(data)$	Message Authentication Code of $data$ using the key $K_{A,D}$
$MAC_{K_{B,D}}(...)$	Message Authentication Code of the complete packet using the key $K_{B,D}$
iK_D	Initial Key of node D issued by the TTP
aK_{B_n}	Access Point B's Authentication Key disclosed in timeslot n
$aK_{AP,D}$	Sensor Node D's Authentication key needed for handoff with access point
$iCert_{TTP}(D)$	Initial Certificate of node D issued by the TTP
$Cert_{A_n}(B)$	TESLA Runtime Certificate of access point B issued by application A, valid in timeslot n
$Cert_A(D)$	Runtime Certificate of sensor node D issued by application A

11. REFERENCES

[1] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[2] A. Weimerskirch and G. Thonet, "A distributed light-weight authentication model for ad-hoc networks," in *The 4th International Conference on Information Security and Cryptology (ICISC 2001)*, pp. 341-354, 2001.

[3] L. Venkatraman and D. Agrawal, "A novel authentication scheme for ad hoc networks," in *IEEE Wireless Communications and Networking Conference (WCNC 2000)*, vol. 3, pp. 1268-1273, 2000.

[4] J. Kong, H. Luo, K. Xu, D. Gu, M. Gerla, and S. Lu, "Adaptive security for multi-layer ad-hoc networks," *Special Issue of Wireless Communications and Mobile Computing*, 2002.

[5] J. Kong, and M. Gerla, "Providing Real-time Security Support for Multi-level Ad-hoc Networks," *MILCOM*, vol. 2, pp. 1350-1355, 2002.

[6] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory IT 2000*, vol. IT-46(2), pp. 388–404, 2000.

[7] P. Gupta and P. Kumar, "Internets in the sky: the capacity of three dimensional wireless networks," *Communications in Information Systems*, vol. 1(1), pp. 33–50, 2001.

[8] S. Zhao, K. Tepe, I. Seskar, and D. Raychaudhuri, "Routing protocols for self-organizing hierarchical ad-hoc wireless networks," in *IEEE Sarnoff 2003 Symposium.*

[9] D. Johnson, D. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking, edited by Charles E. Perkins.* 2001, pp. 139–172, Addison-Wesley.

[10] P. R. Zimmermann, *The official PGP user's guide*, MIT Press, 1995.

[11] ITU-T, "The directory: authentication framework," IT - Open Systems Interconnection.

[12] A. Perrig, R. Canetti, B. Brisco, D. Song, and D. Tygar, "TESLA: Multicast source authentication transform introduction," IETF working draft, draft-ietf-msec-tesla-intro-01.txt.

[13] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology - Crypto '96*, pp. 1-15.

[14] L. Zhou and Z. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[15] "Libtomcrypt," www.libtomcrypt.org.

[16] "Intrinsyc product page," www.intrinsyc.com/products/cerfcube.