Secure, Redundant, and Fully Distributed Key Management Scheme for Mobile Ad Hoc Networks: An Analysis

Deepti Joshi
Kamesh Namuduri
Ravi Pendse
Presented by: Michael Clifford for CSCI 332

Introduction

- Authentication and key management for nodes on Mobile Ad Hoc Networks (MANETs) can not be performed using traditional methods used on wired networks.
- This paper proposes an alternative, using a distributed certificate authority and key management mechanism.

The Problem With PKI on Ad-Hoc Networks

Traditional Public Key Infrastructures 1

On a traditional wired network, a Public Key Infrastructure hierarchy of Certificate Authorities (CAs) is used to create and sign assertions about the identities of users and nodes.

Traditional Public Key Infrastructures 2

A trusted third party called the Root CA issues a certificate, and uses it to sign the certificates of lower CAs, which sign assertions of identity for users.

Traditional Public Key Infrastructures 3

Each node in the hierarchy must be able to obtain the entire certificate chain from a message sender to the Root CA.
Traditional PKI Fails on Ad Hoc Networks

On Ad Hoc networks, there is no guarantee that a node operating as a root CA can be reached, so traditional PKI fails.

Node 1
Node 2
Node 3
Node 4

Overview of Prior Proposed Solutions: Distributed Certificate Authorities

Partially Distributed CA [Zhou and Haas]
- CA functionality is distributed over many nodes.
- Nodes possess partial certificates.
- Nodes must have at least $k$ partial certificates to form the CA key.
- This scheme doesn’t work well because high CPU power nodes are required to take on a server role.

Fully Distributed Key Shares [Luo and Lu]
- Another previously proposed solution is to break up the private key (and operational functionality) of the CA into many “shares”.
- Each node receives one key share.
- Nodes must combine a certain number of key shares to recreate the CA key.
- This doesn’t work well, because it is hard to find a sufficient number of nodes with the right key shares in communications range.
- An Intrusion Detection System (IDS) is also required to detect single node compromises!

Introduction to the Proposal in This Paper

Proposed Solution in This Paper – Multiple Key Shares Per Node
- Each node is allocated multiple key shares.
- The probability of finding a sufficient number of nodes increases as the number of key shares per node increases.
- BUT: The probability of an intruder compromising the CA key also increases as key shares per node increase.
- So, prevent intruders from capturing enough nodes to compromise the CA key.
Definition: “Intruder”
- In this context, an “intruder” is an illegitimate node that enters the network with no key shares.
- Intruders capture or compromise nodes to obtain key shares.

Goal: Make CA Key Compromise Difficult
- Intruders must compromise y nodes to obtain the CA key.
- The goal is to make it difficult to recreate the key if y nodes must be captured, given that y-1 nodes have already been captured.

Background: Secret Sharing Equations

Secret Sharing 1
- For a (k,n) threshold sharing scheme:
  - k = minimum number of shares required to recreate the secret key
  - n = the number of nodes in the network
  - S is a secret that the n nodes want to share
  - id_i is the identifier for each node, where i is the number of the node

Secret Sharing 2
- The following steps must be performed by the “dealer”:
  - First, a prime number P must be chosen that satisfies:
    \[ p > \max(S, n) \]

Secret Sharing 3
- Next, a sharing polynomial
  \[ f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \]
  is constructed, where \( a_0 = s_{kCA} \)
  \( s_{kCA} \) is the private key of the CA
Secret Sharing 4

- Then, the dealer calculates the individual shares for each node using:

\[ S_i = f(id_i) \mod p \]

and, the resulting shares are transmitted to their corresponding nodes.

Secret Sharing 5

To reconstruct a secret key, a node must use Lagrange interpolation. Given the Lagrange coefficient \( l_{id_i}(x) \) of \( l_{id} \), which has the definition:

\[ l_{id_i}(x) = \prod_{j=1, j \neq i}^{k} \frac{x - id_j}{id_i - id_j} \]

Proactive Secret Sharing

The probability that an intruder will compromise enough shares to reconstruct the CA key increases over time.

An update function is used periodically to calculate new shares, and to distribute the new shares to each node. This decreases the probability of a total compromise.

Verifiable Secret Sharing

To ensure that each node receives valid secret shares, the dealer publishes the sharing polynomial.

Nodes then validate their shares against the polynomial to ensure that each share is valid.

The Fully Distributed Certificate Authority [Luo and Lu]
Fully Distributed Certificate Authority Services
The distributed CA must provide the following services:

- Certificate Issuance
- Certificate Renewal
- Certificate Revocation
- Share Initialization
- Share Updating

Share Initialization
- The dealer initializes k nodes, which initialize all remaining nodes.
- To initialize:
  1. The dealer creates a sharing polynomial
  2. Nodes receive their polynomial shares
  3. Polynomial coefficients are broadcast for verification
  4. The dealer destroys the original polynomial
  5. Nodes verify their shares

Addition of New Nodes (1/2)
- To join the network, a new node:
  1. Finds k nodes
  2. Broadcasts an initialization request
  3. Has its certificate validated by those nodes
  4. Receives a shuffling factor $d_{ij}$ and validating "witnesses" from each pair of nodes in k

Addition of New Nodes (2/2)
- 5. Distributes the shares and shuffling factors to those k nodes
- 6. Obtains shares from those nodes. The shares have been shuffled using the shuffling factors, in order to change the share values without altering the CA key
- 7. Combines those shares to create its own CA key share

Share Updating
Shares are updated periodically to prevent an attacker from gradually gaining control over enough nodes to reconstruct the entire CA key

Certificate Issuing and Renewal
- Unlike a regular CA, certificates are not issued. Once the CA certificate has been created, it is maintained, but not changed.
- The dealer takes on the role of adding new nodes to the network.
- Nodes can renew their certificates by checking with k nodes. A new certificate is issued only if the old one has not been revoked.
Certificate Revocation
- Each node maintains its own certificate revocation list (CRL).
- If a node observes another node misbehaving, it adds it to its CRL, and sends an accusation to other nodes, which also add the bad node to their CRLs.

Problem With Previous Work
- The Fully Distributed Certificate Authority model does not scale well because it is hard to find all k nodes to communicate with

Proposed Solution: Redundant Key Shares
- Each node receives q distinct key shares rather than exactly one key share.
- Multiple copies of the same share are present in the network, increasing the probability of finding all needed shares.
- Keep the probability of a legitimate node finding k shares as high relative to the probability of an intruder capturing k shares as possible.

Problems
- The more redundant shares that each node has, the more likely that a CA key compromise will occur.
- An IDS is required to identify attackers and cut off service to them. (This seems to be a MAJOR weakness in the paper.)

Results and Analysis
- If q and k are properly chosen, the probability of an intruder capturing enough nodes to find the CA key is about 10% lower than the probability of a legitimate node finding the key.
- Higher redundancy leads to higher probability of successful attacks.
- Larger values of k (number of nodes required to reconstruct the CA key) reduce the probability of successful attack.

(Their) Conclusions
- Redundant keys increase the ability of nodes to communicate, but also increase the probability of CA Key compromise
- Security is reduced relative to the non-redundant fully distributed CA.
- The trade-off between security and functionality must be tailored to the needs of individual networks.
(My) Conclusions

- The requirement of an IDS probably makes this model unlikely to succeed.
- The IDS is assumed to detect all attackers. Real world IDSs do not do this well.
- The requirement for a dealer and intense node cooperation seems to violate the ad-hoc property of ad-hoc networks!
- Analysis in this paper was poor. Simulation badly needed. Comparison graphs are all on different scales, making comparison difficult!
- High computational contention with attackers!