ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks

Dr.S.Makbul Hussain
Associate Professor of Mathematics,
Osmania College (Autonomous), Kurnool.

G. Mahaboob Basha
Associate Professor of Physics,
Osmania College (Autonomous), Kurnool.

ABSTRACT:

In an olden research on sensor network security mainly considers homogeneous sensor networks, where all sensor nodes have the same capabilities. Research has shown that homogeneous ad hoc networks have poor performance and scalability. The many-to-one traffic pattern dominates in sensor networks, and hence a sensor may only communicate with a small portion of its neighbors. Most existing key management schemes try to establish shared keys for all pairs of neighbor sensors, no matter whether these nodes communicate with each other or not, and this causes large overhead. We propose a novel routing-driven key management scheme, which only establishes shared keys for neighbor sensors that communicate with each other. The performance evaluation and security analysis show that can provide better security with significant reductions on communication overhead, storage space and energy consumption than other key management schemes.

INTRODUCTION:

Wireless sensor networks have applications in many areas, such as military, homeland security, health care, environment, agriculture, manufacturing, and so on. In the past several years, sensor networks have been a very active research area. Most previous research efforts consider homogeneous sensor networks, where all sensor nodes have the same capabilities. However, a homogeneous ad hoc network suffers from poor fundamental limits and performance. Research has demonstrated its performance bottleneck both theoretically and through simulation experiments and testbed measurements.

Several recent work studied Heterogeneous Sensor Networks (HSNs), where sensor nodes have different capabilities in terms of communication, computation, energy supply, storage space, reliability and other aspects. Security is critical to sensor networks deployed in hostile environments, such as military battle field and security monitoring. A number of literatures have studied security issues in homogeneous sensor networks. Key management is an essential cryptographic primitive upon which other security primitives are built. Due to resource constraints, achieving such key agreement in wireless sensor networks is non-trivial. Eschenauer and Gligor first present a key management scheme for sensor networks based on probabilistic key predistribution. Several other key pre-distribution schemes have been proposed.

Probabilistic key pre-distribution is a promising scheme for key management in sensor networks. To ensure such a scheme works well, the probability that each sensor shares at least one key with a neighbor sensor (referred to as key-sharing probability) should be high. For the key pre-distribution scheme In each sensor randomly selects its key ring from a key pool of size P. When the key pool size is large, each sensor needs to pre-load a large number of keys to achieve a high key-sharing probability. For example, when P is 10,000, each sensor needs to pre-load more than 150 keys for a key-sharing probability of 0.9. If the key length is 256 bits, then 150 keys require a storage space of 4,800 bytes. Such a storage requirement is too large for many sensor nodes. For example, a smart dust sensor has only 8K bytes of program memory and 512 bytes of data memory.

The above discussion shows that many existing key management schemes require a large storage space for key predistribution and are not suitable for small sensor nodes. In this paper, we present an efficient key management scheme that only needs small storage s pace. The scheme achieves significant storage saving by utilizing 1) the fact that most sensor nodes only communicate with a small portion of their neighbors; 2) an efficient public-key cryptography. Below we briefly discuss the two issues. More details are given in Sections II and III. Most existing sensor key management schemes are designed to set up shared keys for all pairs of neighbor sensors, without considering the actual communication pattern. In many sensor networks, sensor nodes are densely deployed in the field. One sensor could have as many as 30 or more neighbors. The many-to-one traffic pattern dominates in most sensor networks, where all sensors send data to one sink. Due to the many-to one traffic pattern, a sensor node may only communicate with a small portion of its neighbors, for example, neighbor sensors that are in the routes from itself to the sink. This means that a sensor node does not need shared keys with all neighbors. Below we give a definition that considers the fact.

EXISTING SYSTEM:

- An asynchronous message-passing system is characterized by no message transfer delay and no maximal round-trip delay.
- Even if a process is allowed to use timers, there is no way to detect a process crash, whatever the time-out values it uses.
- It is impossible to design a protocol that provides each process with the set of processes that are currently alive.

DISADVANTAGE:

- It is impossible to distinguish a crashed process from an active process and even it is difficult to identify a single process failure.
- It is difficult to identify whether a process is crashed or communications are very slow.
- No protocol is suitable

PROPOSED SYSTEM:

- The proposed model replaces the existing system problem by assuming on the maximum number of processes that could crash during a given time duration.
- The assumption is processes are provided with non-synchronized local clocks and duration time.
- If the processes are failed to do a message transfer within duration then a process is consider as crashed.

ADVANTAGES:

- The processes are provided with non-synchronized local clocks
- The protocol is based on a simple query-response mechanism and the local clock of the querying process

The protocol that allows to deal with heterogeneous networks

Find the list of Processes:

On System startup all processes are active. Every processes get all processes address at initial state and also form a group of subset.

Message Transfer

An Active Process may send a message to set of processes or to a single process.

Detecting a crashed Processes

A process does not response an arrived is considered as a crashed process. An estimate of the set of crashed processes can easily be computed by subtracting from this set.

Message Transfer to Alive

A process does response an arrived is considered as an Alive and also doing its current execution.

A Local Clock-Based Protocol

This section adapts the previous protocol to a setting without a global clock, each process being provided only with a local clock. As the local clocks are not synchronized, each clock is a "purely" local object (which means that the value of a given clock is meaningless outside its process). The problem consists, for each process pi, in associating a local date _i with each set esti:set, such that _i is as recent as possible, and all the processes that belong to esti:set were alive at time _i (assuming an external observer that uses the local clock of pi to timestamp all the events that occur in the system). As soon as such a time value is determined, pican use it to compute an approximation of the number of processes that can have crashed since the last computationof esti:set (as done at lines 03 and 04 of the global timebased protocol described in Fig. 1). Local Variables. To attain the previous goal, each process is provided with some of the previous data structures plus new ones. esti: This local variable is the same as the previous.

It has two fields esti:set and esti:date with the same meaning. The only difference is that now esti:date refers to a local date defined from the local clock of pi.. rec fromi: This local variable is now a simple set whose meaning is the same as rec fromi:set in the previous protocol. Each process maintains two additional local arrays, denoted helping datei½1::n_ and last datei½1::n_. Their meaning is the following: When a process pj returns a response to a query issued by pi, it sends its current local time value When it receives that time value (that is meaningful only for pj), pi stores it as last datei½j_ (line 05). In that way, pi is able to indicate to pj the date (measured with pj's local clock) at which pj sent itslast response to pi. Unfortunately (as we will see in Theorem 4), this is not sufficient to guarantee the property stated above relating esti:set and esti:date (all the processes of esti:set were alive at _i ¼ esti:date). We need to send back to pj not the last date, but the previous one. That date is kept by pi in helping datei½j_. Process Behavior. The behavior of pi is described in Fig. 2.

It is nearly the same as the behavior defined for the global time-based protocol. When pi sends a response message to pj, it sends the current value of the set rec fromi, the current value of its local clock (to be helped).

The routing structure in Hsns:

In this Section, we present an efficient key management scheme for HSNs which utilizes the special communication pattern in sensor networks and ECC. The scheme is referred to as ECC-based key management scheme. We consider an HSN consisting of two types of sensors: a small number of high-end sensors (H-sensors) and a large number of low-end sensors (L-sensors). Both H-sensors and L-sensors are powered by batteries and have limited energy supply. Clusters are formed in an HSN. For an HSN, it is natural to let powerful H-sensors serve as cluster heads and form clusters around them. First, we list the assumptions of HSNs below.

1) Due to cost constraints, L-sensors are NOT equipped with tamper-resistant hardware. Assume that if an adversary compromises an L-sensor, she can extract all
key material, data, and code stored on that node.

2) H-sensors are equipped with tamper-resistant hardware. It is reasonable to assume that powerful H-sensors are equipped with the technology. In addition, the number of H-sensors in an HSN is small (e.g., 20 H-sensors and 1,000 L-sensors in an HSN). Hence, the total cost of tamper-resistant hardware in an HSN is low.

3) Each L-sensor (and H-sensor) is static and aware of its own location. Sensor nodes can use a secure location service such as to estimate their locations, and no GPS receiver is required at each node.

4) Each L-sensor (and H-sensor) has a unique node ID.

CONCLUSION:

The fly determination of which processes are alive in an asynchronous-distributed system. To that end, the paper proposes to replace the traditional parameter t (that is assumed to define an upper bound on the number of processes that can crash during an execution) by a function that returns an estimate of the number of processes that can crash during a period of _ time units. The paper has proposed two protocols. The first is based on a global clock. The second uses only nonsynchronized local clocks (a local clock is used only to allow a process to measure durations). A simulation study has shown that these protocols ensure a pretty good quality of service in the sense that an alive process never remains suspected for a long time, while the crashed processes are quickly suspected.

REFERENCES:

[1] M. Ben-Or, “Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols,” Proc. Second ACM Symp. Principles of Distributed Computing.

[2] N.A. Lynch, Distributed Algorithms, p. 872. Morgan Kaufmann Publishers, 1996.

[3] T. Chandra and S. Toueg, “Unreliable Failure Detectors for Reliable Distributed Systems,”.

[4] F. Cristian and C. Fetzer, “The Timed Asynchronous Distributed System Model,”.

[5] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the Presence of Partial Synchrony,”.