International Journal of Physical Sciences Vol. 5(10), pp. 1558-1566, 4 September, 2010
Available online at http://www.academicjournals.org/IJPS
ISSN 1992 - 1950 ©2010 Academic Journals

Full Length Research Paper

A novel secure and energy-efficient protocol for authentication in wireless sensor networks

Farzad Nejati* and Hossein Khoshbin

Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Accepted 2 April, 2010

Wireless sensor networks (WSN) are typically deployed in an unattended environment, where the legitimate users can login to the network and access data as and when demanded. Efficiency of WSN depends on the correctness of the information which has been collected. Consequently, verifying authenticity and integrity of delivered data is indispensable for security-sensitive WSNs. At the same time, security is most important to prevent outsiders (illegitimate party) from retrieving the correct information. However, traditional security methods are not suited for WSN because they are not efficient from the perspective of energy, while energy conservation is a pivotal issue in WSN. This paper proposes a user authentication mechanism to countermeasure the outside attacks. The salient feature of the proposed technique is that it establishes shared values and transmits a clue message during a single authentication process without using the public key cryptography. Therefore, our proposed clue authentication scheme for WSN, provides strong authentication and shared value establishment. Our protocol is well-suited in the resource-constrained sensor nodes; furthermore, it is more secure and efficient compared to related security protocols in sensor networks. To standardize the evaluation, this paper implements the authentication protocol in the platforms of SmartDust, Strong Arm chips and Xscale. Finally, the paper analyzes its resource usage and proves its feasibility.

Key words: Authentication, hash function, security, wireless sensor networks, seed.

INTRODUCTION

Wireless sensor network (WSN) is a new network paradigm that involves the deployment of hundreds and even thousands of sensor nodes (Akyildiz et al., 2002). These sensing devices are mostly self powered and well equipped with certain computational capability. Such a device along with a processor, a communication module and a battery supply is called mote (Hill and Culler, 2002). The WSN can be used for a wide range of applications (Arora et al., 2004; Burne et al., 2001; Martinez et al., 2004; Martinez et al., 2004; Polastreet al., 2002; Szewezyk et al., 2004) including target tracking, habitat monitoring, etc. Primary goal of the WSN is to obtain globally meaningful information from strictly local gleaned by individual sensor nodes. The data collected in most of the applications are valuable and need to maintain security. Therefore, security measures to prevent the unauthorized manipulating of the correct information are most essential. We refer to such issues as outside security issues or outside attacks. User authentication is a basic preventive measure against such outside attackers (Somanath and Sukumar, 2008). However, it becomes very challenging to implement user authentication in the WSN applications, because of the limited resources available in sensor nodes.

The unattended nature of a sensor network makes it vulnerable to varying forms of security attacks such as a compromised node injecting false data reports (Karlof and Wagner, 2003; Ye et al., 2004; Zhu et al., 2004). Without identifying false reports, the sink node may reach a sub-optimal or even wrong decision. In addition, routing false reports to the sink wastes the energy of nodes along the routing path, which reduces the lifetime of the network. So identifying compromised nodes is critical since these nodes can exhaust their upstream nodes even

*Corresponding author. E-mail: farzad.nejati@gmail.com.

Abbreviations: CAP, Clue authentication protocol; **SPINS,** Security protocols for sensor networks; **ELK,** Efficient large-group key distribution.

$$A \longrightarrow B:\ C_A$$

$$B \longrightarrow A:\ C_B,\ MAC(K_{AB},\ C_A \parallel C_B)$$

$$A \longrightarrow B:\ MAC(K_{AB},\ C_A \parallel C_B)$$

MAC () : Message Authentication Code

Figure 1. Counter exchange mechanism in SNEP (16).

if the false reports are dropped en-route in just a few hops (Ye et al., 2004; Zhu et al., 2004). Schemes have been proposed to locate misbehaved nodes with en-network detection approaches. Marti et al. (2000) proposed to monitor each node by a neighboring watchdog node. Wang et al. (2003) improved the scheme through the collaborative decision of neighbors around a suspicious node. Both schemes have limitations (Marti et al., 2000) as the watchdog node maybe compromised as well. Thus compromised nodes may not be faithfully isolated.

Our clue authentication protocol (CAP) is proposed for authentication in sensor networks so that sensitive data can be protected. Sensor networks have limited resources, so, authentication has to be re-developed. Clue authentication provides a number of unique advantages. It also focuses on minimizing energy consumption and reducing risks by transmission of the clue. CAP is developed based on hint message from ELK (Penrig et al., 2001) and key chain in Security Protocols for Sensor Networks (SPINS) (Adrian et al., 2002.).

Efficient large-group key distribution (ELK) (Penrig et al., 2001) proposes a key distribution mechanism for key updating and key recovery from hint message. The hint message contains key verification of contribution nodes so that received node can generate key from this information in key updating. Key updating commences with generating a hint message from parent nodes' data. A parent node provides the hint message for child nodes to generate a new key from previous key because it recognizes all secret keys in child nodes. When a child node receives a hint message, it can build the new key from hint data. To avoid malicious messages, the new key can be verified with the hint message so that received node can be assured that the hint message is sent from the parent node. ELK updates joining nodes and leaving nodes by organizing a tree hierarchy.

Although, this can be a disadvantage when implemented. Since it cannot be assumed that network routing in the sensor network is organized in a tree hierarchy, ELK is difficult to implement. Although routing uses a tree hierarchy, sensor networks can regularly change structure. Therefore, updating hierarchy in one part of a tree requires updating the key in every related node. This causes inefficiency in energy consumption which is not suitable for sensor network. Nevertheless, the hint message mechanism provides secure processing because adversaries need to perform O (2^n) using brute force to reveal a key.

This is a motivation for our proposed solution which is described in implementation of CAP. ELK uses pseudo-random function (PRF) to generate and manage the key tree. The PRF uses a key as input to generate four different outputs. These outputs are key length, hint message, encrypted update key message and update key. On constructing the key tree, parent nodes are required to gather all child node keys and use PRF to compute the individual keys. To manage joining and leaving nodes, parent nodes must update the key corresponding to new child nodes' keys as well as acknowledge of every connected node. Therefore, key tree requires a number of message exchanges, which can drain sensor network resources.

Security Protocols for Sensor Networks (SPINS) (Adrian et al., 2002.), is a security protocol designed for energy constrained devices which maintain con-fidentiality, authentication and integrity. SPINS achieves secure communication and trust of data. It also supports key set up in sensor network. In addition, SPINS is able to update keys regularly. Therefore, it should be used as a benchmark to compare it to our proposal.

SPINS contains two security algorithms: SNEP and µTESLA. SNEP is a security mechanism for verifying integrity and data freshness, whereas, µTESLA is an authentication method for data broadcasting. SNEP is an authentication protocol to protect against replay attack. A counter adds an overhead to each packet. The counter is synchronized in both sender and receiver before communicating and incremented with every block of data sent. Therefore, counter number is never repeated. In addition, initial counter value is transmitted securely with the master key. In each packet, overhead size is only 8 bytes. The counter exchange mechanism is shown in Figure 1. C_A and C_B are counters in nodes A and B. KAB is the shared master key among node A and B. MAC (K, M) is the message authentication code of M. In this mechanism, the first two steps synchronize the counter on both parties. The last step is an acknowledgement message to ensure that the counter has been received.

µTESLA is a modified protocol of TESLA to broadcast and secure communication for a large number of nodes. The mechanism uses key verification and a key chain. In key verification, µTESLA uses symmetric cryptography instead of digital signature in TESLA. The number of senders is limited in µTESLA to reduce memory usage because each sender is required to construct a new key chain. Overhead is only per session instead of per packet. These modifications are due to resource constraints in sensor network. To set up the key chain, base station broadcasts K_0 to every node in the cluster. Then, each node can generate K_1, K_2 … K_N from K_0 by using a one-way function as shown in Figure 2. To start secure

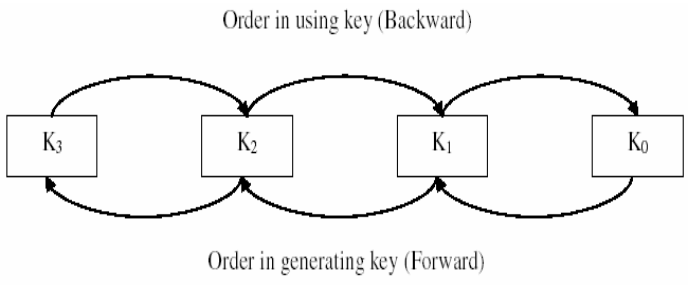

Figure 2. Counter exchange mechanism in SNEP (16).

communication, nodes use the key backward from the last key K_N to K_0, so the adversary cannot generate this chain key. For example, when an adversary can crack the message and obtain K_2, it can generate K_3, K_4 … K_N.

However, the next round of broadcasting messages will use K1 which cannot be generated by the adversary because key chain uses one-way function, thus it only can compute forward. However, it cannot compute backward. Therefore, stealing current key does not affect the rest of the key chain. µTESLA has a nonce and verification in the overhead. Nonce is a value to ensure a freshness of data which is similar to SNEP technique. In the verification, receiving node can verify the correct sender from the correct key. In addition, broadcasting data uses key delay disclosure. This mechanism can avoid a problem in transmission delay and enhance security. Since data is encrypted two keys ahead, current key can be used to verify the packet but it cannot decrypt the data. For example, if current key uses K_3, K_3 is used to encrypt a packet while K_1 is used to encrypt data in the packet. When packet is received, node decrypts the packet with K_3 and waits until K_1 to decrypt data. The benefit of this mechanism is being able to decrypt with a correct key when there is packet loss or delay. Secondly, encrypted data still cannot be revealed even though an adversary can obtain the current key because the one-way chain cannot be computed backward. Therefore, µTESLA supports a sensor network environment with unreliable communication and a large number of receivers. It reduces energy consumption by using self-authenticating keys and low overhead size in communication. However, SPINS has drawbacks in verifying compromised nodes because there is no mechanism to determine the compromised node. Therefore, at an initial stage, if adversary could proceed as one node in the network, then the base station would provide the adversary with the key.

THE ASSUMPTIONS AND MECHANISM OF CLUE AUTHENTICATION PROTOCOL (CAP)

To develop clue authentication, our objective is to secure communication and network. We assumed that the base station has the highest computing capacity and is equipped with an extensive power supply. Second, physical attack must be defended from attacks including key and program stealing. In this paper, nodes are assumed to be safe from physical tampering. Nodes can be protected from tampering by implementing watermarking, tamper-proofing and obfuscation (Collberg and Thomborson, 2002; Soo-Chang and Yi-Chong, 2006; Feng, 2000; Clark et al., 2004). We assumed that network routing is established before performing the authentication. The environment is assumed to be high risk with adversaries surrounding the network. Intruders have the ability to intercept every message of transmission as well as high performance computers and power supply.

In each node, principal seed (S_P) is pre-installed. The principal seed S_P is an initial seed for generating the shared values. It is used as an input for a one-way function to compute a consequence shared value. Authentication with correct shared values ensures authorized senders and receivers. In our clue authentication, base station generates a clue message. In addition, the base station is the most trusted device, so it takes the responsibility of broadcasting and making decisions on shared value setting up and shared value updating.

To construct a shared value, there are two kinds of operations. Both sender and receiver contain two one-way functions F_1 and F_2. These one-way functions can compute forward but they cannot compute backward. Therefore, exposing the running shared value (S_R) does not affect the clue shared value (S_C) and principal seed (S_P). Secondly, using two one-way functions instead of one in SPINS (Adrian et al., 2002) can increase the shared value space, so using two one-way functions makes finding the shared value more difficult for an adversary.

IMPLEMENTATION OF CAP

Sender generates clue message which contains a hashed value of running shared value concatenated with message M. Running shared value S_R is generated from principal seed S_P and one-way function F_1 and F_2. Principal seed S_P is used as an input. This shared value generating computes both F_1 and F_2 iteratively as shown in Figure 3. Generating a clue shared value S_C in a node can secure the running shared value S_R and simplify operations. One-way function F_1 is the first function to iteratively compute for P rounds where P is a random number. Then, one-way function F_2 begins to iterative compute for R rounds where $R = 2P + 1$. Both one-way functions F_1 and F_2 are different functions in increasing shared value space which increase the difficulty of finding the shared value. In addition, using two one-way functions can eliminate the need to re-deploy principal seed S_P because changing the shared values can be done by simply selecting new random number P. Then, running shared value S_R is changed by a new clue shared value S_C because changing number P changes the clue shared value S_C. In the implementation, the random number P in the next round must be greater than the current number so clue shared value S_C will not be repeatedly used. In addition, more rounds of updating shared values can increase shared value space. For example, if random number P is in the range (1, 2, 3 … 20), after 20 rounds, the possibility of P is in the range (20, 21, 22 … 400). Therefore, this shared value supports a long operation lifetime in sensor networks.

To implement, sender is required to update principal seed S_P from previous computing one-way function F_1 because random number P is always added on top of previous value. Therefore, it can reduce processing time by processing from previous data. In addition, sender needs to prepare hashed value from hash function H. The procedure as shown in Figure 4 begins by selecting random number P. Then, sender performs one-way function F_1 on S_P iteratively P times. After that, it finds hashed value H (M/S_C) as S1. Next, sender calculates the second number R for number of computing one-way function F_2. After completing, it computes hashed value H (M/S_R) and stores in S2. At the same time, sender

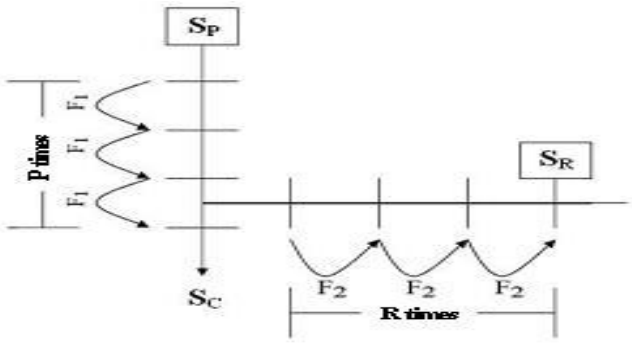

Figure 3. Procedure to find running shared value S_R from principal seed S_P by using one-way function F_1 and F_2 where P is a random number.

```
select random number P
load shared value S = SP
for j = P downto 0 do
compute key S = F1[S]
end for
compute hashed value S1 = H[M|S]
load R=2P+1
for j = R downto 0 do
compute key S = F2[S]
end for
compute hashed value S2 = H[M|S]
broadcast message (M|S1|S2)
```

Figure 4. Generating clue procedure in CAP.

broadcasts the message M with both S1 and S2 to the network. Receiver is pre-installed with principal seed S_P, one way functions F_1 and F_2 and hash function H. Updating shared value procedure in receiver is shown in Figure 5. When updating message is received, receiver obtains S1 and S2 from the message. Next, receiver computes one-way function F_1 with master principal seed S_P as an input. This process continues until hashed value of the computed clue shared value concatenated with message M is equal to S1. If the hashed value of the computed clue shared value concatenated with message M is not equal to S1 after a threshold, namely 20 in our protocol, the sender is not authenticated. Then, the receiver begins computing second one-way function F_2 for R times with computed clue shared value S_C as an input. The result is saved as S. The hashed value H (M|S) is computed and compared with S2. If they match, the sender is authenticated; otherwise the identity of the sender is not valid. Figure 6 depicts how CAP works in sender and receiver nodes.

Clue message is a message that provides information for generating clue shared value S_C and running shared value S_R as shown in Figure 7. The hint message contains two hashed values which are generated from sender as shown in Figure 4. The first hashed value allows a node to construct clue shared value S_C while the second hashed value constructs the running shared value S_R. In the node, principal seed S_P is an input of the process. The principal seed S_P is computed by using the first one-way function to generate the next shared value. The hashed value of the computed output shared value concatenated with message M is compared to the first

```
extract S1 and S2 from broadcasted message
load key S = SPc
load shared value P =0
while H[M|S] not equal to S1
compute key S = F1[S]
if P =0 sender is not authenticated
load P =P+1
end while
load R =2P-1
for j = R downto 0 do
compute key S = F2[S]
end for
compute hashed value H[M|S] and compare with S2
if two values are equal the sender is authenticated,
otherwise it is not authenticated
```

Figure 5. Receiver procedure in CAP.

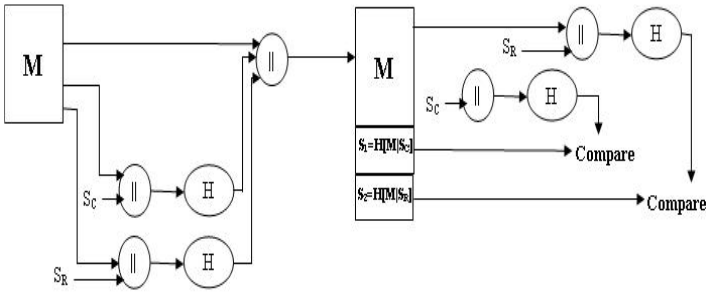

Figure 6. CAP schematic in sender and receiver.

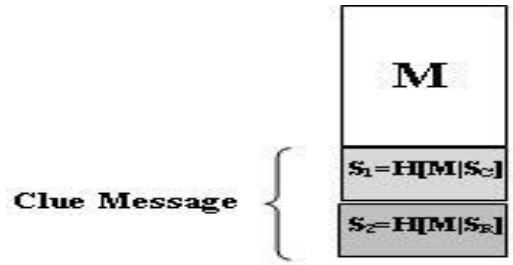

Figure 7. Clue message structure where 1st Hashed value and 2nd Hashed value are broadcasted along with the message M.

hashed value in clue message. When the result does not match, the process re-computes the output with the first one-way function until a threshold and re-compares the hashed value until the result matches. After that, the second hashed value is compared to the hashed value from the second one-way function concatenated with message M. The output from the first one-way function is an input for this process. Then, this second one-way function is iteratively computed for R times. The advantage of using a clue message is that transmitting message in shared medium does not expose a principal seed which means it is more difficult to break the shared values. In addition, hashed value is smaller fixed size compared to a key sent with a message. Therefore, energy consumption in communication can be reduced. Finally, the process is stateless for generating running shared value and clue shared value so they can

be constructed from any clue broadcasting. This also assists nodes with packet loss and joining nodes to generate shared values.

Updating shared values procedure

There are two ways to perform the shared value update procedure. First, updating both shared values is computed as shown in sender and receiver parts. Second, updating running shared value with the previous clue shared value is quicker and uses less energy for short term purposes. Sender uses the same principal seed S_P and reduces random number P to compute hashed value. Receiver is not required to compute first one-way function F_1 because it uses the previous clue shared values. To find running shared value, it computes a shorter clue shared value chain from principal seed S_P to second one-way function F_2. In the implementation, both sender and receiver do not need to compute these one-way functions F_1 and F_2 because these clue shared values have been computed previously. Therefore, sender only looks for hashed value in previous computed clue shared values while receiver only matches the received hashed value and the hashed value of previous computed clue shared values concatenated with message M. As an example, assume both sender and receiver currently use principal seed S_P and running shared value S_R. To update shared values, sender could randomly select S_{P-1}. Since this shared value is already computed, sender simply computes H ($M|S_{P-1}$) and then broadcasts it to the network. When receiving this message, receiver can obtain H ($M|S_{P-1}$). After that, it searches for hashing value in memory and obtains S_{P-1}. An advantage of this updating shared value is reducing computation which minimizes both energy consumption and delay. In addition, using a shared value backward can protect against adversary computing new shared value from previous shared value because this shared value is computed from one-way function.

Joining node and packet loss are supported by our clue authentication protocol. Since next round of shared value updating contains hashed value, joining nodes can generate running shared value from initial principal seed S_P. In packet loss, next hashed value provides sufficient information for receivers to generate running shared value. In addition, the hashed value is unique so it ensures the same running shared value in both senders and receivers. The only drawback is when a joining node has been in operation for some time in a network. So the joining node requires more computation time which can cause a delay in communication.

Features of CAP

As clue authentication protocol is implemented with clue message, clue shared value and shared value self-generating; it provides a number of features which are described as follows. The number of messages required for establishing shared value is reduced. Clue authentication protocol uses principal seed pre-distribution and clue message technique to construct shared values so only one message is required. This message contains only a clue for running shared value which is sufficient for authorized nodes to construct the shared value. When compared with SNEP in SPINS (16), at least three messages are required to set up a secure channel. This can enhance system lifetime as well as reduce setting up period.

Shared values are generated in each node and not needed to be broadcasted by the base station individually. In clue authentication protocol, the base station gives each node a clue so running shared value can be constructed from this clue. The clue message is the hashed value of the running shared value concatenated with message M. Since authorized nodes have pre-installed principal seed, they can compute running shared value from the clue. However, clue message does not provide enough information for adversaries to generate shared values because a hashed value cannot be computed backward to find the actual shared value.

Updating shared value is flexible for base stations

In clue authentication protocol, the base station can update the shared value immediately to correspond to the situation by broadcasting clue message. Then, every node will update their shared value from this broadcast. Therefore, their shared value can be updated dynamically based on the situation. In addition, the base station can use running shared value longer in low risk environment thus energy consumption in updating shared value can be reduced.

In addition, clue message is very small in size so it is attached as a part of transmission data. If the system decides to maximize battery lifetime, base station can attach the clue to a large block of data. Receivers can update keys immediately when the message is received. The data is also protected with the updated shared value in this transmission.

Protocol supports leaving nodes, joining nodes and node failures

Since clue authentication protocol can update the shared value by clue message, organization of tree is not necessary. In leaving nodes and nodes failures adjusting or communicating as in ELK (15) is not required. Joining nodes do not require special maintenance because the hashed value in clue message has sufficient information for new authorized nodes to generate the shared value.

Protocol should minimize resource consumption in key management

Clue authentication protocol uses a clue message which is very small in size. Furthermore, the message is not needed to be encrypted or decrypted. This also increases lifetime of the system besides compact communication. In addition, clue authentication protocol is stateless so it does not require a large space in memory. The memory only stores pre-installed principal seed, clues and running shared values.

Clue authentication protocol (CAP) evaluation

This section gives an evaluation of clue authentication protocol, compared with ELK and SPINS. The section begins by description of metrics, parameters and scenarios in the evaluation. Models are then constructed. Finally, clue authentication protocol is analyzed.

Metrics of evaluation

Resource usage considers the amount of energy consumption in communication and computation. As communication is the most energy intensive activities in sensor networks, it must be minimized. This metric also determines the system lifetime for the protocol operations.

Sensor network devices have a limited resource so protocol must use this resource efficiently. Processing time and memory are also considered because these are limited in sensor networks.

Parameters of evaluation

In this part, parameters of the evaluation are explained. These parameters are considered in resource usage. The number of

messages, message size and frequency are used as parameters because they affect communication energy consumption which consumes the most energy in sensor networks. The evaluation uses this information to calculate an estimation of system lifetime.

Processing time is used to determine the computing capability of Central Processing Unit (CPU). Memory size is also needed to determine the requirements in implementing protocols because sensor network devices have limited memory size.

Evaluation scenario

To standardize the evaluation, Smart Dust (21 - 26), Strong Arm chips (21, 27) and Xscale (21, 28 - 31) are used as analysis platforms. Power is supplied from 3 volts battery with capacity of 2,200 mAh. Our model sets up 10 nodes in each cluster and sampling rate is 1 Hz with 50 Kbps bandwidth. Wireless communication consumes 4.8 mA in receiving and 12 mA in transmitting. In idle mode, energy consumption rate is 5 µA. In addition, there is end-to-end data communication between node A which is a base station of the cluster and node B which is placed in the cluster. A path between A and B is connected along the nodes in the same cluster as: $A \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots n_m \rightarrow B$. This network also has a routing path set up. Each node in network has a strong physical protection. Adversaries cannot break the device to retrieve the principal seed or data inside directly. Also, the length of principal seed is evaluated with 40 and 128 bits. To compute the clue message, MD5 and SHA 1 are used as clue functions (H) to evaluate the protocols.

Adversaries have Sun UltraSparc II 440 MHz server. The UltraSparc is 64 bits RISC based on architecture with 16 KB data cache and 2 MB external cache. Its wireless antenna can reach the entire network. When an adversary launches attacks, it can be initiated from anywhere along a path. The simulator we have programmed is a wireless network simulator for energy consumption which is based on MATLAB. The simulator is based on an event-driven model. The operation of nodes is developed on an event basis. For periodic tasks, clock parameter or clock tick could be used for assigning the task. A wireless communication is built in the program which can adjust the parameters e.g. signal strength and error rate. Our adversary in simulator is also developed in this simulator with high processing capability (UltraSparc II) and high transmitting power. Sensor network nodes are equipped with limited capacity battery and less transmitting power than adversary. The energy consumption varies with signal power, message size and activities. Simulation results are exported to MATLAB for further analysis.

Performance model

This section presents the theoretical model of ELK, SPINS and CAP resource usage metrics.

Performance of ELK

Energy is used to compute and broadcast messages in the established tree when nodes join or leave the network. When nodes are joining, each node can compute individually without broadcasting messages. To update the key, a hint message is broadcast in order to allow new key to be constructed. The best scenario for effective energy consumption is that each node updates its key without broadcasting messages. This only requires small computation and memory. In the average case, hint messages are broadcasted so each node needs to consume power in communication and computing new key. The hint message size corresponds to the number of left and right contribution nodes in the tree because hint messages are generated from all keys in child nodes. In the process of key construction, firstly a message is decrypted and secondly to match with the hint, the key is computed. The worst case scenarios are both setting up tree and leaving nodes. The server begins computing a new key, which corresponds to the current existing nodes and broadcasts the updating message. Each node, then, computes its key. Therefore, it requires the number of messages to verify the status of tree and broadcast updating messages as well as computing key in each node.

As tree structure needs to be maintained, a regular communication is required for ELK. In addition, this protocol does not support packet loss because the consecutive packet loss can be interpreted as leaving node and consequently tree needs to be reconstructed. Furthermore, changing key in the child nodes requires the entire parent nodes to be updated, which is quite expensive. For concrete evaluation, the result is shown in next section.

For the number of key bits n, key space is 2^n. The adversary is required to compute at least 2^{n-1} keys in brute force attack. To update key, hint message is used to hide an actual key from the adversary. In addition, the adversary has some difficulties in obtaining the group key because a cluster contains a large number of nodes. This is a significant advantage in sensor networks because network size tends to be hundreds or thousands nodes.

Performance of SPINS

In SmartDust nodes, 98% of energy consumption is due to communication and 2% of energy consumption for computation. The energy for communication can be categorized into data transmission with 71%, header transmission with 20% and Nonce transmission with 7% of total energy. Computation uses only 2% of energy cost for computing encryption as shown in Table 1. Although most energy consumption is from communication, it is a common behavior in sensor networks. In computing, processing time in key set up is 3.92 ms (16). Memory uses 120 bytes for the protocol. Therefore, SPINS demonstrates a capability of implanting security in sensor nodes.

Key space is 2^n where n is key bits, so brute force can find the current key on average by computing 2^{n-1} times. Yet, decrypting message requires two keys so brute force needs to compute at least 2^n keys. To find a key chain, key space is $R \cdot 2^n$ where R is the maximum number of possible keys in the key chain. However, adversaries have a difficulty in obtaining number R because it has never been stated in any message. Therefore, adversaries require computing all possibilities by beginning from small number of R. For example, $R \cdot 2^n$ where R begins from 1, 2,.. . Hence, the maximum computing time of a master key is $R! \cdot 2^n$. When a base station updates a new key chain, an adversary is required to re-compute this key chain again. Therefore, a key chain is protected by security that is higher than that for a simple key. In addition, a key chain is regularly updated, thus the key chain is secured in a period of time.

Performance of CAP

As CAP uses the similar one-way function as SPINS, memory usage is equal to 120 bytes. However, shared value set up requires the comparison of clue so it requires another 80 bytes. In addition, shared values need to be stored in memory all the times. If shared value size is 64 bits, 80 bytes of memory is required for shared value chain size of 10. Therefore, the total memory is approximately 280 bytes. In simulation, 400 bytes memory is reserved, but on average it uses 220-320 bytes. In communication, energy use is less than SPINS because the number of communication is reduced and message size is smaller. The details are demonstrated in next section.

Table 1. Energy cost in SPINS (16).

Transmission	Percentage
Data transmission	71
Header transmission	20
Nonce transmission (Freshness verification)	7
Encryption computation	2

Table 2. Energy cost in SPINS.

Feature	SPINS	CAP	ELK
Regular renew key or shared value	√	√	
Regular renew key or shared value, based on event			√
Regular renew key chain or shared value	√	√	
Regular exchange information among nodes	√		√
Regular verify time counter (Nonce)	√		

Table 3. Lifetime in communication for each protocol.

Protocol	Message size (bytes)	Estimated operation time (days)
ELK (best case)	23 - 38	962
ELK (average)	23 - 38	102
ELK (worst case)	23 - 38	51
SPINS	598	263
CAP	64	847

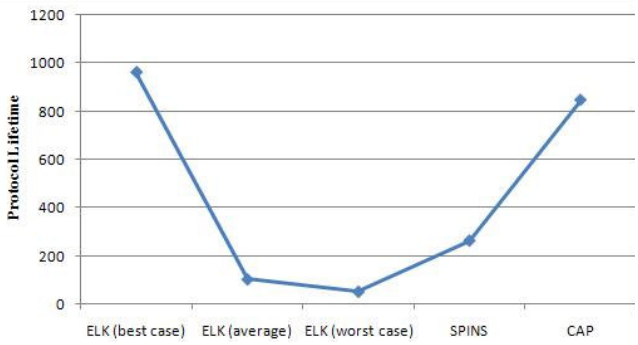

Figure 8. Protocol lifetime.

Shared value space is 2^n for running shared value where n is principal seed bits so on average 2^{n-1} computations is required for brute force attack. To compute possible different running shared values, the shared value space is $R \cdot 2^n$ where R is the maximum number of possible running shared values, as described in the algorithm. To find a principal seed in CAP, shared value space is $P \cdot R \cdot 2^n$ where P is the number of possible clue shared values, as described in the algorithm. However, adversaries have difficulties in obtaining numbers P and R except the node that has already obtained all the principal seed, clue shared value and running shared value because there is no information stated on the numbers. Since the number of P and R could be varied from zero to infinity, it is infeasible to calculate the maximum number of P and R in one time. The adversary requires computing from smaller numbers of P and R. Ideally; the adversary begins computing each set as follows.

(P =1, R =1), (P =1, R =2) … (P =1, R = R);
(P =2, R =1), (P =2, R =2) … (P =2, R = R);
(P = R, R =1), (P =2, R =2) … (P = P, R = R).

Although the adversary could keep the previous computing numbers P and R, it is infeasible to store the previous $2^n \times P \times R$ in UltraSparc II. Therefore, the adversary needs to re-compute numbers P and R. As a result, the maximum computing time of principal seed is $P! \cdot R! \cdot 2^n$. Therefore, principal seed is the largest key space in CAP which is equivalent to the most secure algorithm.

EVALUATION OF RESULTS

This section evaluates ELK, SPINS and CAP in resource usage and energy consumption. As wireless communication is the most energy consumption in sensor networks, our simulation focuses on the message transmission.

Table 3 is the simulation result which shows the energy consumption in CAP, SPINS and ELK. The comparison of results has been depicted in Figure 8. This simulation focuses on the message size and system lifetime. The estimated system lifetime is calculated from protocol

Table 4. Energy savings feature in each protocol.

	CAP	SPINS	ELK
Not require re-organizing structure	✓	✓	
Self-generating key or shared value	✓	✓	
Support packet loss	✓	✓	
Construct key or shared value from clue message	✓		✓
Not require exchanging information	✓		

operations which neglect sensors and non-related operations. ELK (best case) updates the key by self-generating with the low number of messages. In ELK (average), hint messages and tree maintenance messages are used. ELK (worst case) needs to re-organize tree structures frequently due to packet loss and leaving nodes. Therefore, exchanging messages and many key updates are required. In SPINS and CAP, the protocols do not reflect the structure of network so simulation uses the average scenarios.

The result shows that the expected lifetime in CAP is almost 3.2 times greater than SPINS because SPINS requires an authentication among the nodes before transmitting the data. In addition, ELK (average) and ELK (worst case) consumes more energy than CAP because ELK uses a tree to distribute keys, which are opposed to traditional broadcasting in CAP. Additionally, a leaving node in lower branch of the tree in ELK requires many messages to adjust the tree as well as update the key. Although ELK (best case) shows the best performance, it rarely occurs in practice because sensor networks are unreliable and many unexpected events often occur. SPINS demonstrates the average performance among three protocols because it reduces the number of communication and uses a self-generating key. CAP shows the best energy consumption because it uses only one broadcasting message to authenticate while joining nodes do not need the extra communication, as next round of shared value updating contains hashed value. In addition, packet loss does not affect the authentication in CAP, since next hashed value provides sufficient information for receivers to generate running shared value.

In computing resource, our simulation uses MD5 as a hash function. MD5 consumes 0.59 µJ/byte, which can be compared to 3DES computation 6.04 µJ/Byte (32). So sensor nodes have the capability to compute this function and are also able to perform CAP. High power processor Strong Arm chip computes each MD5 140 µs in small wireless network device (32). In simulation, random number P is in the range of 1, 2, 3 … 20. On average, MD5 is required to be computed 10 times (average 10 times for P). This equals to 1.40 ms (140 µs x 10 times). To compute MD5 in low power CPU (Xscale in energy safe mode), it requires 180 µs (32) for each computation or 1.80 ms (180 µs x 10 times) per authentication. CAP uses the similar one-way function as in SPINS. Therefore, the total operation time is the sum of hash function and one way function. As each one way function uses 3.92 ms, two one way functions use average 7.84 ms. The total time in generating the message to be sent in CAP is between 10.64 and 11.40 ms. Therefore, our simulation ensures that computation time in CAP does not exceed the capabilities of a sensor node. However, this processing time is more than 3.92 ms in SPINS. In ELK, it is the worst performance in simulation because operations in the protocol are involved with asymmetric cryptography. It uses up to 2 min for generating key in the deep tree hierarchy. In summary, the highest computation time is for ELK which is a large difference from CAP and SPINS because ELK does not focus on energy consumption and using asymmetric cryptography. SPINS uses the average energy consumption while CAP saves most energy because of the least communication messages.

ELK uses the largest memory size because of asymmetric cryptography. In the 10 levels tree, 6.86 MB is used to compute a key which is infeasible for sensor nodes. SPINS uses only 120 bytes memory for the protocol. In addition, CAP uses 280 bytes in the memory. Therefore, both SPINS and CAP could be implemented in sensor nodes while SPINS is the most efficient in memory usage.

Conclusion

In conclusion, approximately 98% of energy usage in the protocols is from communication task as shown in Table 1.

The characteristic of the protocols are shown in Table 2 and 4. Finally, the comparisons of energy consumption in these protocols are shown in Table 3. ELK uses an excessive resource especially memory and CPU which are infeasible to implement in sensor networks. The reason is that ELK uses asymmetric cryptography and it does not focus on minimizing the resource usage. In SPINS, memory and CPU usages are the lowest among three protocols followed by CAP and ELK, respectively. In addition, CAP uses the lowest energy in operation. Therefore, CAP can enhance the most system lifetime. However, it still uses memory and CPU processing more than SPINS.

REFERENCES

Adrian P, Robert S, Tygar JD, Victor W, David EC (2002). "SPINS: security protocols for sensor networks" Wireless Network, 8: 521-534.

Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002). Wireless sensor networks: a survey, Comput. Networks 38(4): 393-422.

Arora A, Dutta P, Bapat S, Kulathumani V, Zhang H, Naik V, Mittal V, Cao H, Demirbus M, Gouda M, Choi Y, Herman T, Kulkurni S, Arumugam U, Nesternko M, Vora A, Miyastha M (2004). A line in the send: a wireless sensor network for target detection,classi.cation and tracking. Comput. Networks. 46(5): 605-634.

Burne RA, Buczak AL, Jamalabad VR, Kadar I, Eadan ER (2001). Selforganizing cooperative sensor network for remote surveillance improved target tracking results, in: Proc. SPIE. 4232: pp. 313-321.

Clark T, Jasvir N, Ram S, Charles H (2004). "Tamper-proofing software watermarks," in Proceedings of the 2nd workshop on Australasian information security, Data Mining and Web Intelligence and Software Internationalisation, Dunedin, New Zealand, Australian Computer Society, Inc.

Collberg CS, Thomborson C (2002). "Watermarking, tamper-proofing and obfuscation - tools for software protection," IEEE Trans. on Software Engine. 28: 735-746.

Feng B (2000). "Multimedia content protection by cryptography and watermarking in tamper-resistant hardware," in Proceedings of the 2000 ACM Workshops on Multimedia, Los Angeles, California, United States: ACM Press.

Hill J, Culler D (2002). Mica: a wireless platform for deeply embedded networks, IEEE Micro., 22: 6.

Karlof C, Wagner D (2003). Secure routing in wireless sensor networks: attacks and countermeasures. In: IEEE international workshop on sensor network protocols and applications, pp. 113–27.

Marti S, Giuli TJ, Lai K, Baker M (2000). Mitigating routing misbehavior in mobile ad hoc networks. In: MOBICOM.

Martinez K, Hart J, Ong R (2004). "Environmental sensor networks," *IEEE* Comput. 37: 50-56.

Martinez K, Ong R, Hart J (2004). Glacsweb: a sensor network for hostile environments, in: Proc. IEEE SECON. pp. 81-87.

Penrig A, Song D, Tygar D (2001). "ELK, a new protocol for efficient large-group key distribution," in Security and Privacy, Oakland, CA. pp. 247-262.

Polastre J, Szewczyk R, Culler D, Anderson J (2002). Wireless sensor networks for habitat monitoring, in: Proc. ACM Workshop on Wireless Sensor Networks and Applications. 88 – 97.

Somanath T, Sukumar N (2008). “Defense against outside attacks in wireless sensor networks,” Comput. Comm. 31: 818-826.

Soo-Chang P, Yi-Chong Z (2006). "Tamper proofing and attack identification of corrupted image by using semi-fragile multiple-watermarking algorithm," in Taipei, Taiwan: ACM Press. Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security.

Szewezyk R, Osterweil E, Polastre J, Hamilon M, Mainwaring A, Estrin D (2004). Habitat monitoring with sensor networks, Commun. ACM. 47(6): 34-40.

Wang G, Zhang W, Cao G, Porta TL. On supporting distributed collaboration in sensor networks. In: IEEE MILCOM, 2003.

Ye F, Luo H, Lu S, Zhang L (2004). Statistical en-route detection and filtering of injected false data in sensor networks. In: IEEE INFOCOM 2004.

Zhu S, Setia S, Jajodia S, Ning P (2004). An interleaved hop-by-hop authentication scheme for fltering of injected false data in sensor networks. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, California.