Reducing Radio Energy Consumption of Key Management Protocols for Wireless Sensor Networks

Bo-Cheng Charles Lai[1] David D. Hwang[1] Sungha Pete Kim[2] Ingrid Verbauwhede[1]

1: Electrical Engineering Department
University of California, Los Angeles
Email: {bclai,dhwang,ingrid}@ee.ucla.edu

2: Institute of Intelligent Systems, Mechatronics Center
Samsung Electronics Co., LTD.
Email: yevgeny.kim@samsung.com

ABSTRACT

The security of sensor networks is a challenging area. Key management is one of the crucial parts in constructing the security among sensor nodes. However, key management protocols require a great deal of energy consumption, particularly in the transmission of initial key negotiation messages. In this paper, we examine three previously published sensor network security schemes: SPINS and C&R for master-key-based schemes, and Eschenhaur-Gligor (EG) for distributed-key-based schemes. We then present two new low-power schemes, which we call BROSK and OKS as alternatives to master-key-based schemes and distributed-key-based schemes, respectively. Compared to SPINS and C&R protocols, BROSK can reduce energy consumption by up to 12X by reducing the number of data transmissions in the key negotiation process. Compared with EG, OKS reduces energy by up to 96% and reduces memory requirements by up to 78%.

Categories and Subject Descriptors

C.2.2[Computer-Systems Organization] : Network Protocols

General Terms: Security

Keywords: Sensor Network, Key Management Protocol

1. Introduction

Due to emerging low power, embedded and wireless technologies, distributed sensor networks can be realized by establishing networks among a large amount of tiny and resource constrained sensor devices. The application spectrum ranges from object tracking, habitat monitoring [1], the smart space [2], to ubiquitous computing environments [3].

Because vast quantities of sensor nodes are distributed in the network, extremely low cost and low power become the core design challenges. The low cost constrains the resources that can be implemented on the devices, and low power requires the operations to be done in a highly efficient way. Moreover, due to the large scale and distributed nature of wireless sensor networks, the protocols and algorithms must be scalable.

ISLPED'04, August 9–11, 2004, Newport Beach, California, USA.

The security of sensor networks is very challenging. Applications of sensor networks are as diverse as habitat monitoring, homeland healthcare, disaster site rescue, and military surveillance. Although different security levels are required for different applications, baseline security mechanisms are needed to ensure the functionality of the network and protect the privacy and integrity of the sensitive data. Imagine a scenario in which a sensor network is deployed for homeland security. If there is no authentication and access control over this sensor network, an intruder can potentially give a false command to the sensor nodes and turn them into sleep mode without detection of abnormal activities. Thus the sensor network should only accept legitimate queries, commands, and re-configuration.

Security protocols are rooted on secret keys which are pre-shared among the members in the network. Members in the network use the secret key to authenticate other members or encrypt the sensitive data that is transmitted in the air. However, using the same secret key on every wireless link will significantly increase the chance of crypto-analytic attacks. Keys that are different for different links are called link-dependent keys or session keys. Session keys have higher security assurance because they vary over time and space, and thus are preferred for use on wireless links.

Conventional security protocols are usually master-key-based or distributed-key-based management schemes. Master-key-based schemes are those in which every node shares a single pre-installed master-key. Session keys that will be used on different wireless links can be negotiated by, for example, a simple three way handshaking and authentication protocol [10] based on the master-key. This type of key management scheme has the underlying assumption that the sensor nodes are tamper proof and the master-key which is stored inside each node cannot be retrieved by the adversary [9][10]. However, the assumption that the nodes are tamper-proof cannot be ensured in many sensor network applications because sensor nodes are usually left unattended in a hostile environment. Once the master-key has been hacked, the adversary can use it to break the security of the entire network.

Distributed-key-based schemes pre-install a subset of keys on each node from a large key-pool [4]. By successfully hacking one node an adversary only obtains a small portion of the whole key-pool. This makes the sensor network more robust to node capture attacks. Due to the necessity of storing multiple keys, a

distributed-key-based scheme requires more memory space than a master-key-based scheme.

In this paper, we propose two energy efficient key management protocols: *BROSK* (BROadcast Session Key negotiation protocol) and *OKS* (Overlap-Key-Sharing), a master-key-based scheme and distributed-key-based scheme, respectively. Compared with existing master-key protocols, BROSK can save up to 12X energy consumption by reducing the number of data transmissions in the key negotiation protocol, while OKS can save 96% of the energy consumption versus existing distributed-key protocols. OKS can also achieve 78% memory requirement reduction.

This paper is organized as follows. Section 2 describes existing master-key-based key management protocols and the BROSK protocol. Section 3 discusses distributed-key-based key management protocols and introduces the OKS protocol. The energy consumption and resource requirements are compared and discussed in Section 4, and conclusions are given in Section 5.

2. Master-Key-Based Key Management Protocols

Master-key-based key management protocols pre-install a shared master-key on every node. Session keys that will be used on different wireless links can be negotiated by a handshaking protocol [10]. This section introduces two previously published existing master-key-based key management protocols: SPINS [8] and challenge-and-response [10]. A new session-key negotiation protocol, BROSK, will also be presented.

2.1 Notation

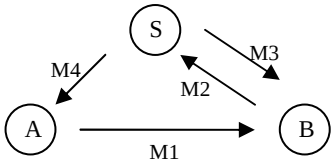
The following is the convention we use to describe protocols in this paper.

- A | B: data A concatenated with data B.
- $\{M\}_{KAS}$, encryption of message M by key K_{AS}.
- $MAC_K[M]$: MAC (message authentication code) of message M created by key K.
- N_A : a nonce generated by node A. A nonce is a one-time random bit-string, usually used to achieve freshness. Data freshness implies that the data is recent, and it prevents the adversary from replaying old messages.
- ID_A : the identity of node A.

2.2 SPINS

SPINS is a well-known security suite for sensor networks [8]. It includes two protocols, SNEP and μTESLA. The former is for confidentiality, two-party data authentication, integrity, and freshness, and the latter provides authentication for data broadcasting. Here we focus on the key negotiation protocol of SPINS. As shown in Fig. 1, assume that node A wants to establish a session key K_{AB} with node B through a trusted third party S, the central key distribution center (KDC). This is a server that can perform authentication and key distribution. Nodes in SPINS have individual shared master-keys (i.e. each node and the server share a unique key). Each node uses this master-key to authenticate itself with the server.

Node A wishes to establish a secure key with Node B. Node A sends a request message (M1) to node B in Fig. 1. Node B receives this message and sends a message (M2) to key server S. Key server S performs authentication and generates the shared session key (K_{AB}) and sends the key back to node A and node B respectively (M3 and M4).



M1: $N_A| ID_A$

M2: $ID_B|\{N_A|N_B| ID_A\}_{KBS}$

M3: $\{K_{AB}|N_A| ID_B\}_{KAS}$

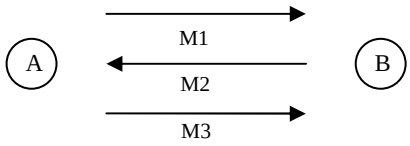
M4: $\{K_{AB}|N_B\}_{KBS}$

Fig. 1 Session key agreement protocol of SPINS

Strictly speaking, SPINS cannot be categorized as a master-key-based protocol, because nodes in SPINS have different "master-keys" with the key server and negotiate session-keys to other nodes through the server. However SPINS has a similar security hole as a pure master-key-based protocol in a sense that once a node's "master-key" to the server has been hacked, the adversary can negotiate valid session-keys with any other nodes within the local area through the server. Therefore we group SPINS together with the other master-key-based protocols.

2.3 C & R (Challenge and Response)

The Challenge and Response protocol (C&R) is a simple protocol that can authenticate and negotiate keys in an ad hoc scheme [10]. Nodes in the network have a shared master-key that they use to authenticate each other. In C&R, a key server is not needed to perform the key negotiation process.



M1: request | N_A | ID_A

M2: $(ID_B|ID_A|N_A|N_B) | MAC_K[ID_B|ID_A|N_A|N_B]$

M3: $ID_A|N_B|MAC_K[ID_A|N_B]$

$K_{AB} = MAC_K[N_A|N_B]$

Fig. 2 Key negotiation protocol of C&R

A simple example is shown in Fig. 2. Node A first sends a request message (M1) to node B. Node B replies with message (M2) as a challenge to node A. When node A receives this message, it proves its authenticity by sending the message (M3) back to node B. This is a mutual challenge and authentication procedure and both node A and node B use K_{AB} as their shared session key.

2.4 BROSK (BROadcast Session Key Negotiation Protocol)

BROSK is a fully ad hoc key negotiation protocol. Each node can negotiate a session key with its neighbors by broadcasting the key negotiation message (M1 in Fig. 3).

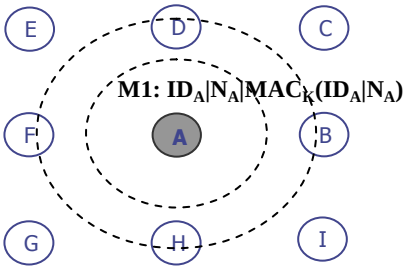


Fig. 3 BROSK: Node A broadcasts key negotiation message

K is the master-key shared among all nodes. ID_A is the identity of node A, and different nodes have different IDs. Once a node receives the introduction message broadcasted by its neighbor, it can construct the shared session key by generating the MAC of two nonces. For example, in Figure 3 node B will receive the broadcast message from node A. Node A will also receive the broadcast message from node B (M2 in Fig.4). They then use K_{AB} (in Fig.4) as their shared-session key.

M2: $ID_B|N_B|MAC_K(ID_B|N_B)$

K_{AB}: $MAC_K(N_A|N_B)$

Fig. 4 BROSK: Message M2 and shared session key K_{AB}

3. Distributed-Key-Based Key Management Protocols

Distributed-key-based key management protocols specifically address the security hole in which sensor nodes can be physically captured by the adversary, causing secret keys to be retrieved. In these schemes, only a subset of keys are selected (usually at random) from a large key-pool and stored in each node. Thus capturing a node reveals only a small part of the total secret key-pool. After nodes have been deployed, a *key-discovery procedure* is required in which sensor nodes show their neighbors which keys they own (by broadcasting only the indices of the keys). Neighboring nodes form a secure link when they own the same keys.

Definition 1: Two sensor nodes have a *secure-link* if and only if they are in the radio range of each other and they have the shared secret key between them.

Definition 2: The network is *secure-connected* when the network can be connected by only *secure-links*.

Because the keys are selected at random from the key-pool, by probability, two neighboring nodes may not have the same key. Thus there is no shared key that can be used on this wireless link. This problem can be solved by additional protocols that negotiate keys through other secure paths [11], but this issue is beyond the scope of this paper. Here we focus on deriving an equation such that for a given size of the network (number of nodes in the network), the parameters such as the length of the key-string and number of key-strings can be decided to achieve a high probability that the entire network is secure-connected.

3.1 Random Key Distribution

Eschenhaur-Gligor [4] propose a random key distribution scheme, which we call EG. Given a long key-pool of *S* keys, *m* distinct keys are randomly selected from the key-pool and installed into each sensor node. The number of keys in the key pool, *S*, is chosen such that two random subsets of size *m* in *S* share at least one key with a high probability *p*. In [5], Erdos and Redyi showed that for a graph *G(n, pi)* with monotone properties, where *n* is the number of vertices and *pi* is the probability that any two vertices have a link, it is possible to get the expected degree for each vertex in the graph such that the probability of the graph to become connected is high. Here we define the degree of a node as the number of wireless links a node has to other nodes in the network. As analyzed by [4], the necessary expected node degree *d* in terms of the number of the nodes in the network *n* is:

$$d = (\frac{n-1}{n})(\ln(n) - \ln(-\ln(p))) \quad \textbf{(1)}$$

Here we give a simple example. We want to achieve the probability *p* that the network is connected to be 0.99999. If there are 10,000 nodes in the network, then the degree of each node needs to be 20.

3.2 OKS (Overlap-Key-Sharing)

The OKS protocol generates a long bit-string to be the key-string-pool (*KP*) of the sensor network, and randomly assigns a subset of the key-string-pool to be the key-string stored in each sensor (shown in Fig. 5). Sensors in OKS protocol use the overlap intervals (number of bits overlapping between neighbors) of the key-strings as the shared secret key with their neighbor nodes. This differs from EG in that the EG protocol [4**]** has key-pools of distinct *keys*; if a key (or many keys) are shared between two nodes, a secure connection is formed based on the key(s). In OKS, the key-string pool is a string of *bits* (not a pool of distinct keys). The bits of the key-string which overlap between two nodes are used as the shared secret key between them. As shown in Fig. 5, node A has been assigned key-string K_A and node B has been assigned K_B. If node A and node B are neighbors, they use the overlap interval of K_A and K_B as the shared secret key K_{AB} between them.

Because the key-strings are randomly assigned to each sensor node, different pairs of sensor nodes have different sizes of overlap intervals. For instance, node A and node B may have a 64-bit overlap interval that can be used as their shared-secret-key while node A and node C may only have a 16-bit overlap interval. However, it will significantly complicate the architecture of the hardware to support keys with arbitrary length. To create a fixed key size, we can simply add padding bits to create a fixed length, or we can apply a hash-function to expand the key length to fixed length. For example, any amount of key overlap can be entered into a hash function to produce a fixed key size of (say) 128 bits. The keyed-hash function can be formed by the same block cipher as for encryption, using a cipher-block chaining message authentication code (CBC-MAC) configuration.

Different key lengths have different algorithmic security levels and different key spaces. For example, a key formed from an initial 16 bits of overlap would have a lower security level than a key formed from an initial 64 bits of overlap. A higher layer protocol must be designed to handle this feature. The security level management is beyond the scope of this paper.

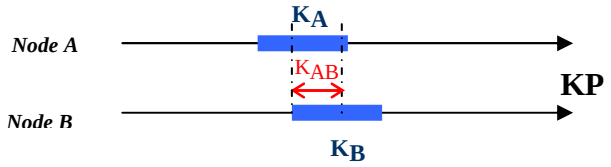


Fig. 5 Overlap-Key-Sharing protocol

According to equation Eq. 1, each node needs to have a degree of *d* such that the network is connected with high probability. If we want the network to be secure-connected, each node must have *d* secure-links with its neighbors. Let *n'* be the average number of neighbor nodes within the communication range of each node. Then probability *P'* is required to achieve the high probability that the network is secure-connected.

$$P' = d / n' \qquad \textbf{(2)}$$

Now we define *Ps* as the probability that any two sensor nodes in the network have an overlap interval. *P'* poses a constraint on the *Ps*. *Ps* should be equal to or higher than *P'*. For a specific *Ps*, we must determine what is the size of the key-string-pool (*KP*), the size of each key-string (*K*) and the number of key-strings (*R*) assigned to each node, in order to generate *Ps*. As shown in Appendix, *Ps* is derived as a function of the *KP*, *K* and *R* as:

$$Ps = 1 - \frac{KP(KP - 2R * K)^{2R-1}}{(KP(KP - R * K)^{R-1})^{2}} \qquad \textbf{(3)}$$

This equation can be used as follows: a desired probability *Ps* is given as a specification. For a fixed key-string-pool size (*KP*) and number of the key-strings assigned to each node (*R*), this equation can determine the required length *K* of the key-string stored on each node. Fig. 6 shows the total number of key-strings *R* that are stored in each node for different network sizes. We use the assumption that each node has 40 neighbors. For example, a requirement of 20 secure links for each node requires *Ps* to be larger than 0.5. Given a network with 10,000 nodes, with each node having *R*=6 key-strings, each key-string needs to be *K*=95 bits and a total of 570 bits are needed for six key-strings. Capture of one node reveals only 5.7% of the total key-string-pool as opposed to 100% of the keys in master-key protocols.

After nodes are deployed, the *key-discovery phase* is performed as nodes broadcast information about which keys they own and shared keys among neighbors are identified. This phase is a bootstrap procedure when the sensor network is first deployed.

In terms of memory size requirements, OKS scales very well with the size of the network. Only a few more bits are needed when the network size is ten times larger. Assigning more key-strings to a sensor can achieve smaller memory size to store the key-strings.

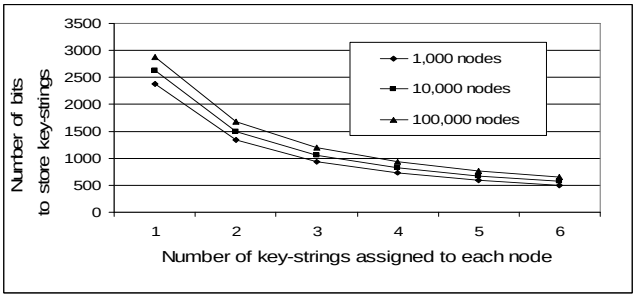


Fig. 6 Number of bits to store key-strings for different network sizes

As we mentioned in Section 3, for the OKS protocol, different pairs of sensor nodes might have different lengths of overlap intervals of their key-strings. For security reasons, we may require that the overlap interval must be longer than a certain length, which we call the *overlap threshold* (OV). Fig. 7 shows the total lengths of the key-strings *K* for different OVs. The higher the OV, the longer the key-string length that is required. However, from the figure we see that these additional memory requirements are not significant.

This also demonstrates the flexibility of Overlap-Key-Sharing. Based on different security levels and applications, OKS protocol can be easily tuned to meet the requirements. Moreover, adding a new node into the network is straightforward for OKS protocol. New nodes are deployed, and as long as the node degree constraint addressed in section 3 is satisfied, new nodes are expected to have overlap intervals with their neighbors. Hence the new node can construct *secure-links* with its neighbors.

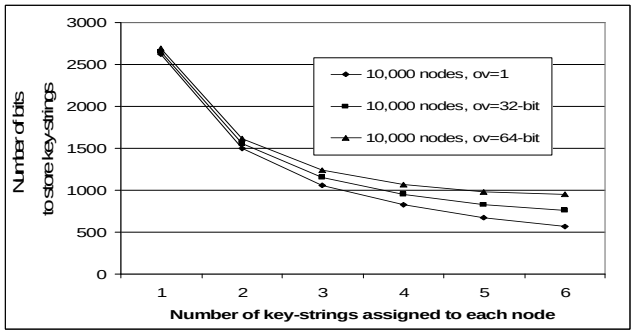


Fig. 7 Number of bits required to store the key-strings for different overlap thresholds

4. Energy Reduction, Resource Requirements, and Scalability

By examining the energy costs of security in a sensor network, it is clear the largest part of energy consumption is radio transmission [8]. In the master-key-based protocol SPINS, the radio transmission aspect of the protocol uses over 97% of the total security energy, while the actual encryption requires less than 3%. Studies in [9] show that for sensor networks, the energy/bit expended in radio transmission is three orders of magnitude greater than the energy/bit expended in AES symmetric-key encryption. Hence, minimizing data transmission cost in the security architecture is the key design goal. Here we compare the radio transmission cost of key management protocols of sensor networks and demonstrate how BROSK and OKS

significantly reduce security energy requirements at the protocol level.

4.1 Master-Key-Based Protocols

We now compare the numbers of data transmissions that are needed to negotiate session keys for different protocols. Due to the uncertainty of the wireless channel, many factors have impact on the number of transmissions, e.g. noise interference, channel fading, and even signal collisions with another neighbor node. However, for simplicity, we assume all the data can be transmitted successfully. We use the number of the transmissions to evaluate the energy consumption of the radio for each protocol.

4.1.1 Energy Consumption

A simple N by N grid topology (Fig. 8) is used for comparison. We assume a sensor node uses omni-directional antenna and each node can hear the data transmitted by the nodes immediately around it, which means most nodes have eight neighbors. The comparison is based on two scenarios: Scenario-A is when sensor nodes have to negotiate a session key with each of their neighbor nodes. Scenario-B is when only the nodes on single, regular data path need to have shared session keys. The path used is from lower right to upper left (as shown in Fig. 8). Both of the cases are extreme cases and the normal cases will usually reside in between. Fig. 9 shows the average number of

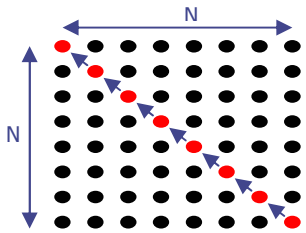


Fig. 8 Grid topology

transmissions per node for Scenario-A, while Fig. 10 is for Scenario-B.

For Scenario-A, the average transmissions for nodes in SPINS and C&R are about 8 and 12 respectively, while BROSK only needs one. SPINS has fewer average transmissions than C&R because there are only two data transmissions for a certain wireless link, while C&R needs three data transmissions for a link. SPINS alleviates the transmission load for sensor nodes by shifting the load to the server side. However, this makes SPINS non-scalable, an issue discussed in the next section. For Scenario-B, the average transmissions for nodes in SPINS and C&R are about 2 and 3 respectively, compared with only one transmission per node in BROSK. Hence BROSK can save up to 12X the amount of required transmission energy.

4.1.2 Scalability of Master-Key-Based Schemes

In terms of the number of transmissions for key management protocols, both BROSK and C&R scale very well with the size of the network. As mentioned in the section 4.1.1, SPINS shifts the transmission load to the server side. From Fig. 9 and Fig. 10, we can see the number of transmissions for the server in SPINS is several orders of magnitude larger than the transmissions of the other protocols. The problem is exacerbated by the exponential increase of transmissions for the server. Although normally the server has more resources, high transmission loads can still be a huge burden for the server in terms of computation and transmission and make the server the bottleneck of the network. This feature makes SPINS very non-scalable with the size of the network.

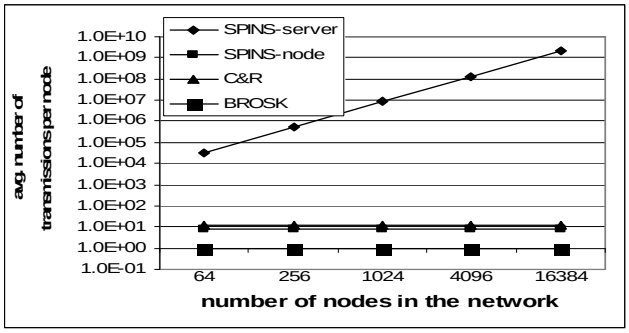


Fig. 9 Average number of transmissions for Scenario-A

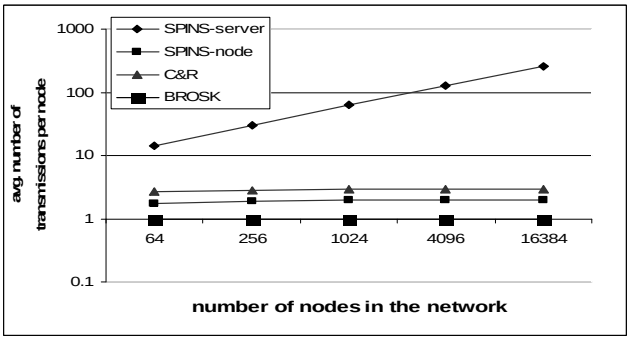


Fig. 10 Average number of transmissions for Scenario-B

4.2 Distributed-Key-Based Protocols

4.2.1 Energy Consumption

We now consider energy consumption for distributed-key-based protocols. For a graph connectivity of 0.99999 and a key-pool size of 10,000, the EG protocol requires 75 keys to be stored in each node. For OKS protocol, we assume there are R=3 key-strings on each node. We also assume that each key is K=64 bits and each key identity is 14 bits. The key identity is the address of the key within the key-pool. These identities (rather than the keys themselves) are what are broadcast to neighbor nodes to establish awareness of key overlap.

The energy required when the key identity information is broadcast to the neighbor nodes during the *key-discovery phase* is proportional to the amount of information that needs to be transmitted. The EG protocol needs to broadcast the 75 identifiers of 14 bits each, resulting in a transmission of 1050 bits. The OKS protocol (with OV = 64) only needs to broadcast the *starting* identities of each of its key-strings, since the remaining identities are implicitly known from the starting identities. In this example, there are three key identities, each of 14 bits, which is a transmission of 42 bits total. This results in a potential transmission energy reduction of 96%. This difference in energy consumption can be significant for mobile nodes, because mobile nodes have to initiate the *key-discovery phase* when entering a new area and keep broadcasting their key identities.

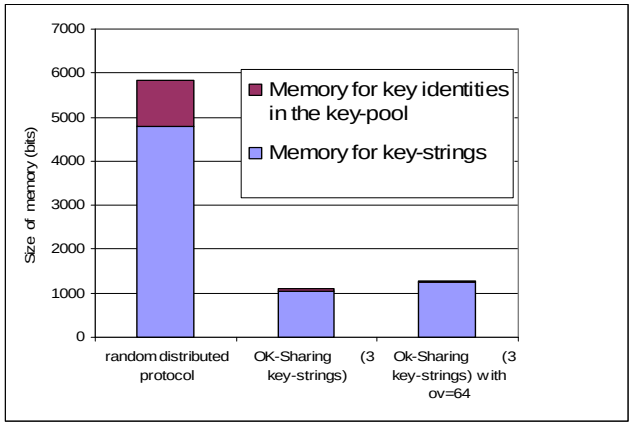


Fig. 11 Comparison of memory requirements of the random distributed protocol (EG) and OKS

4.2.2 Memory Requirement

For EG protocol, the total memory required for key storage is 5850 bits. In contrast, the OKS protocol (without OV) requires approximately 1100 bits and the protocol (with OV = 64) requires approximately 1300 bits. Therefore, a memory requirement reduction of at least 78% is achieved. Fig. 11 shows the memory space requirements of the protocols.

5. Conclusion

In this paper, we examined three previously published sensor network security schemes, SPINS and C&R for master-key-based schemes, and EG protocol for distributed-key-based schemes. We then presented two new low-power schemes, which we call BROSK and OKS as alternatives to master-key-based schemes and distributed-key-based schemes, respectively. Compared to SPINS and C&R protocols, BROSK can reduce energy consumption by up to 12X by reducing the number of data transmissions in the key negotiation process. Compared with EG protocol, OKS reduces energy by up to 96% and reduces memory requirements by up to 78%.

In addition to low power, scalability is extremely important to sensor networks. We have shown that BROSK and OKS are very scalable to the size of the network. Server based schemes, e.g. SPINS, are non-scalable for a large network size. Other preferred features for sensor networks, such as easily adding new nodes and supporting mobile nodes, are also important when considering the protocols of sensor networks.

6. Acknowledgements

The authors would like to acknowledge the support of the National Science Foundation (Grant CCR-0098361) and the Fannie and John Hertz Foundation (DH).

Appendix A. OKS Mathematical Derivation

This section will give the mathematical derivation of Overlap-Key-Sharing protocol. For a given size of the network, decisions of the parameters such as the length of key-string and number of key-strings can be made in order to achieve a high probability that the network is secure-connected.

Given a key-string pool *KP*, the size of each key-string *K*, and the number of the key-strings *R* that are assigned to each sensor node, we have to calculate the probability *Ps*, which can be formulated as equation A.1.

$$Ps = 1 - \Pr[\text{ two nodes have no overlap interval }] \qquad \textbf{A.1}$$

In order to calculate the probability that two nodes have no overlap interval, we have to know how many different cases we can find that multiple key-strings have no overlap interval on the key-string-pool *KP*. If we have *R* different key-strings on *KP*, then the number of different cases N[R] that these key-strings have no overlap interval is the following:

$$\mathbf{N[R]} = \frac{\mathbf{KP}}{\mathbf{R}} \sum_{A_1=1}^{\mathbf{KP-R*K}} \sum_{A_2=1}^{A_1} \bullet\bullet\bullet \sum_{A_{R-1}=1}^{A_{R-2}} A_{R-1} \qquad \textbf{A.2}$$

When KP is large, A.2 can be approximated by:

$$N[R] \cong \frac{KP}{R} * \frac{(KP - R*K)^{(R-1)}}{(R-1)!} \qquad \textbf{A.3}$$

Now we can calculate *Ps* as the following equation:

$$\begin{aligned} &Ps = 1 - \Pr[\text{ two nodes do not have overlap interval }] \\ &= 1 - \frac{(\#\text{ of cases that 2R key - streams have no overlap})}{(\#\text{ of cases to choose R different key - streams from the KP})} \\ &= 1 - \frac{\binom{2R}{R} N[2R]}{N[R] * N[R]} \\ &= 1 - \frac{KP(KP - 2R*K)^{2R-1}}{(KP(KP - R*K)^{R-1})^2} \end{aligned} \qquad \textbf{A.4}$$

References

[1] A. Cerpa *et al*., "Habitat monitoring: Application driver for wireless communications technology," *Proc. ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.

[2] C. D. Kidd *et al*., "The Aware Home: A living laboratory for ubiquitous computing research," *Proc. CoBuild*, Oct. 1999.

[3] W. K. Edwards and R. E. Grinter, "At home with ubiquitous computing: seven challenges," *Proc. UbiComp 2001*, pp.256-272, Sept. 2001.

[4] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," *Proc. 9th ACM Conference on Computer and Communication Security*, pp. 41-47, Nov. 2002.

[5] J. Spencer, *The strange logic of random graphs, algorithms and combinatorics 22*, Springer-Verlag 2000, ISBN 3-540-41654-4.

[6] I. F. Akyildiz *et al*., "Wireless sensor networks: a survey," *Computer Networks*, Vol. 38, No. 4, pp. 393-422, March 2002.

[7] G. Pottie and W. Kaiser, "Wireless Sensor Networks," *Communications of the ACM*, Vol.43, No. 5, pp. 51-58, May 2000.

[8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *MobiCom 2001*, pp. 189-199, July 2001.

[9] D. W. Carman, P. S. Kruss, and B. J. Matt, "Constraints and approaches for distributed sensor network security," *NAI Labs Technical Report #00-010*, Sept. 1, 2000.

[10] A. J. Menezes, P. C. van Oorschot, and S. A. Varstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[11] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proc. 2003 IEEE Symposium on Research in Security and Privacy*, pp. 197-213, May 2003.