A Cost-Efficient Counter-Intrusion Scheme for One-Time Sensor Networks

Chandana Gamage [1], Jussipekka Leiwo [2], Kemal Bicakci [1], Bruno Crispo [1], and Andrew S. Tanenbaum [1]
[1] *Department of Computer Science, Vrije Universiteit, Amsterdam, The Netherlands*
{chandag,kemal,crispo,ast}@cs.vu.nl
[2] *School of Computer Engineering, Nanyang Technological University, Singapore*
asjleiwo@ntu.edu.sg

Abstract

We propose a secure one-time sensor scheme that is highly resistant to forged messages and replay message attacks. A sensor in a one-time sensor network transmits only a single message in its life time but retransmits messages from other sensors to provide message routing. The only security-specific computational capability required from a one-time sensor in our scheme is a hash function. The bulk of security related data in our scheme is static and therefore can be stored in non-volatile memory. This is an important design criteria as energy is the most critical resource in commonly used low-cost battery-powered wireless sensors. We further improve the storage efficiency of the proposed solution using Bloom filters.
Keywords: *Sensor network security, one-time sensors, intrusion resistance, hash functions, Bloom filters*

1. INTRODUCTION

In many applications of distributed ad-hoc wireless sensor networks, intruders can cause serious damage [1]. For example, a perimeter surveillance sensor network installed to guard a large facility like an airport complex or a military encampment could have its efficacy seriously eroded if intruders are able to *repeatedly* generate false alarms as in the classic tale of the shepherd boy crying wolf. A prime reason for using wireless sensor networks is the availability of low-cost sensor nodes that can be deployed densely to achieve a high coverage as well as a greater level of fault tolerance. Additionally, the wireless communication capability of the battery-powered sensors allows for rapid establishment of a sensor network without the need for extensive infrastructure facilities such as electrical power and communication lines.

For low-cost wireless sensor network applications to be widely used, it is necessary to design security schemes that embody the application-specific security needs that reduce cost of implementation and operation while increasing resistance to attacks. The sensor network security research that we describe in this paper is focused on applications with a sensor-to-base station uni-directional messaging model [11]. This model selection was motivated by the large number of real-world applications ranging from perimeter protection to natural disaster detection that can be implemented with low-cost sensors. In contrast, we have found it quite difficult to envisage a similarly large class of peer-to-peer type of sensor network applications in which a low-cost and limited functionality sensor would have an application-level requirement to securely communicate with another similar capability sensor.

The problem that we address is intrusions into a sensor network by an attacker who either records past messages or captures sensors to raise false alarms at a base station. We present a technique that is computationally and storage efficient for use in sensor networks to defeat intruders who aim to send repeated false alarms to the base station and also to deplete the battery power of sensors throughout the network and not just at the local neighborhood of attack.

The proposed security scheme and the sensor network model on which it is based is given in section 2 with security and performance analysis in section 3. The techniques for performance improvement are described in section 4 with example values to illustrate the benefits. Related work is discussed in section 5; concluding remarks are given in section 6.

2. THE SECURE ONE-TIME SENSOR SCHEME

In this section, we first describe the model for a sensor network using one-time sensors. Then we describe our proposed security scheme.

A. The sensor and network model

We assume each sensor to be configured with a unique identifier bit-string id uniformly and randomly selected from a large set. The sensor network, at the communication layer, operates as a peer-to-peer system with each sensor providing a message forwarding service for routing of communications. At the application layer, the sensor network operates as a client-server system with the messages originating from sensors being forwarded to a central base station for application-specific processing. These low-cost sensors (with limited chip area) are assumed to have severe constraints in processing capability, storage capacity and battery power. Furthermore, as the energy cost of transmitting a single bit is roughly equivalent to the execution of 1,000 instructions [20], it is necessary to keep the number of bits transmitted as low as possible.

0-7803-9399-6/05/$20.00 © 2005 IEEE

In the proposed sensor network model, we assume that all the sensor identities for the nodes to be used in the deployment are known at startup time and the full list of id values are stored at the base station. The base station is a device with adequate protection and it does not have storage, processing or power constraints.

The most important assumption we make is that the sensors are of *one-time* use. For example, sensors used for applications such as forest fire detectors, biological/chemical warfare agent detectors, or pressure-based ground surveillance detectors would have a sensor element that would operate correctly only once. Its activation would render the sensor element unusable without recalibration or replacement. However, the wireless communication element of the sensor would continue to operate by providing message relaying towards the base station. Furthermore, we do not assume any form of tamper-resistance for these low-cost sensors and therefore if an intruder were to capture a sensor, he will have full knowledge of all its content including unique identifier value id and any other security-related data.

B. Proposed solution

We consider a network of one-time sensors that report to a base station. An attacker has the following main objectives: (1) To raise a (false) alarm to which the base station responds positively so that its resources are depleted and (2) to make sensors in the network repeatedly process and retransmit alarm messages so that their fixed energy store is exhausted.

Consider a sensor network of n nodes with the id values randomly and uniformly selected from a large set of size N with $n \ll N$. The message transmitted by a one-time sensor is formatted with a header consisting of an index value (idx, from $0 \ldots n-1$) and identifier value (id, from $0 \ldots N-1$) pair. As this is simply an alarm message generated by a one-time application-specific sensor, the message can be implicit with a null-length payload. Thus, the total header length of $\log_2(n) + \log_2(N)$ bits will equal actual transmitted message length.

Let's assume that each sensor in a network of n nodes is configured with a full list of id values for the nodes in the network by storing the values $\texttt{HASH}(idx \parallel id)$ in a list $\texttt{L}$[1]. It also has a bit string $\texttt{S}$ of length n with each bit initially set to value 0. As the index values are a consecutive series $0 \ldots n$, the list $\texttt{L}$ needs to store only the computed hash values with the index being implicit.

Now, when a sensor receives a message, with a header in which the index and identifier value pair is (idx_τ, id_τ), before retransmitting the message, the sensor would first check if the idx_τ^{th} bit in the string $\texttt{S}$ is set to the value 1. If the value is not set, then it checks in its list $\texttt{L}$ if the identifier value in the message has a match by computing the hash value for the received value and directly matching at location idx_τ in $\texttt{L}$. If a match is found then the sensor would set the idx_τ^{th} bit in $\texttt{S}$ to 1 and the message is retransmitted according to the routing rules applicable to the sensor network. Otherwise, it is assumed that the message was a copy received due to multipath routing or injected into the network by an intruder and simply discarded.

[1]Although it is sufficient to compute a simpler $\texttt{HASH}(id)$ value list for storing in a sensor, it prevents an optimization that we describe later. Therefore, the value is computed as $\texttt{HASH}(idx \parallel id)$ with additional redundancy.

3. Analysis

In this section, we first analyze how the proposed security solution can completely prevent certain attacks on a one-time sensor network and how some of the other attacks are deterred either by preventing its propagation or by causing the intruder to incur a high attack cost. Then we analyze the extra computational and storage requirements imposed on a sensor by the proposed solution. We continue the analysis in section 4 by presenting a scheme that can provide a trade-off between security and performance that does not significantly weaken the security strength.

A. Security analysis

1) Replay attacks.: An intruder who enters the sensor network will not be able to transmit messages generated by himself as he will not have valid id values to use. Therefore, the intruder has to monitor the traffic and capture some messages. An intruder cannot mount a successful replay attack in the same area from which he has captured a message as each local sensor will have the corresponding id value flagged in string $\texttt{S}$ as having already transmitted its message. If the intruder moves to a new area and replays the message, then it will be forwarded to the base station. However, as the original message would have reached the base station already with a high probability[2], the replayed message will be discarded there as a copy forwarded due to multipath routing. As it is only a replayed message and is duly discarded, this attack does not result in an application-context damage.

Therefore, a static intruder can mount a power exhaustion attack *only* on immediate neighbour sensors by forcing them to repeatedly perform message receive (RX), hashing and list matching. However, these operations require much less battery power than the more energy intensive message transmission (TX), which the static intruder has to incur. Only heuristic measures such as an exponential back-off on responding can protect against this type of power exhaustion attacks on sensor nodes in which an intruder repeatedly engages a sensor in communication with a view to exhaust its battery power. Another possibility is for the sensors to use technology for power harvesting[3] from the received signal for the RX part of the processing, similar to how RFID tags operate.

A powerful roaming intruder can cause the losses due to power exhaustion attack to increase by moving through the

[2]We do not consider as practical the scenario in which an attacker selectively jams sensor network traffic to prevent legitimate messages being received at the base station while invalid messages are let through.

[3]There are many techniques such as energy scavenging [17], energy hunting [21] and RF energy harvesting [15] to obtain ambient energy for the operation of low-power devices. Unlike RFID chips that operate purely from RF energy harvesting, one-time sensors may use a mix of own battery energy and externally sourced energy.

network and replaying messages that would get retransmitted towards the base station. In this attack scenario, the intruder has to expend an even higher amount of energy to both transmit messages and to move around the network.

2) Forgery attacks.: For an intruder to successfully attack the base station and cause application-context damage (such as raising a false biological weapons attack alarm), he has to first obtain a valid id. The only method available to do this other than to capture a sensor is to randomly guess. For a sensor id value length of $\log_2(N)$ bits and a sensor network of n nodes, the probability P_f of a successful guess is $P_f = n/(n \cdot N) = 1/N$. By selecting appropriate values for N (say, 64 bits) the probability of an intruder successfully forging a message that would get retransmitted to the base station and processed correctly could be made as small as required. Therefore, with a very high probability, the immediate neighbor sensors will discard the forged message without retransmission.

Similar to the replay attack, an intruder can repeatedly mount forged message attacks on a sensor causing power exhaustion. But as explained earlier, the attack will not propagate and the intruder has to expend much more energy than the sensor under attack.

3) Sensor capture attacks.: The only method by which an intruder can successfully cause an application-level damage with high probability is by capturing a sensor and then transmitting a false message. Once an intruder captures a sensor, he has full control over it and can use all its data and functionality. The proposed scheme does not provide any direct countermeasure against this type of powerful attack. However, by the very nature of the one-time sensor paradigm, the intruder can send only a single valid message using a captured sensor. Any other message transmission attempts will be either forged or replay message attacks and will be unsuccessful. Therefore, the sensor network may implement application-specific heuristics to counter this threat. These heuristics can be as complex as necessary as they are implemented at the base station or beyond and not on the sensors.

As an example, for a sensor network deployed at a large public gathering such as an international sports event to detect chemical warfare agents, the base station may not raise a public safety alarm unless messages from a cluster of at least s_t nearby sensors arrive at the base station, where s_t is the threshold value. It is clearly required for sensor network application developers to calibrate message interpretation to deal with false positives.

4) Sybil and black-hole attack.: An intruder will not be able to forge multiple valid identities if the id values for the sensors are chosen uniformly and randomly from a large pool. This is an effective counter measure against the Sybil attack [8], [16] on a sensor network. An intruder will not have any increased advantage in carrying out a Sybil attack even after capturing a set of valid sensor nodes and extracting their id values due to the one-time nature of the sensors.

A black-hole attack is semantically the inverse of a Sybil attack and occurs when an intruder takes over a sensor and continually discards received messages without ever retransmitting them. If this particular sensor is located in a message routing path that has no other alternatives, it could potentially partition the network and isolate one or more sensors from communicating with the base station. So, rather than illegally introducing multiple identities to the network, as in Sybil attack, many valid identities will disappear from the network.

A black-hole attack may be carried out by an intruder even without capturing a valid sensor if the MAC algorithm used in the sensor network link layer uses a dynamic neighbour discovery protocol [19]. For example if the MAC algorithm uses a scheme by which transmission power is incremented by a fixed quantum until a neighbour acknowledges reception, then an attacker could masquerade as a valid sensor and carry out the denial of service attack. The only effective counter measure for black-hole attacks is to deploy a sensor network with adequate density to provide multiple paths for message transmission to the base station.

B. Performance analysis

For a sensor network of n sensors with an id value length of $\log_2(N)$ bits (with $n \ll N$) and a hash function HASH with output length r bits (for example, SHA-1 with 160 bits), the total security related memory requirement at a sensor is $(\log_2(n) + \log_2(N) + nr + n)$ bits respectively for its own (idx, id) pair, list L, and control string S. However, it is strictly not necessary to store the full hash function output as the size of the network n is significantly less than the output size of this hash function. Therefore, we can fold the hash value output without any loss of randomness by simply taking the t least significant bits (for example, 64 bits) as all the bits in a computed hash value are equally random. This allows us to maintain the required $P_f = 2^{-t}$ against forgery attacks. Also, the string S is used to control network communications by preventing both multiple retransmissions and replaying of messages. Therefore, processing and storage costs associated with S should not be considered as a purely security-related cost. The index value idx of a sensor is also not solely security related as it is required to identify the sensor from which a message has originated.

Therefore, we can approximate the direct security related memory cost to be $(\log_2(N) + nt)$ bits or just nt bits for $\log_2(N) \ll nt$. For sample values of $\log_2(N) = 256$, $n = 1024$, and $t = 64$, the total security-related memory cost is approximately 64 Kbits per sensor.

For each received message, the sensor has to compute a single hash value (for $cost$(HASH)) and then do a direct match with list L with index value being the list address pointer. For a single bit-comparison cost of $cost$(BEQ), the comparison cost is $t \times cost$(BEQ). Therefore the total of strictly security-related computational cost can be approximated as $cost$(HASH) $+ t \times cost$(BEQ) $\approx cost$(HASH) as in general, $cost$(BEQ) $\ll cost$(HASH).

4. AN IMPROVED SOLUTION USING BLOOM FILTERS

Two of the main design objectives of the proposed secure one-time sensor scheme were firstly to minimize the amount of

useful information an intruder can obtain by capturing a sensor and secondly to minimize the security-related complexity of the sensor processing element. These two objectives were successfully achieved by storing a list of hashed (idx, id) value pairs on the sensor and requiring only hash computations on the sensor. These two design decisions allow us to use a classic algorithm called a *Bloom filter* [4] to further reduce the memory required to store the list L.

A Bloom filter is an algorithm to compactly store a list of keys and then efficiently search it to see if a candidate key is a member in that list. The algorithm requires that the list of keys is fixed at the startup and the membership result tolerates a certain probability of false positives. The one-time sensor network applications that we have outlined earlier can tolerate both these algorithmic limitations. As Bloom filters store the list of (idx, id) value pairs as HASH(idx, id) one-way hashes, an intruder who captures a sensor cannot recover the actual id values without performing an exhaustive search of the full (idx, id) value space. Even in such a brute force attack, the intruder cannot be certain of finding the correct id values due to the probability of false positives associated with the Bloom filter scheme.

To be successful, an intruder will require "valid" id values for forging messages for which the base station will react positively in comparison to "arbitrary" id values that pass the Bloom filter test but are rejected at the base station, which compare the received id value against the stored master list.

Construction of a Bloom filter requires use of a number of distinct (say, k) hash functions. We can satisfy this requirement by initializing the common hash function HASH with a set of k different values stored on the sensor. The Bloom filter uses a bit vector V of a suitable length m bits and the hash function output must be in the range of $0 \ldots m-1$, so that the computed hash value acts as an index to a bit position in the vector V. The bit vector is initialized by setting all bits to 0. Before the sensors are deployed, the Bloom filter is computed and stored on them by running each of the concatenated $idx \parallel id$ values for all n sensors through the k hash functions and setting the bit in V corresponding to the hash value to be 1. If a bit at a particular position is already set to 1, it remains unchanged.

After the sensor network is deployed and when a sensor receives a message with header index value and identifier value pair (idx_τ, id_τ), the membership in the Bloom filter is checked by computing the k hash values for the concatenated input $idx_\tau \parallel id_\tau$ and checking to see if the corresponding bit positions in V is set to 1. If any of the bits remains as 0, then the id_τ value does not belong to the list of valid identifiers and indicates a fake message. If all the corresponding bits are set to 1, then the id_τ value can be accepted as a valid member with a high probability. The classic original paper on Bloom filters [4] gives the probability of a false positive acceptance P_c as

$$P_c = (1 - e^{-\frac{kn}{m}})^k \tag{1}$$

We need to choose k and m so that P_c is at an acceptable level based on the sensor network application heuristics. As

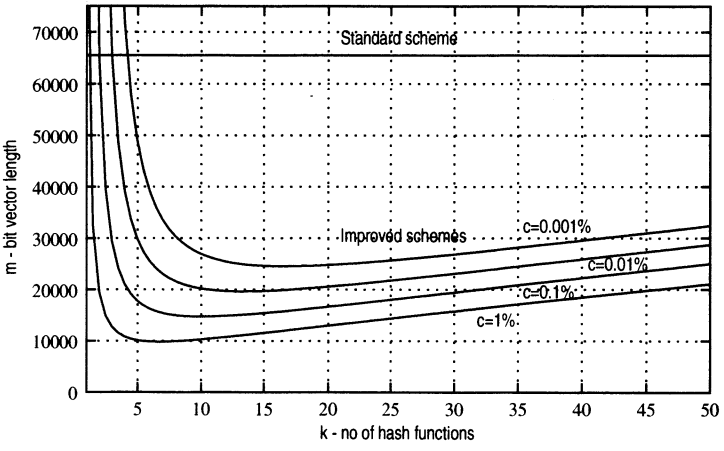

Fig. 1: The optimal number of hash functions (k) and bit vector length (m) for different false positive probabilities (P_c) in a sensor network of $n = 1024$ nodes

we would be setting the probability of false positives P_c and number of hash function initializers k, the size of the bit vector V stored on each sensor can be given as

$$m = \frac{-kn}{\ln(1 - P_c^{\frac{1}{k}})} \tag{2}$$

The total security-related memory required under the Bloom filter improved scheme is $(\log_2(n) + \log_2(N) + m + k\log_2(m) + n)$ bits respectively for (idx, id) pair, vector V, the k number of HASH function initialization values, and the string S. As before, we can exclude the memory required for the retransmission control string S and index value idx from this purely security-related cost. This memory cost can be further approximated as m bits by removing the cost of fixed id value of the sensor and the very small memory requirement of $k\log_2(m)$ for hash function initializers. For a sample value of $n = 1024$, $P_c = 0.001$, and $k = 5$, the security memory cost is approximately 17 Kbits per sensor, which is a significant saving from the earlier non optimized scheme cost of 64 Kbits for $t = 64$ bits. The graph in figure 1 illustrates the possible performance improvements in memory usage.

This memory saving is achieved at a cost of false positive probability of P_c (for the example values above, $P_c = 0.001$) and additional computational costs in hashing. For each message received at the sensor, hash values must be computed k times and bit comparisons done k times for a total cost of $k\ cost(\text{HASH}) + k\ cost(\text{BEQ}) \approx k\ cost(\text{HASH})$ which compares favorably for small k with the non optimized scheme cost of a single $cost(\text{HASH})$.

The specific number of hash functions (k) to be used with Bloom filter optimization technique should be determined in conjunction with the relative cost of computing a hash value against the acceptable level of false positives as these in turn trigger message forwarding through the network towards the base station. As sensors are powered by fixed-capacity batteries, it is important to note that when considering the cost of computations in sensor networks, it is more pertinent to consider the actual energy cost per operation than the number

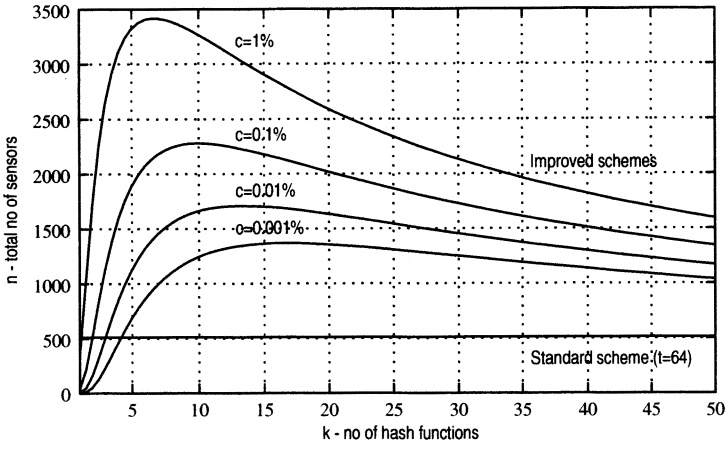

Fig. 2: The total number of sensors a network can support for different false positive probabilities (P_c)cost for a fi xed bit vector size ($m = 32$ Kbits or 4 KBytes)

of processor cycles to complete an operation.

In practice, a sensor manufacturer would fix the amount of memory available for users. Therefore, it is interesting to check the total number of sensors in a network that can be supported by a given fixed amount of memory available for implementing the proposed security scheme. The graph in figure 2 shows the total number of sensors that is possible for a sensor with 4 KBytes of memory for different false positive probabilities that an application could tolerate. The graph shows that for $P_c = 0.001$, the maximum size of the sensor network is approximately 2300 nodes at $k = 9$.

A. Colluding intruders

The probability of false positives (P_c) in the Bloom filter based one-time sensor network security scheme has an interesting benefit. Let us assume, for example, that an international sports event is being protected against attacks with chemical warfare agents and that the security administrators have decided on a threshold of s_t alarm messages before activating public emergency services. Let us also assume that an intruder to the sensor network that has captured s_c number of sensors ($s_c < s_t$) has found out this threshold value and is negotiating with some other intruders to obtain the remaining number of sensor id values to mount a false alarm attack.

This being a transaction among less than honorable parties, the first requirement is for the sellers to prove to the buyer that they have possession of some sensors without revealing the actual id values of the sensors they have until the sales negotiation is complete. Now, if a seller computes the set of k hash function values for the sensor id he has and sends them to the buyer, the buyer can convince himself of the truthfulness of the sellers claim of possession of a sensor with only a probability of $(1 - P_c)$. Therefore, an intruder who already has access to the Bloom filter cannot prove with absolute certainty to another intruder who also has access to the same Bloom filter of its possession of a valid sensor, thus potentially complicating transactions among colluding intruders. The second requirement, which is more important, is for the buyer to be certain that the id value given by the seller is a valid identifier that would be accepted by the base station. As the intruder who is offering to sell an id value can query its Bloom filter with arbitrary values to find a (idx, id) pair that would pass the membership test, the buyer would have no guarantee that it is a valid id.

B. Offline oracle attacks

An intruder who captures a secure one-time sensor, optimized with a Bloom filter, can use it as an oracle to create a list of "arbitrary" (idx, id) value pairs for which the membership test succeeds without having to use a brute force attack to determine "valid" (idx, id) value pairs. Depending on the probability of false positives the Bloom filter was configured to accept, the intruder can query the captured sensor repeatedly with random (idx, id) value pairs to buildup a set of values that pass the test. For example, a $P_c = 0.001$ probability would allow the intruder to build a list with 100 values that pass the membership test by simply querying 100,000 times, which is a relatively small effort.

Now, the intruder can use these arbitrary (idx, id) value pairs to forge messages that would be correctly accepted and forwarded by other sensors towards the base station. While there would be no application-level damage caused as the base station would detect these values to be bogus by comparing with the original set of (idx, id) pairs it stores, the sensors in the network would be subject to a serious power exhaustion attack. We can mitigate this offline oracle attack by computing a set of distinct Bloom filters (for example, by using different hash initialization values) and randomly selecting a Bloom filter to be stored in a particular sensor. In the extreme case, each sensor could be stored with its own unique Bloom filter. As the computation of Bloom filters and then configuring the sensors with them is done offline prior to the actual deployment of a sensor network, the costs associated with this solution are not a significant factor.

This multiple Bloom filter approach would greatly reduce the success probability of an intruder, who captures a sensor and computes a list of acceptable (idx, id) pairs, being able to simply transmit messages constructed using the list and have those messages accepted and retransmitted by other sensors. This is an effective countermeasure against the power exhaustion attack.

5. Related work

The concept of one-time sensors was introduced by Bicakci, et al. [2] with protection against node capturing attacks provided using several techniques including public key cryptography and Merkle hash trees [14]. This concept paper [2] explains the usefulness and justification for one-time sensor network applications. The Bloom filters were originally introduced for use in efficient compression and fast searching of data structures such as dictionaries and were later found wide use in database applications. In recent times, Bloom filters have been used in many network applications as indicated in the survey by Broder and Mitzenmacher [6].

One of the first security protocol suites for sensor networks was the SPINS architecture by Perrig, et al. [18] that was based on a model of sensor-to-base station communication. For peer-to-peer secure communication between sensors, a shared key was established using the base station as a TTP. Recent research work on sensor network security has focused on peer-to-peer secure communication by sensors and the most widely proposed security scheme for conventional sensor networks has been pair-wise key predistribution. A *probabilistic* hop-by-hop key establishment protocol was proposed by Eschenauer and Gligor [10] where each sensor is preinstalled with a randomly selected subset of keys from a large key pool. A secure communication path between two sensors is established by selecting a set of intermediary nodes with a single pair-wise shared key between each sensor pair.

This scheme by Eschenauer and Gligor has the weakness that an attacker who is capable of capturing a small set of sensors is able to determine a large number of end-to-end paths. The work by Chan, et al. [7] was directed towards reducing this problem by requiring a sensor to have more than one pair-wise shared-key with a neighbour to establish a hop in the end-to-end path. The random pair-wise key establishment scheme of Du, et al. [9] is based on the key predistribution scheme of Blom [3] and the similar scheme of Liu and Ning [13] was based on the two-party key establishment for dynamic groups of Blundo, et al. [5] combined with multiple key spaces.

We do not make a direct comparison of our proposed sensor network security scheme with other work cited above for two reasons: (1) the security model used by us is based on a one-time sensor concept while all the previous work consider multi-use conventional sensors and (2) the main objectives of previous security proposals have been to provide message confidentiality (by encryption) and integrity (by MAC) using the shared keys and prevent exposure of a shared key established using the particular protocol to an attacker while we focus on countering an intruders ability to raise false alarms by using message forgery, message replay or node capture techniques.

6. Conclusion

We have presented an efficient security scheme for one-time sensor networks with performance improvements using Bloom filters. Both the proposed security scheme and the use of Bloom filters in sensor network security are new contributions in this evolving research area. Importantly, the use of Bloom filters to reduce memory requirements on the sensor does not increase the size of messages and thus keeps the important cost of bit transmission fixed. This is an important factor, as message transmission constitutes a major portion of sensor energy consumption. The feasibility for the one-time sensor network applications to tolerate false positives (that is, allowing a forged message to reach the base station) rather than false negatives (that is, disallowing a correct message to be transmitted within the network) is a key element in the applicability of the proposed Bloom filter based security scheme to real-world scenarios.

The research result presented in this paper, based on the novel one-time sensor idea, is part of an on-going effort to develop efficient and secure algorithms for low-cost sensor networks. To increase the range of sensor network applications that can utilize the basic idea introduced in this paper, we have extended the algorithm to a k-time sensor scheme [12] and at present we are setting up an experimental network to evaluate its energy use and other properties.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *A survey on sensor networks*, IEEE Communications Magazine, 40(8):102-116, IEEE, 2002.

[2] K. Bicakci, C. Gamage, C. B. Crispo, and A. S. Tanenbaum, *One-time sensors: A novel concept to mitigate node-capture attacks*, European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS), LNCS, Springer, 2005.

[3] R. Blom, *An optimal class of symmetric key generation systems*, EUROCRYPT 1984, LNCS 209, pages 335-338, Springer, 1985.

[4] B. H. Bloom, *Space/time trade-offs in hash coding with allowable errors*, CACM, 13(7):422-426, ACM Press, 1970.

[5] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, *Perfectly secure key distribution for dynamic conferences*, CRYPTO 1992, LNCS 740, pages 471-486, Springer, 1993.

[6] A. Z. Broder and M. Mitzenmacher, *Network applications of Bloom fi lters: A survey*, Internet Mathematics, 1(4):485-509, 2003.

[7] H. Chan, A. Perrig, and D. Song, *Random key pre distribution schemes for sensor networks*, IEEE Symposium on Security and Privacy, pages 197-213, IEEE Computer Society, 2003.

[8] J. R. Douceur, *The Sybil attack*, Workshop on Peer-to-Peer Systems, LNCS 2429, pages 251-260, Springer, 2002.

[9] W. Du, J. Deng, Y. S. Han, and P. Varshney, *A pairwise key predistribution scheme for wireless sensor networks*, ACM Conference on Computer and Communications Security (CCS), pages 42-51, ACM Press, 2003.

[10] L. Eschenauer and V. D. Gligor, *A key-management scheme for distributed sensor networks*, ACM Conference on Computer and Communications Security (CCS), pages 41-47, ACM Press, 2002.

[11] J. Elson and D. Estrin, *Sensor networks: A bridge to the physical world*, Wireless sensor networks, pages 3-20, Kluwer Academic Publishers, 2004.

[12] Reference blinded for review, *Countering SPAM and DOS attacks in a sensor network*, 2005.

[13] D. Liu and P. Ning, *Establishing pairwise keys in distributed sensor networks*, ACM Conference on Computer and Communications Security (CCS), pages 52-61, ACM Press, 2003.

[14] R. C. Merkle. *Protocols for public key cryptosystems*, IEEE Symposium on Security and Privacy, pages 122-134, IEEE Computer Society, 1980.

[15] M. H. Mickle, M. Lovell, L. Mats, L. Neureuter and D. Gorodetsky, *Energy harvesting, profi les, and potential sources*, International Journal of Parallel and Distributed Systems and Networks, 4(3):150-160, ACTA Press, 2001.

[16] J. Newsome, E. Shi, D. Song, and A. Perrig, *The Sybil attack in sensor networks: Analysis and defenses*, ACM Symposium on Information Processing in Sensor Networks, pages 259-268, ACM Press, 2004.

[17] J. A. Paradiso and T. Starner, *Energy scavenging for mobile and wireless electronics*, IEEE Pervasive Computing, 4(1):18-27, IEEE Computer Society, 2005.

[18] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, *SPINS: Security protocols for sensor networks*, Wireless Networks, 8(5):521-534, Kluwer Academic Publishers, 2002.

[19] J. Polastre, J. Hill and D. Culler, *Versatile low power media access for wireless sensor networks*, ACM Conference on Embedded Networked Sensor Systems (SenSys), pages 95-107, ACM Press, 2004.

[20] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava, *Energy-aware wireless microsensor networks*, IEEE Signal Processing Magazine, 19(2):40-50, 2002.

[21] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann and D. Estrin, *Studying the feasibility of energy harvesting in a mobile sensor network*, IEEE International Conference on Robotics and Automation, pages 19-24, IEEE, 2003.