A Per-Hop Security Scheme for Highly Dynamic Wireless Sensor Networks

Syed Taha Ali, Vijay Sivaraman and Ashay Dhamdhere
School of Electrical Engineering and Telecommunications
University of New South Wales
Sydney, NSW 2052, Australia
Email: (taha@student., vijay@, ashay@)unsw.edu.au

Abstract

Certain popular wireless sensor network applications, including disaster recovery, battlefield communication and athlete monitoring, are characterized by extensive node mobility, intermittent contact between nodes and a highly dynamic network topology. Traditional routing protocols and security schemes are designed for essentially static networks and do not perform well in this case. This has given rise to a new multi-hop routing paradigm, that of "mobility-assisted" routing in which nodes make strategic data store-and-forward decisions on a per-hop basis. In this paper we discuss the security challenges relevant to mobility-assisted routing and propose a scheme to secure data communication between nodes in highly mobile sensor networks. Our solution utilizes symmetric-key encryption to ensure data confidentiality and varies encryption key in a verifiable, non-forgeable manner to allow easy authentication. This scheme also provides data freshness, semantic security and per-hop encryption to enable secure data aggregation. To validate our basic assumptions and fine-tune our scheme, we collect and analyze link connectivity statistics from a dynamic sensor network application, athlete monitoring during a first-division university soccer club match. We show that our scheme is well-suited for certain dynamic environments and serves as an effective first step towards securing communications for mobile sensor networks.

1. Introduction

Wireless sensor networks (WSNs) have evolved from traditional isolated applications (habitat monitoring, preventive maintenance) to take on a more people-centric focus (body sensor networks, vehicular networks) and this has led to new challenges in network design. Securing sensor networks is a popular research area and there is a plethora of mechanisms which address various aspects of network vulnerability. Almost all these schemes, however, operate on the assumption that sensor nodes are stationary and network topology is static for the most part. This assumption is contrary to certain high profile and highly dynamic applications that include disaster recovery, healthcare, athlete monitoring and battlefield communications where node mobility is essential to the application.

We envisage a scenario where sensor nodes are highly mobile within the network and only come in intermittent contact with other nodes. Link connectivity is sparse and routes are constantly changing. Complete paths from source to sink do not exist most of the time. Traditional routing mechanisms are highly inefficient when faced with unpredictable route changes and network splits and joins. In the domain of ad hoc networks, this has motivated proposals for a new routing paradigm, that of "mobility-assisted" routing [1] which operates on the principle that end-to-end paths-over-time may exist from source to destination and data can be effectively delivered by making local store-and-forward decisions at the nodes themselves. Messages could be sent over one link and stored at the next hop till another link comes up in the path and then forwarded and so on till the destination is reached. Securing communication over this underlying routing mechanism poses a research challenge.

A highly active zone such as a disaster recovery site [2] is a good example of a dynamic network application: wireless sensor devices would be worn by first responders, firefighters, etc. to monitor their vital signs and transmit essential information, e.g. location [3]. Sensors would also be deployed on the injured for triage purposes. Paramedics could carry PDAs or hand-held devices to function as mobile base-stations. The network would be in a constant state of flux, most nodes would not know of definite routes to the sinks, they would transmit data to the nearest nodes in range which would buffer and forward it on in turn.

There is a strong trend towards securing privacy in real world applications, especially in the case of medical data. The need for security is also apparent if one considers a battlefield deployment where data is of critical importance and the potential for attack on the network is far greater. If military personnel are equipped with WSN devices on the field to monitor condition, location or simply enable communication, it would be reasonable to assume a high level of activity and unpredictable link connectivity.

Athlete monitoring [4] similarly presents a highly dynamic network application. There have been efforts to deploy miniature devices to remotely monitor sports such as cycling [5] and rowing [6]. To motivate our security solution,

we take up the example of a soccer game in which players wear small wireless sensor devices. These devices, worn on the soccer field, form a network and use multiple hops to route vital information, such as player heart rate, body temperature, velocity, etc. back to the base-station where it can be viewed in real time. This data is invaluable for athlete training, monitoring player performance and preventing injury.

This scenario presents considerable security challenges. Players would be moving very fast and there will be frequent falls and physical contact. Network routes would last for mere seconds before being disrupted. Per-link encryption keys clearly cannot be used. A global key, shared by all nodes, addresses data confidentiality, but is hard to protect and allows nodes to impersonate as others. Implementing pairwise keys between nodes requires a neighbour-discovery protocol for communication and restricts the node to unicast encryption and limits routing efficiency. Giving nodes unique keys which they share with the base-station for end-to-end encryption does not permit data aggregation and exposes the network to battery-drain attacks.

In our scheme, every device in the network essentially uses symmetric key cryptography with time-varying keys from a one-way key-chain to enable per-hop encryption of data. Encrypting the data ensures **confidentiality** and the inherent one-way structure of the key-chain enables **authentication**: new keys can be verified by receivers as to their source on the basis of older ones in the chain, but cannot be forged. Encrypting on per-hop basis allows nodes to verify and **aggregate** incoming data thereby limiting attacks and reducing wireless overhead. Additionally, by exploiting the limited-broadcast nature of sensor networks, our security scheme can be easily adapted to support more advanced mobility-assisted routing mechanisms [7] such as controlled flooding, single-copy [8], spraying techniques [9] [10], etc.

The contributions of this paper are:

- we list operating assumptions a security scheme for dynamic wireless sensor networks must take into account and support them with experimental results from a real soccer match deployment
- we propose a scheme to enable secure communication of data as an ambitious first step
- we demonstrate how our scheme can be tuned as per application scenario using link-connectivity statistics

The rest of the paper is organized as follows: Section 2 details our soccer match deployment, the threat model, solution requirements and summarizes previous work from literature. We discuss our security solution in Section 3. In Section 4, we analyze link connectivity statistics to show how the scheme can be configured for different mobility characteristics. We conclude in Section 5 with ideas for future work.

2. Problem Overview and Related Work

This section discusses the setup for soccer player monitoring, preliminary results, outlines the threat model, solution requirements and summarizes related work from the literature.

2.1. Experimental Setup

We attached MicaZ motes using arm-straps to players from the University of New South Wales Football Club (UNSWFC) first division mens soccer team during a pre-season trial match. The goal was to collect link connectivity statistics to enable off-line application modeling (in our case, soccer player monitoring). The game was played on a field measuring 90m x 45m. Fig.1 and Table 2.1 depict player positions. Eight base stations are mounted around the field at regular intervals to act as sinks for the data.

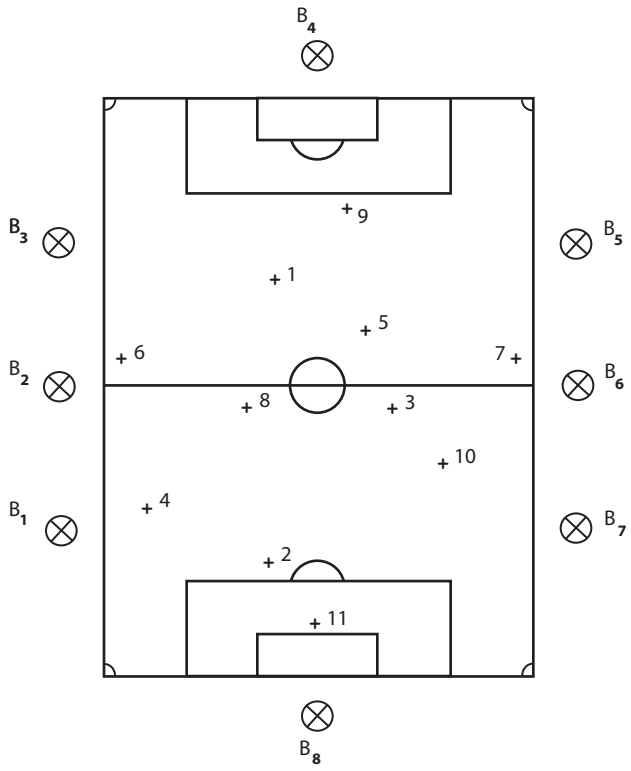

Figure 1: Player positions on the soccer field

Motes worn by the players acted as beacons and broadcast their unique ID and a sequence number at the highest available power level (1mW) once per second. A base-station synchronized the entire network before game commencement and, to prevent packet collisions, each mote was assigned a unique 90 ms slot per second for data transmission.

Mote ID	Position
B_1 - B_8	Base Stations
1	Centre Attacker
2	Back
3	Centre Midfield A
4	Left Back
5	Midfield
6	Left Wing
7	Right Wing
8	Centre Midfield B
9	Striker
10	Centre Back
11	Goal Keeper

Table 1: Table 1: Player Positions and Mote Numbers

Packet receipt was noted and time-stamped by all motes and base-stations within range. From these results, we compiled a record of node 'encounters', a time-line of which mote can be heard at what time and by which entity. Readings were collected over a 20 minute period (limited by MicaZ onboard memory). The results of this experiment are available for online viewing in a dynamic Java applet at [11]. We acknowledge that repeat trials will yield a different set of results but that, over time, key characteristics and trends can be identified, measured and anticipated.We present some initial findings:

The **number of neighbouring players** reachable by a certain player varies sharply over a given period of time. We plot in Fig.2 the neighbour count for Node 10 (Centre Back) over a period of 600 seconds and observe a high level of variation.

The **encounter duration** between two nodes can be defined as the maximal number of consecutive packets transmitted by one node and received by the other. This is equivalent to the time a pair of nodes are 'connected' (since each node transmits one packet per second). We plot encounter duration over all pairs of nodes in Fig.3 and note that almost 70% of encounter durations last 3 seconds or less. These figures clearly indicate that the network is in a constant state of flux. This is expected: sensor devices have primitive radios, they are small and lightweight so as not to restrict movement and the players move at great speed, there are frequent falls, physical contact and sudden changes in body orientation, all of which have a highly disruptive effect on connectivity.

2.2. Operating Assumptions and Threat Model

We assume a simplistic one-step data transmission process whereby a node simply broadcasts data in its allotted time-slot. To keep communication minimal, there is no handshaking or negotiation process with other nodes in the vicinity. We format the packet structure in such a way that the message can be processed by any of the other receivers in range. Such an approach allows for a transmitting node to utilize multiple paths to the destination and facilitates data aggregation and prioritization.

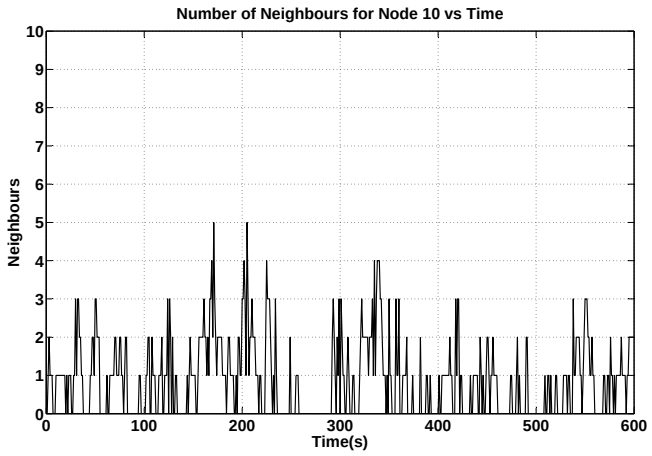

Figure 2: Number of Neighbours for Node 10 (Centre Back)

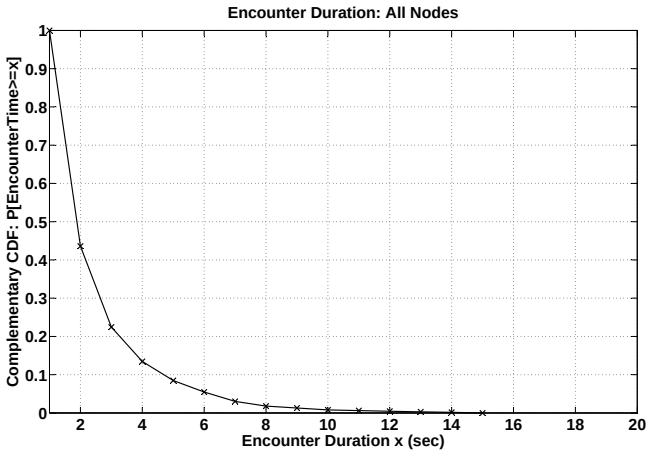

Figure 3: Encounter Duration over All Pairs of Nodes

To bootstrap trust in the network, we assume pre-distribution of keying material before the network initializes. We specify details in the description of our scheme.

We assume our adversary engages in eavesdropping, packet replay, data injection and flooding attacks. Allowing per-hop encryption limits flooding and enables nodes to identify the attack. However, if a node is captured, cryptographic keys can be extracted. In this event, confidentiality is compromised, but, our security scheme still guarantees authenticity of the broadcast source. We assume the network can defend against signal jamming by some form of frequency hopping or spread spectrum techniques. In this paper we do not expressly consider more advanced attacks such as denial-of-service, wormhole attacks, etc.

2.3. Solution Requirements

We seek a security solution that provides the following properties:

1) Confidentiality: Eavesdroppers cannot decipher the data that the nodes transmit.
2) Authenticity: Nodes in the network are able to verify the source of all messages they receive. A node should not be able to masquerade as another.
3) Replay Protection: An attacker may capture legitimate messages and inject them into the network at a later time. Receivers should be able to identify and reject captured and replayed messages.
4) Semantic Security: Encryption should sufficiently disguise the content of the messages, so that an adversary can not decipher them based on patterns in the ciphertext.
5) Data Fusion: Allowing nodes to inspect and aggregate whatever data they receive will significantly reduce network traffic and cut down on power consumption over the wireless link. Receivers would also be able to *prioritize* traffic: e.g. in a disaster recovery scenario, nodes would give higher priority to transmitting medical alarms; nodes on the battlefield could choose to transmit critical military intelligence over lesser-priority data. On the soccer field, nodes could prioritize data of certain players over others.

2.4. Prior Work

To the best of our knowledge, there is no prior scheme that addresses the issue of security for resource-constrained nodes in a highly dynamic wireless sensor network. A security solution is typically implemented over an existing underlying routing mechanism; and efficient mobility-assisted routing protocols have yet to be developed to meet the unique operating challenges of dynamic sensor networks.

Security solutions are typically designed for stationary networks with minimal route disruption and a fairly constant set of neighbours. High node mobility is more characteristic of mesh and ad hoc networks. To address routing and security challenges, these networks typically maintain dynamic routing tables and utilize route-discovery and synchronization protocols. Neighbours are easily verified using digital certificates and trusted third parties. Sensor network devices are too resource-limited to permit easy adaptation of these solutions: sensor nodes have radios with far less range, a restriction on wireless usage and not enough resources to maintain routing tables or use public-key cryptography.

Of the common security strategies, using a **global key** to secure the network (as in the case of the TinySec [12] security architecture) would facilitate mobility but such a key is difficult to safeguard in a large deployment, and it would allow a hijacked node to masquerade as any other. **Per-link keys** are explicitly meant for static networks and would be very inefficient. Giving all nodes **individual encryption keys** which they share with the base station is not very effective. Data aggregation would not be possible, there would be end-to-end encryption and a resourceful adversary could flood the network with arbitrary data. Nodes would be unable to differentiate it from legitimate traffic and waste precious resources in forwarding it. **Pairwise keys** between nodes would work for low mobility scenarios but a handshake process would be required for communication and nodes would not be able to multicast data.

Of popular security mechanisms, the μTesla [13] protocol resembles our scheme in that it assumes a time-synchronized network and utilizes a one-way key-chain to ensure source authenticity of data. However, μTesla is a broadcast authentication protocol expressly designed to authenticate base-station transmissions. μTesla uses time-varying keys to compute message authentication codes (MACs) on the data. The issue of confidentiality is not addressed. Whatever messages the node receives from the base-station, it buffers till key disclosure. The network is time-synchronized and keys are disclosed after a delay, allowing nodes to verify the MACs. The key in question expires after it has been disclosed rendering it useless to an attacker.

The MiniSec [14] communication architecture utilizes OCB encryption to provide data confidentiality and authentication. MiniSec has different modes for unicast and broadcast communication. Unicast mode assumes that each node possesses a pair of keys for bidirectional communication with another node. For broadcast communication, the encryption key is a global shared key. Both these strategies are not well-suited for mobility and carry with them the risks mentioned earlier. MiniSec also assumes network time-synchronization. Replay attacks are detected by demarcating network lifetime into 'epochs' and implementing space-efficient Bloom filters.

3. Our Solution

Nodes use successive keys from a one-way key chain to encrypt and transmit data. Encryption keys are varied on a per message basis. Considering our soccer game scenario, this would translate to a new key being used every second. The one-way nature of the key-chain ensures that receivers can verify new keys from older ones but are unable to forge them.

Packet structure is depicted in Fig.4. The Data Field, D_i contains the data to be transmitted, encrypted with the current key, k_i, from the chain. This current key is then itself encrypted with a specific older key from the chain, denoted as the 'epoch-key', k_E, and put into the Key Field.

To provide a common reference point for nodes in intermittent contact, we divide network lifetime into epochs. For

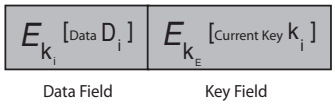

Figure 4: Packet Structure

each transmitting node, an epoch is characterized by specific epoch-keys in the chain which the node uses to encrypt the Key Field for all packets transmitted in that epoch.

A detailed description of our scheme follows. We discuss transmitter and receiver operations individually for clarity, but it must be kept in mind that devices in the field perform both functions.

3.1. The Procedure

1) **Key-Chain Generation**: Nodes build their encryption key-chain prior to network initialization in one of two ways: The receiver could be supplied a seed value, k_s from which it would generate a key-chain, k_s, k_{s-1},...,k_1, k_0, where k_{i-1} is obtained by hashing k_i for $i = 1, ..., s$ (using SHA1 or MD5). Alternatively, the entire key-chain could be constructed beforehand and pre-programmed into the device prior to deployment. This key-chain should be long enough to last the lifetime of the network.
2) **Bootstrapping Trust**: To initialize trust, each node is provided a set of keys containing the initial 'root-key' (k_0) for every other node in the network. This can be pre-programmed or transmitted to the node using a secure Diffie-Hellman exchange.
3) **Transmitting Data**: When a node has to transmit data in its slot, it creates a packet consisting of a Data Field and a Key Field as mentioned earlier. The communication process is detailed in Fig.5: data to be sent, D_i is encrypted using the current key in the chain, k_i (step 1) and passed into the Data Field. The current key, k_i, itself is then encrypted using the epoch-key, k_E (step 2) and put into the Key Field. The packet is then transmitted to the receiver (step 3).
4) **Receiving Data**: The receiver decrypts the Key Field to extract the current key, k_i using the transmitter's epoch-key k_E (step 4). To authenticate, it hashes the current key to see if it matches an older key the receiver might possess belonging to the transmitter's chain (this could even be the epoch key itself, k_E). Once verified, the receiver uses this key to decrypt the data D_i (step 5).
5) **Specifying epoch-keys**: We assume the network initializes at $j = 0$s where j is a clock keeping track of network time. The root-key serves as the very first epoch-key. We designate a value, n, such that every $n - th$ key to follow in the chain is to be the

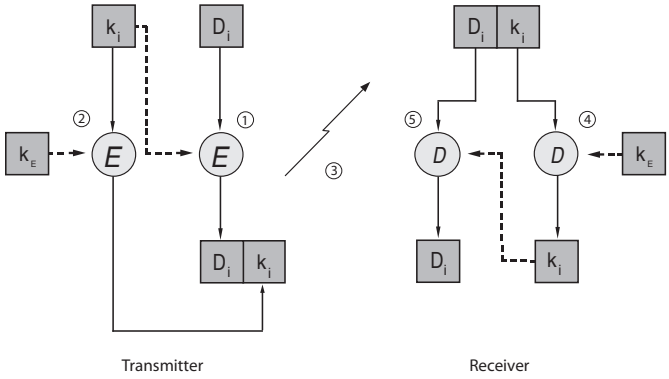

Figure 5: Encryption and Decryption Operations

epoch-key for a later interval. As an example, we choose $n = 100$. This means that, after the root-key k_0, the second epoch-key will be k_{100} which will be disseminated by the transmitter at $j = 100$s. Assignation of epoch-keys is described in Fig.6: the root-key, k_0, is used to encrypt all data for the first epoch, which spans k_0 to k_{200} (i.e. 201 seconds). Network receivers already possess the transmitter's root-key (from the Bootstrapping Trust Phase) and can decrypt this data. We assume that the transmitter encounters every other node in the network at least once in the interval , $100 < j < 200$. Every node will have then received a key, k_i such that $100 \leq i \leq 200$. The receiver can hash this key back $(i - 100)$ number of times to yield k_{100}, i.e. the next epoch-key. This new epoch-key comes into effect when $j = 201$. The first epoch lasts for 201s (which can be expressed as $2n + 1$) out of necessity. Following epochs last 100s each. Again, we can assume there is high probability (quantified in the next section) that all receivers receive a packet from the transmitter at least once in the interval $200 < j < 300$. This will enable them to all possess a key that can be hashed to yield the third epoch-key k_{200} that comes into effect at $j = 301$s. Epoch-key updates follow this basic pattern.

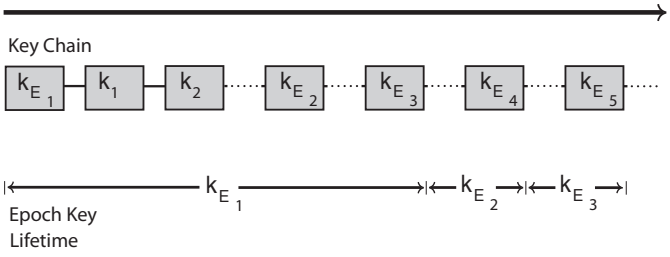

Figure 6: Assignment of Epoch Keys

However, if a node were to receive no packets from the transmitter within an epoch, it would be unable to decrypt future communication. This only applies to *direct* communication. The node may still receive data from the transmitter via multi-hop from other nodes.

6) **Key Management**: Receivers have to store and update two keys for every node in the network, every node's epoch-key, k_E and the last received key together with time-stamps or count value. The epoch-key allows the receiver to decrypt communication within the epoch. The last received key minimizes the verification process; the receiver does not have to hash k_i all the way back to the transmitter's epoch key to verify it, it could simply hash back to the more recent key, e.g. if the receiver were in extended contact with the transmitter, it would take only one hash operation to verify each message received during that time.

3.2. Discussion

Our use of the key-chain is somewhat similar to μTesla in that both schemes depend on the one-way nature of the key-chain to provide source authentication. Time-synchronizing the network also ensures key validity is bounded by time: this protects against replay attacks. If an older packet is replayed, the key for that has already expired, the verification process will fail and the receiver discards the packet. Using different encryption keys creates high message entropy and provides semantic security. And the per-hop encryption allows the benefit of data fusion.

The epoch-key itself varies over time to add an extra layer of security to the network. If an attacker were to try to hack the epoch-key, he would be restricted to doing so within the window of the epoch itself, and in the next section we describe a method for reducing epoch length to an optimal value. And, even if he somehow procured the key, he could decipher messages but, would not be able to generate future keys and masquerade as the transmitter.

We note that maintaining long key-chains and key-tables for other nodes can quickly overwhelm the memory resources of the typical sensor device. For this reason, we suggest that key-chains be kept short, and, if possible, be coded into ROM rather than flash memory. In the case of the soccer match, it would be feasible to use remote programming software to re-key the network during breaks.

4. Analysis

In this section, we fine-tune the performance of our scheme using link-connectivity statistics.

We define a receiver's **attachment probability**, *P*, to be the probability that the receiver can successfully decrypt future messages from the transmitter. This success depends on the receiver procuring at least one key per epoch. A receiver that typically has frequent communication with the transmitter will consequently have a high attachment probability.

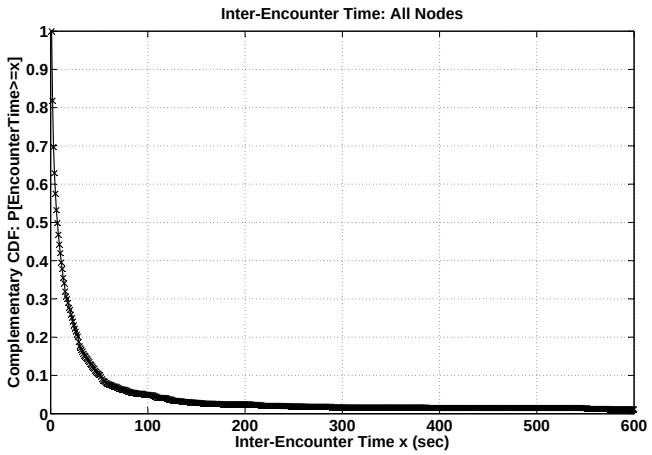

Figure 7: Inter-Encounter Times over All Pairs of Nodes

Another strategy to increase attachment probability, *P*, is to extend epoch lengths and maximize the chances of the receiver and transmitter encountering each other. However, nodes that meet rarely within an epoch will have to perform a larger number of verification operations on the received message. We denote this metric as **computation cost**. Smaller epochs will entail lesser computation cost in packet verification, but the attachment probability will suffer.

Inter-encounter time can be used to compute appropriate values of epoch length that represent a balance between verification operations and attachment probability. Inter-encounter time between nodes is a measure of the time between encounters, or the maximal period of disconnectivity. Revisiting our soccer game experiment, we plot the Complementary Cumulative Distribution Function (CCDF) for inter-encounter time over all pairs of nodes in Fig.7. The result indicates that inter-encounter time values are surprisingly low across the network. Almost 60% of encounters are spaced 10s apart or so. The probability of inter-encounter time being more than 100s is approximately 0.05. This distribution clearly favors smaller epoch sizes.

Since the encryption key changes once every second, the maximum computation a node would have to perform to verify a message would be if it were to receive only the last message within an epoch. In this case it would hash back an amount equivalent in seconds to the epoch length. This one-to-one correspondence allows us to plot inter-encounter time against epoch-length/maximum-computation-cost. We invert the CCDF to yield the probability that nodes do *not* encounter each other and plot it against epoch-length/maximum-computation-cost in Fig.8 in log scale.

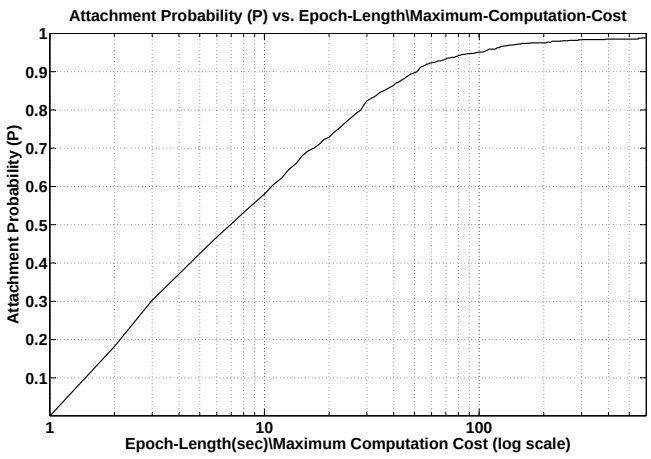

Figure 8: Probability of Attachment vs. Epoch-Length/Maximum-Computation-Cost over All Pairs of Nodes (log scale)

The results bear out intuition: the greater the epoch length, the higher the probability of attachment and the greater the number of worst-case verification operations. However, small epoch sizes yield a very high attachment probability: an epoch length of 20s gives over a 70% chance of successful attachment. An epoch of 30s takes the probability to over 80%. We recall from earlier that smaller epochs provide greater security but that could lead to nodes being isolated from others permanently. Using link-connectivity statistics, a network operator can tune epoch length as per requirements to balance security and computation against probability of disconnectivity.

5. Conclusion and Future Work

In this paper, we identified fundamental security challenges in highly dynamic wireless sensor networks and highlighted network characteristics with results from a real sensor network deployment, i.e. athlete monitoring during a soccer match. We proposed a per-hop security solution utilizing time-bound symmetric-key cryptographic mechanisms to provide confidentiality, source authenticity, replay protection, semantic security and data fusion capability. We use link connectivity statistics to identify the optimal epoch length that minimizes verification operations against probability of attachment.

Our future work will focus on researching security mechanisms that allow greater scalability, extended network lifetime and dynamic node joining.

References

[1] T. Spyropoulos, K. Psounis, and C. S. Raghvendra, "Performance Analysis of Mobility-Assisted Routing," in *Proceedings of the 7th ACM international Symposium on Mobile Ad hoc Networking and Computing*. Florence: ACM, 2006.

[2] K. Lorincz, D. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, and M. Welsh, "Sensor networks for emergency response: Challenges and opportunities," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 16–23, October 2004.

[3] K. Lorincz and M. Welsh, "Motetrack: A robust, decentralized approach to rf-based location tracking," in *International Workshop on Location-and-Context-Awareness, LoCA*. Munich: DLR, May 2005.

[4] S. Armstrong, "Wireless connectivity for health and sports monitoring: a review," *British Journal of Sports Medicine*, vol. 41, pp. 285–289, January 2007.

[5] T. Kuhn, T. Jaitner, and R. Gotzhein, *The Engineering of Sport 7*. Springer, 2008, ch. Online-Monitoring of Multiple Track Cyclists During Training and Competition.

[6] R. M. Smith and C. Loschner, "Biomechanics feedback for rowing," *Journal of Sports Sciences*, vol. 20, pp. 783–791, February 2002.

[7] T. Small and Z. J. Haas, "Resource and Performance Tradeoffs in Delay-tolerant Wireless Networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. Philadelphia: ACM, 2005.

[8] T. Spyropoulos, K. Psounis, and C. S. Raghvendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Single-Copy Case," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 63–76, February 2008.

[9] ——, "Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility," in *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops(PerComW'07)*. New York: IEEE, 2007.

[10] ——, "Spray and Wait: an Efficient Routing Scheme for Intermittently Connected Mobile Networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. Philadelphia: ACM, 2005.

[11] "Developing Protocols for Soccer Player Monitoring, http://www.ee.unsw.edu.au/ vijay/athlete/index.html," 2009.

[12] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," in *SenSys '04*. Baltimore, Maryland: ACM, Nov. 2004.

[13] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Wireless Sensor Networks," in *Mobile Computing and Networking*. Rome, Italy: ACM, 2001.

[14] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A Secure Sensor Network Communication Architecture," in *International Conference on Information Processing in Sensor Networks*. Cambridge, Massachusetts: ACM/IEEE, April 2007.