Secure Aggregation for Wireless Networks

Lingxuan Hu David Evans
Department of Computer Science
University of Virginia
Charlottesville, VA
[lingxuan, evans]@cs.virginia.edu

Abstract

An emerging class of important applications uses ad hoc wireless networks of low-power sensor devices to monitor and send information about a possibly hostile environment to a powerful base station connected to a wired network. To conserve power, intermediate network nodes should aggregate results from individual sensors. However, this opens the risk that a single compromised sensor device can render the network useless, or worse, mislead the operator into trusting a false reading. We present a protocol that provides a secure aggregation mechanism for wireless networks that is resilient to both intruder devices and single device key compromises. Our protocol is designed to work within the computation, memory and power consumption limits of inexpensive sensor devices, but takes advantage of the properties of wireless networking, as well as the power asymmetry between the devices and the base station.

1. Introduction

Wireless sensor networks are emerging technologies that have a wide range of potential applications such as battlefield surveillance and emergency response [5, 13]. Research on sensor networks generally assumes a trusted environment, but in many likely sensor network applications, the network will be deployed in situations where an adversary may be motivated to disrupt the function of the network. An adversary may be able to position several intruder nodes within the network and use them to transmit false messages. Further, an adversary may compromise a node in the network and gain access to its key material. In this paper, we focus on an adversary who wants to corrupt the information being produced by the sensor network. We regard confidentiality of the messages themselves to be unnecessary and focus only on the integrity of the results transmitted to the base station. Although we acknowledge that wireless networks are vulnerable to a wide array of denial of service attacks, we consider the most serious threat to be an adversary who can compromise the network to provide false readings without detection by the operator. Such an adversary could make a network designed to monitor an area for physical intrusions fail to report an intrusion without the operator realizing anything is wrong.

The primary security challenges for wireless sensor networks lie in addressing the conflict between limited resources (energy, computational overhead, memory, etc.) and security requirements. Traditional security techniques, such as public-key cryptography, require expensive computations and long messages that would quickly deplete the battery of typical sensor devices. We focus on lightweight security mechanisms, and design our protocols to offload as much of the processing as possible to the base station.

In the next section we provide background on related work. In Section 3, we describe our assumptions and cryptographic primitives and an overview of our protocol for secure aggregation. The details of protocol are discussed in Section 4. Section 5 analyzes possible attacks. Section 6 evaluates the costs of our protocol. We propose a more scalable variation on our protocol in Section 7.

2. Background

Several researchers have dealt with various security issues in low-energy ad hoc wireless networks. Buttyán and Hubaux provide a summary report on recent work [3]. Wireless networks pose interesting key management challenges because of the lack of a fixed infrastructure for supporting a traditional certificate authority. Zhou and Haas [24] propose using threshold cryptography to allow a group of ad hoc devices to manage keys. Kong et al. [12] propose asymmetric mechanisms using localized threshold signatures for certificates. In our work, we assume a fixed base station that can establish secrets with the ad hoc wireless nodes before deployment, so we do not address these issues further.

Asymmetric algorithms are likely to require more computation and power then is feasible for many sensor network applications. Basagni et al. [2] argue that clusters of nodes would share a symmetric key, but this design is exposed to single-node-compromise attack. Balfanz et al. [1] present a solution to complete an authenticated key exchange protocol over the wireless link without a public key infrastructure, but it requires a privileged side channel for pre-authentication.

Many routing protocols have been proposed for ad-hoc networks [6] including DSR [11] and AODV [18]. Secure routing in the wireless networks has received increased attention recently. Marti et al. [15] complement DSR with a *watchdog* for detection of denied packet forwarding and a *pathrater* for routing policy rating, which enable nodes to avoid malicious nodes in their routes but require extensive listening. Papadimitratos and Haas [17] showed how impersonation and replay attacks can be prevented for on-demand routing by disabling route caching and providing end to end authentication using an HMAC primitive [16]. Dahill et al. [4] focus on providing hop-by-hop authentication for the route discovery stage of AODV and DSR relying on digital signatures. Hu et al. [8] absorb the idea of SPINS to produce a hardened version of DSR suitable for sensor networks.

SPINS [19] is a suite of security protocols optimized for resource constrained environments, and it achieves asymmetry from clock synchronization and delayed key disclosure. Our work is most directly inspired by this SPINS work, and we use their μTESLA protocol (described in the next section). Yuval [23] presents an algorithm for authenticating messages with short MACs. Symmetric encryption algorithms designed for low-power environment have also been proposed, including TEA [21] and RC5 [20]. Our protocol requires efficient and secure MAC algorithms, but we are not dependant on any particular encryption algorithm.

To reduce the power consumed forwarding messages, researchers have identified the importance of data aggregation [5, 10]. Aggregation collects results from several sensors and calculates a smaller message that summarizes the important information from a group of sensors. For example, suppose the operator is interested in the average sensor reading for some value in the network. An inefficient way to find this would be for every sensor node to send its reading to the base station (possibly over multiple forwarding hops), and for the base station to calculate the average of all readings received. A more efficient way to collect the same information would be for intermediate nodes to forward the calculated average value of the readings they receive along with a count of the number of readings it incorporates. Each node then calculates the average for all of its descendents and only need send that value and the number of descendants to its parent. Several recent research efforts have explored different aggregation protocols for sensor networks assuming a trusted environment including directed diffusion [9], LEACH [7], greedy aggregation [10], and Cougar [22]. TAG is an in-network aggregation service for TinyOS motes that supports a SQL-like language for expressing aggregation queries over streaming sensor data [14]. For our work, we assume a simple hierarchical tree aggregation, but our protocol could readily be adapted to any hierarchical aggregation protocol.

3. Secure Aggregation

Message aggregation can reduce communication overhead significantly, but message aggregation makes security more difficult. Each intermediate node can modify, forge or discard messages, or simply transmit false aggregation values, so one compromised node is able to significantly alter the final aggregation value. Further, aggregation interferes with message encryption. We cannot encrypt messages using a unique key shared between each device and the base station since each intermediate node needs to understand the received messages to perform aggregation. We cannot risk storing the same key on every device to enable encryption or authentication, since an adversary who recovers the key from a single device would then be able to control the entire network.

Our design is aiming at providing lightweight security mechanisms to effectively detect node misbehavior (dropping, modifying or forging messages, transmitting false aggregate value). We enable a base station to trust results from a sensor network, even if an adversary may be able to deploy intruder nodes inside the network and recover the key material from a single node. Our design exploits two main ideas: delayed aggregation and delayed authentication. Instead of aggregating messages at the immediate next hop, messages are forwarded unchanged over the first hop and then aggregated at the second hop. This increases the transmission costs, but will enable integrity guarantees for networks where two consecutive nodes are not compromised. Instead of attempting to authenticate messages right away, we save resources by authenticating messages after a time delay. This enables authentication keys to be symmetric keys, revealed to the authenticator after the time delay has expired. We design our protocol around these assumptions:

1. The base station is powerful and can broadcast messages to all nodes directly. Sensor devices are low power and can only communicate with nearby nodes.
2. There are low-level network mechanisms in place to provide reliable message delivery.
3. The network is spread out enough so there are likely to be many hops between a typical node and the base

station. The network is dense enough so that there are usually several nodes within one-hop distance of any particular node.

4. Before deployment, each node can establish shared secrets with the base station.

3.1 Encryption Primitives

Our protocol will depend on the base station broadcasting a series of authenticated messages to the sensor nodes. Due to the limited power and memory of sensor node, it is impractical to use an asymmetric algorithm for message authentication. We adopt the μTESLA protocol for authentication of messages transmitted by the base station [19]. μTESLA is a protocol that provides authenticated broadcast for resource-constrained environments, it achieves asymmetry from clock synchronization and delayed key disclosure.

The base station generates a one-way key chain using a public one-way function F where $K_i = F(K_{i+1})$. Each device stores K_0 before deployment where $K_0 = F^n(K)$ (that is, n applications of F to a secret K). We can imagine doing this using a location limited channel where the keys are established in a secure environment near the base station before they are deployed [1].

The first base station transmissions will be encrypted using key $K_1 = F^{n-1}(K)$. After all messages transmitted using K_1 have been received, the base station reveals key K_1. Sensor nodes can compute $F(K_1) = F(F^{n-1}(K))$ and verify that it matches $K_0 = F^n(K)$. After this, the sensor nodes can decrypt the messages previously transmitted encrypted with K_0. Successive keys can be revealed in a similar manner, until $K_n = K$ is reached. If necessary, a new hash sequence could be started before this point by having the base station send a new $K'_0 = F^n(K')$ using K_n from the original hash chain.

Our protocol will depend on sensor nodes being able to produce messages that can be authenticated by the base station (we are not concerned with confidentiality of these messages). For this, we assume each node is initialized before deployment with a symmetric secret key, K_{AS}, shared with the base station. A temporary encryption key will be computed by encrypting a counter value using this key. For example, $K_{A0} = E(K_{AS}, 0)$. The base station knows K_{AS} and can synchronize counter values with the sensor devices. After each stage, the temporary encryption key will be revealed to enable other sensor devices to authenticate messages transmitted by nearby sensors.

3.2 Protocol Overview

Figure 1 depicts part of an example sensor network comprising eight power limited sensor nodes (*A–H*) and a powerful base station. The base station collects information from all sensor nodes and transmits it to the operator over a wired network. The sensor nodes are all identical, but four are acting as leaves (*A–D*), and the remaining nodes are acting as intermediate nodes. A typical network would have hundreds or thousands of nodes and a branching factor greater than two.

We assume a secure self-organizing protocol is used to form a routing hierarchy where each node has an immediate parent. Each leaf node transmits its reading to its parent. The raw messages include the node data reading and a node id that uniquely identifies the node. A message authentication code, MAC (K_{Ai}, R_A), is included with the message. It uses a key that is known to the node and base station, but not yet known to the other sensor nodes. The parent node will store the message and its MAC until the key K_{Ai} is revealed by the base station. Then, it will verify the MAC and raise an alarm if it does not match.

Message aggregation is performed in each intermediate step. Nodes wait for a specified time to receive messages from their children, and then retransmit the messages and MACs they receive directly from immediate children. Each child can contribute at most one reading in each time step. Nodes aggregate the data they receive from their grandchildren (via their children) and transmit the MAC of the aggregation value. Delayed aggregation ensures that an adversary who obtains key material from a compromised node cannot tamper with many sensor readings.

After a stage of messages arrives at the base station, the base station reveals the temporary node keys along with a MAC generated using base station's current μTESLA key. Once the key is revealed, nodes advance to the next temporary node key. After this, the μTESLA key is revealed to enable authentication, and the base station advances to the next key in the chain.

4. Protocol Details

Our protocol involves separate stages for sending data towards the base station and authenticating that information retroactively. We describe the protocol using the small sample network in Figure 1. Keep in mind, though, that the benefits of our protocol come when there are a large number of nodes arranged in a deep tree, so that many readings can be aggregated in a single message. For simplicity in this presentation, we assume only leaf nodes are collecting sensor readings, and intermediate nodes are just aggregating and forwarding data. Extending our protocol to support sensor readings at intermediate nodes is straightforward. We will use the following notation to describe our protocol:

A, B, C, …	Sensor nodes
S	Base station
$A \rightarrow B$	Node *A* sends a message to *B*. Since messages are wireless, this is a local

	broadcast and nearby nodes will also hear the message.
ID_A	Unique ID of node *A*.
$M_1 \mid M_2$	Concatenation of messages M_1 and M_2.
$E(K, M)$	Encryption of *M* using key *K*.
MAC (K, M)	Authentication code of *M* using key *K*.
Aggr (x, y)	Result of the aggregation function on *x* and *y*. The aggregation function must be deterministic, distributive and not depend on the order readings are incorporated.
K_{AS}	Unique key shared between node *A* and base station *S*.
R_A	Data reading value of node *A*.
K_{Ai}	The i^{th} key for node $A = E(K_{AS}, i)$
K_i	The i^{th} key in the base station μTESLA hash chain = $F^{n-i}(K)$.

Data Transmission. Assume nodes *A*, *B*, *C* and *D* are sending messages to base station through the tree shown in Figure 1.

1. Leaf nodes send messages to their parent. These messages include MACs calculated using their respective temporary node keys. For example:

   $A \rightarrow E \qquad R_A \mid ID_A \mid \text{MAC}(K_{Ai}, R_A)$

   The message does not need to include a nonce to prevent replay attacks since each temporary MAC key is used only once.
2. Intermediate nodes receive messages from their children. The parent node cannot yet verify the MAC since it uses the child's temporary key, which will be revealed to the parent later in the verification phase. For now, the parent just stores the message and MAC. The intermediate node waits until a time has elapsed, and then sends a message to its parent that retransmits the sensor readings and MACs, as well as the computed MAC of the calculated aggregate value:

   $E \rightarrow G \qquad R_A \mid ID_A \mid \text{MAC}(K_{Ai}, R_A) \mid R_B \mid ID_B \mid \text{MAC}(K_{Bi}, R_B) \mid \text{MAC}(K_{Ei}, Aggr(R_A, R_B))$

   There is no need to transmit the computed aggregate value, since both R_A and R_B are transmitted to *G*, who independently computes $Aggr(R_A, R_B)$. It is also not necessary to transmit the ID_E to *G* since *G* knows enough about the network topology to determine the sending node from the grandchild IDs.
3. Node *G* receives messages from *E* and *F*. It calculates the aggregate values of its grandchildren's readings through each child. That is, the aggregate value of R_A, R_B from *E* and aggregate value of R_C, R_D from *F*. It then transmits the aggregate values of its grandchildren and its children's ID and MAC values. It also computes and transmits the MAC of the next aggregate value, $Aggr(R_A, R_B, R_C, R_D) = Aggr(Aggr(R_A, R_B), Aggr(R_C, R_D))$. Since the aggregation function is deterministic and known to all nodes, the MAC calculated by *E* will authenticate the value computed by *G*. The sensor readings and MAC values received from *E* and *F* are stored for later

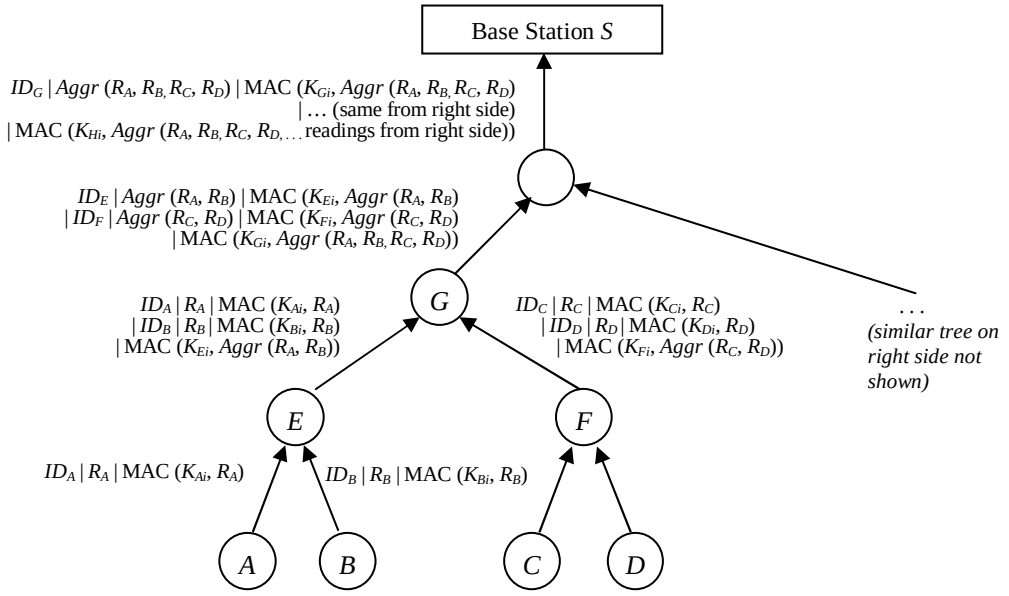

Figure 1. Example Sensor Network.

authentication.

$G \rightarrow H$

$ID_E \mid Aggr\ (R_A, R_B) \mid \text{MAC}(K_{Ei}, Aggr\ (R_A, R_B))$
$\mid ID_F \mid Aggr\ (R_C, R_D) \mid \text{MAC}(K_{Fi}, Aggr\ (R_C, R_D))$
$\mid \text{MAC}(K_{Gi}, Aggr\ (R_{A,}, R_B, R_C, R_D))$

4. Similarly, node *H* receives messages from *G* and another branch, and transmits the aggregate message to base station. Note that the message length would not increase if the network was deeper.
5. The base station receives the message from *H*. It can calculate the final aggregate value, $Aggr\ (R_A, R_B, R_C, R_D, \ldots)$ from $Aggr\ (R_A, R_B, R_C, R_D)$ and the aggregate values reported by its other children.

Data Validation. Our protocol is designed to ensure that a single compromised node can only mislead the network about its own reading. It should not be able to make false claims about other node's readings, or produce aggregate values that improperly represent the state of the network.

Since the base station has a shared temporary key with each sensor node, it can verify that the message it receives in the final step was transmitted by *H* by calculating the MAC of the aggregation it calculated using K_{Hi} and comparing it to the MAC transmitted in the message. This validates that *H* sent the final message, but does not validate that it correctly reflected reading from the other sensor nodes. The base station also receives the MACs and readings of its grandchildren, and can authenticate those values. Our goal, though, is to authenticate all the readings that contributed to the aggregation value, without requiring every reading to be sent to the base station.

To validate data, the base station reveals temporary node keys to the network. Using wide-area broadcast, the base station sends out each temporary node key along with a MAC using its current μTESLA key, K_i. Nodes will advance to the next temporary node key for succeeding messages. Requiring that the base station reveal every node key for every aggregate reading it receives does not scale to large sensor networks or work well in situations where frequent readings are desired. In Section 7, we consider a protocol variation that addresses this issue.

Note that although all the node keys are sent out by the base station, each sensor node only needs a few of them. For example, node *E* needs K_{Ai} to verify MAC (K_{Ai}, R_A). If the key broadcasts are synchronized, it will not be necessary to listen to all the key broadcasts to find the relevant ones.

After sending out all the node keys, the base station sends out its current μTESLA key, K_i, to enable nodes to check transmitted MAC values, and advances to the next key in the chain for future messages. After receiving K_i, nodes verify the MAC for the node keys. Nodes verify K_i is legitimate by calculating $F\ (K_i)$ and comparing it to K_{i-1}.

If a node detects a forged message in the data validation stage, it sends out an alarm message. Alarms are raised by a parent when it detects an inconsistent MAC from a child or grandchild, and sent to the base station along with a MAC computed using the node's temporary key.

5. Attack Analysis

Our protocol defends against attacks involving many intruder nodes that do not obtain any key material, and limits the effectiveness of attacks in which all the key material stored on a particular node is compromised.

Intruder Node Attacks. Preventing intruder nodes with no access to keys from injecting bogus data into the aggregation is straightforward. Delayed aggregation is not necessary to accomplish this, it is sufficient to require all transmitted data to include a MAC generated with a key that will be revealed later. An intruder node may attempt to transmit readings with an ID of a legitimate child node (which the intruder can easily obtain since it is transmitted unencrypted) but will be unable to compute a valid MAC for the bogus data. Both the parent node that receives and forwards this reading and the grandparent node that receives and computes an aggregate using this reading will detect the bogus transmission when the temporary sensor node keys are revealed. These nodes will generate alarm messages that are received by the base station. Note that replay attacks are ineffective since the sensor node keys change with every reading. A replayed message will appear to have an invalid MAC, since the previous MAC was calculated using a different key.

The intruder node will succeed in preventing the base station from acquiring correct readings during that time stage. In many applications, alerting the base station to the presence of an intruder is sufficient – if the goal of the sensor network is to detect the presence of intruders it has done its job. In other applications, the network must continue to provide valid readings. The simplest solution would be for the parent that detected the intruder node to recognize that an intruder node is within transmission distance, and stop forwarding any messages it receives. The application would continue to function, just without incorporating any readings from the vulnerable area.

Compromised Node Attacks. An adversary who obtains key material from a node is more dangerous. Since the sensor node devices are inexpensive, it is likely that a competent adversary with physical access to a device would be able to obtain the key material stored on the device. Since the μTESLA protocol means no secure information about the base station keys need be stored on

the sensor devices, an adversary who obtains key material from a node does not have the ability to forge base station messages. The adversary does, however, have the ability to forge node messages.

If the compromised node is a leaf node, the adversary can transmit false readings without detection. There is no cryptographic way to prevent this, since we are assuming the adversary has obtained all key material on the compromised node. There are, however, ways to limit the effectiveness of this attack. For some aggregation functions (for example, calculating an average temperature reading over thousands of nodes where all readings are checked to be within a reasonable range) a single bogus value may not significantly perturb the final result. In other cases, parent nodes receiving values from several sensor nodes may ignore a single outlying value. Even in cases where there is no compromised node, it is likely that some sensors will fail and produce bogus values so it is important that applications handle this case.

A more serious concern is the case where a node higher up the routing tree is compromised. For a straightforward protocol with child→parent message authentication but without delayed aggregation, a node can generate a bogus message that represents readings for many nodes. Assuming the base station can determine the network topology, the compromised node effectively can misrepresent all of its descendant's readings. An adversary who compromises a node near the top of the routing hierarchy could effectively misrepresent the readings of a large portion of the sensor network.

Delayed aggregation limits the damage an adversary who compromises a single node can do. The immediate parent of that node expects to receive the MACs of the retransmitted messages from all the compromised node's children, and will raise an alarm if the transmitted aggregation values are inconsistent with those MACs. In this case, it is unclear to the parent node detecting the attack whether it is the child or the grandchild that has been compromised. If our goal is only to detect the intrusion this is sufficient. At worst, two nodes with need to be excluded from the sensor network.

Another attack strategy would be to selectively drop readings to prevent readings that reveal the intrusion from reaching the base station. A compromised node near the base station could drop aggregation results from selected children. If the node is within two hops of the base station, it would be apparent to the base station that no readings were being received from that child. This would raise some concern, although it is indistinguishable from normal node failure (except in that case, the routing tree would eventually reorganize to use a different node). If the node is further away from the base station, but still deep enough that selective dropping can accomplish the attacker's goals, it would not be apparent to the base station that readings are missing. A possible defense involves nodes snooping on their parents. If a child node transmits a message to a parent, but its readings are not included in the parent's transmission, the child node can raise an alarm. Since its parent is the compromised node, this alarm must be routed through the network in a way that does not depend on the parent forwarding it.

Our protocol does not thwart an attacker who simultaneously compromises a parent and child node. For example, an attacker who compromised both nodes *E* and *G* in our sample network would be able to transmit a bogus aggregation value for R_A and R_B, along with the legitimate reading for *F*'s subtree. In general, an attacker who simultaneously compromises a child and parent can misrepresent the readings for every node in the child's subtree. We view success of such an attack as unlikely, since the attacker would need to compromise the parent node's key material before the routing hierarchy reformed. If the parent is disabled for too long, its children will reorganize to avoid the disabled node. In situations where this attack is a serious concern, it could be thwarted by adding another delay stage to the aggregation. That is, instead of retransmitting values and MACs from your children and aggregating values from your grandchildren, nodes would retransmit values and MACs from their children and grandchildren, and only aggregate values from their great-grandchildren. This would, however, substantially increase the communication costs required. Perhaps a more realistic defense would depend on nodes snooping on transmissions from other nearby nodes.

6. Cost Analysis

We analyze the costs of our protocol. We focus on the communication costs as the energy required to transmit and listen for messages tends to overwhelm the computational and memory requirements. Our protocol is designed to provide a balance between security and communication costs. We provide more security than is possible with full in-network aggregation, but require substantially less communication than is necessary for networks with no aggregation.

Consider an ideal routing tree where the leaves are *d* hops away from the base station and each node has *b* children. For the first step each of b^d leaf nodes sends it reading to its parent. We use *m* to represent the length of the data reading and *c* for the length of the node ID and MAC. In the next step, each parent retransmits the readings and MAC values, as well as the MAC of the aggregate value. The aggregation function output is the same length as the original sensor reading. For some aggregation functions, such as the average example, it may be a constant factor longer, but for aggregation to be useful,

it should not scale with the number of readings. So, a parent with b children will need to transmit $b(m+c) + c$ bits. There are b^{d-1} second level nodes, so the total number of bits transmitted in the second step is $b^d(m+c) + b^{d-1}c$. At the next level, and every successive level, each node transmits the MAC values from each child and the new aggregate value and MAC: $b(m+c)+c$ bits. The transmission size does not increase at successive levels in the tree because we assume the aggregation function is always the same length as the original sensor reading. Hence, the total number of bits communicated using our protocol is $m(2b^{d+1}–b^2–b)/(b–1) + c(2b^{d+1}+b^d–b^2–2b)/(b–1)$.

For comparison, without aggregation each node transmits messages from all its descendants. This requires transmission of $(m+c)db^d$ total bits. With insecure aggregation method, b messages are aggregated into 1 message at each intermediate node, so each node need only transmit m bits for $m(b^{d+1}–b)/(b–1)$ total bits transmitted.

To give a sense of what these numbers mean for typical applications, we select m = 22 bytes and c = 8 bytes based on the assumptions in [19] (for messages where no MAC is included, 2 bytes are required for a message integrity CRC). Given a network with b = 4 and d = 5, there are a 1024 leaf nodes. The total communication in a time segment where each leaf node transmits a reading is 150KB with no aggregation, 82.5KB with our protocol (a 45% reduction), and 32KB with full insecure aggregation. As the depth of the tree increases, the relative advantages of our protocol over no aggregation increase. With d = 8, our protocol requires 2.75 times the amount of communication of the insecure aggregation protocol, but only 34% as much communication as is necessary without aggregation. The total communication of our secure aggregation protocol is significantly less than non-aggregation method, and is roughly a little more than two times of communication that required for insecure aggregation.

The base station also needs periodically broadcast the temporary keys of all sensor nodes. This requires substantial communication, but since the base station is connected to the power grid it does not pose a serious problem. The time sensor nodes must spend listening is a more serious concern. By using synchronized communication time windows, these costs can be made acceptable.

The computational and memory costs are likely to be insignificant compared to communication. Each intermediate node needs to compute aggregate values for its grandchildren's readings, requiring b aggregation function applications. Each intermediate node also needs to authenticate both children and grandchildren nodes after keys are disclosed by base station, so b^2+b MAC computations are needed. The memory requirements are unlikely to be an important constraint. All nodes need to store messages received from their children until keys are disclosed by base station. These are data values and associated MAC values from its grandchildren and children. For the ideal routing tree, the leaf node does not need to store anything, each first level node needs to store $(m+c)b$ bits, and every higher level intermediate node needs to store $(m + c)b^2+cb$ bits. Using the values of m = 30 bytes, c = 8 bytes, b = 4, a first level node needs to store 152 bytes and higher level nodes store 640 bytes.

7. Scalable Variation

Our original protocol requires the base station to transmit all the temporary sensor node keys after each stage. This is reasonable for small sensor networks, but not for sensor networks with thousands of nodes. Further, data received by the base station is not validated until after the sensor node keys have been transmitted and the base stations has waited long enough for nodes to verify the MACs and transmit alarm messages to the base station. In this section, we describe a variant on our protocol that addresses both of these issues, but increases the amount of communication required by sensor nodes.

Instead of relying on the base station to broadcast temporary node keys, we can use μTESLA key chains to reveal and authenticate keys locally. Before deployment, each node will establish a shared secret key with the base station K_{AS} as in the original protocol. In this case, though, we will determine successive temporary node keys by calculating $K_{Ai} = F^{n-i}(K_{AS})$.

After deployment, each node needs to establish the μTESLA key chain with its parent by securely revealing K_{A0}. Since we do not know where the nodes will be located before deployment (which, for example, may involve dropping the nodes from an airplane) and we cannot store all the keys in the sensor nodes, this requires a transmission for the base station authenticated using its μTESLA key.

Once the key chains have been established, we can transmit messages using the μTESLA keys. Node A sends a message to its parent node E containing its reading and revealing a μTESLA key:

$$A \rightarrow E \qquad ID_A \mid R_A \mid K_{A1} \mid \text{MAC}\,(K_{A2}, R_A)$$

Node E can authenticate K_{A1} immediately by checking $F(K_{A1})$ matches K_{A0}. The next message will reveal K_{A2}, which E can use to validate the MAC.

If an intruder node attempts to forge a message from A, it will be detected immediately the intruder does not know K_{A1}, and the transmitted value will fail the $F(K_{A1}) = K_{A0}$ check. A more sophisticated intruder node may attempt to transmit a message using the overheard K_{A1} value, but with a different R_A value. Without knowing K_{A2} however, the adversary will be unable to produce a correct MAC value and the forged message will be detected in the next stage.

This depends on synchronization between the nodes to know when the μTESLA keys are revealed. Since the base station can broadcast stage update messages, however, this is feasible.

8. Conclusion

Heretofore, designers of sensor network applications have been faced with a choice between either authenticating sensor messages or aggregating readings in the network to save power. Our protocol offers a new tradeoff – resistance to single node compromises with in-network data aggregation. Since many sensor network applications will operate in hostile environments, providing a way to increase confidence in the integrity in the sensor readings without giving up the opportunity to aggregate intermediate results in the network is a valuable design option.

Acknowledgements

This work was funded in part by grants from the National Science Foundation (CCR-0092945 and EIA-0205327) and NASA Langley Research Center.

References

[1] Dirk Balfanz, D. K. Smetters, Paul Stewart and H. Chi Wong. *Talking To Strangers: Authentication in Ad-Hoc Wireless Networks*. In Proceedings of the ISOC 2002 Network and Distributed Systems Security Symposium, February, 2002.

[2] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. *Secure pebblenets*. In Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2001.

[3] Levente Buttyán and Jean-Pierre Hubaux. *Report on a Working Session on Security in Wireless Ad Hoc Networks.* Mobile Computing and Communications Review, Volume 6, Number 4. November 2002.

[4] B. Dahill, B. Levine, C. Shields, and E. Royer. *A secure routing protocol for ad hoc networks.* Technical Report, Department of Computer Science, University of Massachusetts, August 2001.

[5] Deborah Estrin and Ramesh Govindan and John S. Heidemann and Satish Kumar. *Next Century Challenges: Scalable Coordination in Sensor Networks*. Mobile Computing and Networking. 1999.

[6] L. Feeney. *A taxonomy for routing protocols in mobile ad hoc networks*. Technical Report T99/07, Swedish Institute of Computer Science, October 1999.

[7] Wendi Rabiner Heinzelman, Anantha Chandrakasan, Hari Balakrishnan. *Energy-efficient communication protocol for wireless microsensor networks.* In Proceedings of the Hawaii International Conference on System Sciences, Maui, Hawaii, January 2000.

[8] Yih-Chun Hu, Adrian Perrig and David B. Johnson. *Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks.* Mobicom 2002.

[9] Chalermek Intanagonwiwat, Ramesh Govindan and Deborah Estrin. *Directed diffusion: A scalable and robust communication paradigm for sensor networks.* Mobile Computing and Networking, August 2000.

[10] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. *Impact of network density on data aggregation in wireless sensor networks*. In Proceedings of International Conference on Distributed Computing Systems, November 2001.

[11] D. Johnson and D. A. Maltz. *Dynamic source routing in ad hoc wireless networks*. Mobile Computing, 1994.

[12] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang. *Providing Robust and Ubiquitous Security Support for Mobile Ad-hoc Networks*. IEEE 9th International Conference on Network Protocols, 2001.

[13] Large Scale Networking Coordinating Group of the Interagency Working Group for Information Technology Research and Development, *Report from the Workshop on New Visions for Large-Scale Networks: Research and Applications*. March 2001. http://www.itrd.gov/iwg/pca/lsn/lsn-workshop-12mar01/

[14] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. *TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks.* OSDI, December 2002.

[15] S. Marti, T. Giuli, K. Lai, and M. Baker. *Mitigating routing misbehavior in mobile ad hoc networks*. Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking, August 2000.

[16] National Institute for Standards and Technology (NIST). *The Keyed-Hash Message Authentication Code (HMAC).* No. FIPS 198, 2001.

[17] Panagiotis Papadimitratos and Zygmunt J. Haas. *Secure Routing for Mobile Ad Hoc Networks*. In SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), January 2002.

[18] Charles Perkins and Elizabeth Royer. *Ad hoc On-Demand Distance Vector Routing.* MILCOM '97, November 1997.

[19] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and Doug Tygar. *SPINS: Security Protocols for Sensor Networks.* Wireless Networks Journal (WINE), September 2002.

[20] R. L. Rivest. *The RC5 Encryption Algorithm.* In Proceedings of the 1994 Leuven Workshop on Fast Software Encryption, Springer-Verlag, 1995.

[21] D. J. Wheeler and R. M. Needham. *TEA, a tiny encryption algorithm.* Lecture Notes in Computer Science, 1008:363-366, 1995.

[22] Yong Yao and J. E. Gehrke. *The Cougar Approach to In-Network Query Processing in Sensor Networks*. Sigmod Record, Volume 31, Number 3, September 2002.

[23] G. Yuval. *Reinventing the travois: Encryption/MAC in 30 ROM bytes.* Fast Software Encryption Workshop, Springer-Verlag, 1997.

[24] Lidong Zhou and Zygmunt J. Hass. *Securing ad hoc networks.* IEEEE Network Magazine, 1999.