ARCMA: Attack-Resilient Collaborative Message Authentication in Wireless Sensor Networks

Min-gyu Cho and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109-2121, USA
{mgcho,kgshin}@eecs.umich.edu

ABSTRACT

Data Centric Storage (DCS) is a well-known data storage and query processing mechanism for Wireless Sensor Networks (WSNs), storing sensed data or their metadata at pre-specified locations. Queries issued by mobile users are sent to, and processed at, such storage nodes. However, securing DCS is very difficult because WSNs usually operate in an unattended environment and hence are subject to node-capture attacks. Even after capturing a single node, an attacker may be able to subvert the entire system by using the keying material extracted from the captured node.

To remedy/alleviate the above problem, we propose *Attack-Resilient Collaborative Message Authentication* (ARCMA), in which sensor nodes collaboratively authenticate messages to be sent to, or received from, remote nodes. In ARCMA, each node belongs to one of k groups, and constructs an *Authentication Tree* (AT) which is formed with k nodes, each from a distinct group. Each node collaborates with the other nodes in its AT to authenticate messages. We propose two heuristics, called $\mathcal{MIN}$ and $\mathcal{OPT}$, to construct ATs. Our analysis shows that the security of ARCMA does not degrade until the attacker capture k or more nodes. We also evaluate the overhead of constructing ATs and the cost of authenticating messages using ATs.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General–Security and Protection

General Terms

Design, Security

Keywords

collaborative message authentication, sensor networks

1. INTRODUCTION

Data Centric Storage (DCS) [8] is a prominent data-storage and query-processing mechanism on Wireless Sensor Networks (WSNs). Resource-limited sensor nodes collaboratively identify the events of interest, and report them to storage nodes at predefined locations according to the event type or the range of sensed data. The stored event information can then be queried, typically by mobile users, from anywhere in the network, and the queries are forwarded to the relevant storage nodes. At storage nodes, the queries are processed and the results are sent back to the issuer(s).

Despite its importance, securing DCS is very difficult since WSNs usually operate in an unattended environment. Since sensor nodes may be physically exposed to attackers, it is relatively easy for attackers to capture and then reverse-engineer them to extract their secret (keying) information. In such a case, attackers can subvert the entire system by compromising even a single sensor node. The attacker may fabricate seemingly valid packets using the keying material extracted from the captured node to (1) insert fabricated information into the network, or (2) make unauthorized data accesses.

Detecting forged messages is very difficult if they are generated using the valid keying material, especially when they came from remote nodes. Key pre-distribution schemes [3, 4] may tolerate eavesdropping when some of sensor nodes are compromised, but they cannot prevent the issuing of queries or reporting of events using the valid keying materials. Even an expensive way of signing with public keys [7, 11] cannot handle this problem when messages are generated using valid (stolen) keying materials. Research has been on the detection of data forgery using reputation systems and statistical methods [5], the results of which are effective only for local data forgery, not for messages sent from remote nodes. En-route filtering mechanisms [12, 13] perform data-forgery detection even in the presence of captured/compromised nodes. However, these mechanisms rely on the existence of a base station, and cannot be applied directly to DCS, where data is sent to resource-limited storage (sensor) nodes, located at arbitrary locations.

In this paper, we propose *Attack-Resilient Collaborative Message Authentication* (ARCMA) to secure DCS, especially under node-capture attacks. ARCMA focuses on the authentication of event-report, event-query, and query-response messages that may take multiple hops to reach their destinations. Specifically, in ARCMA, a set of sensor nodes cooperate to evaluate a *Collaborative Message Authentication Code* (CMAC) for the messages sent to, and received from, remote nodes. Like any other traditional MAC, a CMAC is evaluated both at the sender and the receiver sides, and only those messages with valid CMACs are accepted by the receiver. Since the keying material of sensor nodes may be exposed under node-capture attacks, sensor nodes must collaborate to provide necessary protection against such attacks. This is akin to the sensor nodes' collaboration (to overcome their limitations in sensing capability) to identify the events of interest to the underlying applications.

The main contributions of this paper are summarized as follows. First, we propose a novel attack-resilient message authentication mechanism, called ARCMA, in which each node belongs to one of k groups, and a message is collaboratively authenticated and verified by k nodes, each from a distinct group. To achieve this, each node maintains an *Authentication Tree* (AT), which is a spanning tree with k nodes, each from one of k distinct groups, and collaborates with all the nodes in its AT to evaluate CMAC for message authentication. We also encrypt messages with location-dependent keys to validate their origin.

Second, we show how ARCMA can be applied to authenticate messages associated with DCS operations. We show how each parameter can be set for DCS messages, i.e., event-report, event-query and query-response messages. Most of previous work focused on securing event-report messages destined for the base station, and does not consider securing the processing of queries issued by mobile users.

Third, we present two distributed heuristic algorithms, $\mathcal{MIN}$ and $\mathcal{OPT}$, to construct ATs without any global view of the network since it is not available, or too expensive to obtain in a WSN.

Finally, we analyze the security of ARCMA and evaluate its performance. We show that the security of ARCMA does not degrade if less than k nodes are compromised. Also, we evaluate the overhead of constructing ATs and exchanging extra messages for message authentication for design parameter k and node density.

The rest of this paper is organized as follows. Sec. 2 describes the system model and assumptions, and the key management for ARCMA is described in Sec. 3. Then, ARCMA is detailed in Sec. 4, followed by descriptions of the AT construction Sec. 5. Sec. 6 presents the security analysis and the performance evaluation of ARCMA. The paper concludes with Sec. 7.

2. SYSTEM MODEL

2.1 Sensor Network Model

DCS operates on a WSN, which is usually composed of a large number of resource-limited sensor nodes. For these nodes, hardware-supported tamper-proofing is not a feasible solution due to its higher hardware cost. We assume sensor nodes are densely deployed such that events of interest can be detected by a set of cooperating sensor nodes near the locations of their occurrence. Detected events will be sent to remote storage nodes, where information about the event is stored or aggregated with other information. The storage nodes are nothing but sensor nodes, which are selected *a priori* according to a pre-defined rule. An event-report message may traverse a large number of hops if the storage nodes are located far away from the location of its occurrence. External mobile nodes, which may be carried by users or attached to ground/air vehicles, may interact with nearby sensor nodes, issuing queries for the information of interest. Sensor nodes will forward the such queries to the storage nodes, which will then reply with the queried information. The mobile nodes usually have more resources than sensor nodes.

In designing ARCMA, we assume existence of the following three common services: localization [9], time synchronization [10], and geographic forwarding routing [2]. All of these services are basic and usually required for other sensor network applications than DCS; localization and time-synchronization services are required to provide the location and the time of each identified event, and geographic forwarding routing is required to forward event-report and query messages. ARCMA exploits these "already available" services without incurring additional costs.

2.2 Attack Models

WSNs are inherently vulnerable to various security attacks as they usually operate in an unattended and hostile environment. Thus, we assume that attackers can launch any passive or active attacks including node-capture attacks. Under node-capture attacks, attackers will obtain all the valid keying materials from the compromised nodes. We assume that the goals of attacks with those keying materials are (1) to disable the targeted DCS by inserting false data so that the legitimate users will receive incorrect data, or (2) to access the stored data for unauthorized purposes. The main focus of this paper is to secure DCS against this type of severe attacks.

We, however, assume that the network is safe for a while after the deployment, during which the initialization can be performed. This is realistic, since it takes time for attackers to learn the existence and operation of a WSN, and to compromise sensor nodes.

3. KEY MANAGEMENT

3.1 Polynomial-based Key Pre-distribution

We first summarize polynomial-based key pre-distribution [1], on which the key management of ARCMA relies. A $(k{-}1)$-degree[1] bivariate polynomial is defined as $f(u,v) = \sum_{i,j=0}^{k-1} a_{ij}u^i v^j$ over a finite field F_q, where q is a prime number large enough to accommodate a cryptographic key. When a bivariate polynomial is symmetric (i.e., $f(u,v) = f(v,u)$), we define a share of the polynomial of node s as $f(s,v)$, which is a $(k-1)$-degree univariate polynomial. By distributing the share of polynomial to every sensor node, any two nodes in the network can set up a pairwise secret by exchanging only their node IDs. For example, two nodes s and r share a secret by replacing the variable of their share of polynomial with the correspondent's ID, i.e., $f(s,r) = f(r,s)$. The common secret between any two nodes is proven to be safe if less than k shares of the $(k-1)$-degree polynomials are revealed to the adversary, i.e., less than k nodes are compromised.

3.2 Key Distribution and Assignment

ARCMA uses three sets of bivariate polynomials to generate keys for different purposes. *Pairwise-key polynomial*, $f_p(u,v)$, is used to generate the pairwise key between any two sensor nodes in the neighborhood, or between a mobile node and a sensor node. Each polynomial variable will be replaced by the node ID. *Group-key polynomial*, $f_g(u,v)$, is used to evaluate the MACs of messages. One variable of the group-key polynomial will be replaced by group number, and the other by the node ID. *Spatial-key polynomial*, $f_s(u,v)$, is used to generate encryption-keys for messages forwarded to remote locations. Since the destinations of such messages are specified as their location (not ID) in DCS, the variables are replaced with the location information of the sender and the receiver of a given message.

Before their deployment, sensor nodes are preloaded with a common master key K_0. The coefficients of the above three-polynomials are derived from this key using the common keyed hash function $H_K()$. The coefficient a_{ij} of f_p are evaluated as $a_{ij} = H_{K_0}(i \cdot k + j + B)$ if $i \le j$, and $a_{ij} = a_{ji}$ otherwise, where B is set to 0. The coefficients of f_g and f_s are evaluated similarly by setting B to k^2 and $2 \cdot k^2$, respectively. K_0 is permanently removed from the memory after evaluating the coefficients of three polynomials.

Once deployed, sensor nodes evaluate their share of pairwise-key polynomial by replacing one of the variable with their own ID.

[1]We use $(k{-}1)$-degree polynomials instead of k-degree polynomials as in most literature to simplify notations in the later sections.

Then, each sensor node broadcasts its own ID, and establishes pairwise keys with its neighbors by evaluating the pairwise-key polynomial with the neighbors' IDs. After discovering its neighbor nodes, each sensor node generates a random number for a cluster key to encrypt local broadcast messages. The cluster key is individually delivered to all neighbors after being encrypted with pairwise keys. Let CK^s be the cluster key of sensor node s.

Once local relations are secured, the underlying localization protocol is evoked and then each sensor node evaluates its share of spatial-key polynomial. The pairwise keys and cluster keys established as above may be used to encrypt/decrypt messages of the localization protocol, if necessary. When evaluating the spatial-key polynomial, quantized coordinates are used instead of the raw coordinates. Since the exact coordinates of remote nodes are, in general, not known *a priori*, the exact coordinates of the destination cannot be used with the spatial-key polynomial to obtain a matching key both at the source and the destination. The quantized location (L_s) of a sensor s located at (x, y) is defined as $L_s = \lfloor \frac{x}{\ell_0} \rfloor || \lfloor \frac{y}{\ell_0} \rfloor$, where the quantization index ℓ_0 determines the granularity of quantization. Then, s's share of the spatial-key polynomial can be evaluated to be $f_s(L_s, v)$.

Group assignment is performed concurrently with the aforementioned localization. Each node can be assigned to a specific group either by a group-assignment protocol or by a pre-determined method, e.g., group number g of sensor node s is computed as $g = (s \bmod k) + 1$. Once group number is assigned, each node evaluates its share of f_g.

Each mobile node also has its share of the pairwise-key and group-key polynomials by replacing one of the variables with its ID, i.e., mobile node m will be loaded with $f_p(m, v)$ and $f_g(m, v)$. The mobile node need not hold any information on spatial-key polynomial.

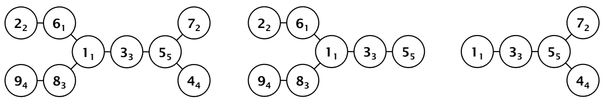
4. ATTACK-RESILIENT COLLABORATIVE MESSAGE AUTHENTICATION (ARCMA)

4.1 Collaborative Message Authentication

In ARCMA, sensor nodes collaboratively authenticate messages sent to, and received from, remote nodes. To prevent attacks using the valid keying material from the captured nodes, ARCMA requires messages to be authenticated collaboratively by a set of sensor nodes. Specifically, each sensor node belongs to one and only one of k distinct groups, and members of each group share a group-key polynomial to generate a common key to evaluate MACs. When authenticating a message, k nodes from k distinct groups, one from each group, evaluate MACs using group-keys, and the CMAC is evaluated by XORing these MACs.

To evaluate a CMAC, each sensor node needs to know at least one node for each of k groups in its neighborhood. Thus, each node maintains an AT, a spanning tree with at least k nodes to maintain information on nodes for each of k groups. Using its AT, a sensor node can evaluate or verify the CMAC. For convenience, the AT maintained by node s is represented by AT^s, and sensor nodes in AT^s are represented as AT_i^s, where i is the group the node belongs to. Each sensor s node maintains the following information in its AT: the node ID for AT_i^s, the next hop to AT_i^s, and the number of hops to AT_i^s for each group i ($1 \leq i \leq k$).

Figure 1 shows an example of ATs. Fig. 1(a) depicts a given topology, where vertices and edges represent nodes and connectivity, respectively. The main number and the subscript number in a vertex represent the node ID and its group number, respectively. Figures 1(b) and 1(c) show possible AT^1's when $k = 5$.



(a) An example topology (b) A possible AT^1 (c) Another AT^1

Figure 1: Example of authentication trees. The main number and the subscript number in a vertex represent the node's ID and group number, respectively. These figures show two possible examples of the authentication tree for node 1 for a given topology when $k = 5$.

We now describe how messages are exchanged when sensor node s located in L_s sends a message M to anther sensor node r located in L_r. In summary, s computes a CMAC for M and encrypts them with a proper spatial-key. When r receives M, it evaluates the CMAC after decrypting it, and checks if the evaluated CMAC and the received CMAC match. It accepts the message only when the two CMACs match.

Messages at the sender's side are exchanged as follows. First, s sends M to all nodes in AT^s with ID id, timestamp ts, its location, L_s, and a proof of message, P_M; id is a parameter other than group number for the group-key polynomial, ts is used to prevent replay attacks, L_s is used for the authenticity of the location of reported data, and P_M is used to prevent the compromised node from asking the message authentication for forged messages. Depending on the type of a given message, these fields will be filled in with different values. This will be detailed in Sec. 4.2. Second, the nodes in AT^s evaluate the MAC for a given message with the timestamp and the location of s, and then the evaluation result is returned to s. AT_i^s evaluates the MAC as $\text{MAC}(f_g(i, id), M|ts|L_s)$. Note that this is processed only when a valid proof P_M for the given message M is provided. Third, s computes the CMAC by XORing the MACs it received, where CMAC is defined as $\text{CMAC} = \text{MAC}(f_g(1, id), M|ts|L_s) \oplus \text{MAC}(f_g(2, id), M|ts|L_s) \oplus \cdots \oplus \text{MAC}(f_g(k, id), M|ts|L_s)$. Finally, s encrypts M with its spatial key, and sends it to r. s concatenates the timestamp, its location and CMAC with the message, and sends the concatenated to r using the underlying routing protocol.

When r receives the message, it evaluates the CMAC using its own AT similarly to the sender side as follows. First, r sends M to the nodes in AT^r. r decrypts the message with $f_s(L_s, L_r)$, and sends it to the nodes in its AT. Here, the proof of message is not required since r's location is different from s's location, L_s. Second, r receives the MACs evaluated by the nodes in AT^r. When the nodes in r's authentication tree, AT^r, receive the message, they evaluate the MAC for the given message with the group-key and return it to r. Third, r evaluates the CMAC and accepts the message only if CMACs match. Upon receiving replies from the nodes in AT^r, r evaluates the CMAC for the given message by XORing the received MACs. Only if the CMAC from s matches the CMAC it computed, r will accept the message.

In the above process, the same message M is transmitted to all the nodes in AT^s and AT^r. This will be very expensive, thus calling for an optimization to reduce the number of message transmissions. Without optimization, the cost of exchanging messages to evaluate the CMAC increases linearly with k. However, we need not send the same message to all the nodes individually, since those nodes may share the same node as the parent node in AT^s and the wireless transmission can be simultaneously received more than one node. When sending a message, s can encrypt it with its cluster key, CK^s, and a node that forwards the message asking for MAC, i.e., it has child node(s) in AT^s, it can again send the message using its own cluster key. When MACs are received, the intermediate nodes in a node's AT can XOR all the MACs they receive from their children, and then report the XORed MACs to their parents. When s receives such reports from its children, it can compute the

CMAC by XORing them. The CMAC obtained from this optimization will be the same as that obtained by XORing individual MACs at s since XOR is transitive. When the CMAC is verified by r, a similar step can be taken.

A CMAC mismatch indicates a compromise. If the CMAC computed by r does not match the CMAC from s, at least one node in either AT^r or AT^s did not report a proper MAC. In such a case, r can send the same message M to all the nodes in AT^r individually without optimization. Then, r can send all of the MACs to s. Upon receiving such a message, s also requests nodes in AT^s to re-evaluate the MACs. If the MACs for a certain group do not match, either the nodes of a group in AT^s or AT^r, or both are compromised. In such a case, both nodes are revoked. This revocation process is expensive since the same message should be sent to all the nodes in AT^s and AT^r. However, this type of attack is *not* effective from an attacker's standpoint not only because the attack is easily detectable but also because the compromised nodes are revoked upon their first attack.

4.2 Securing DCS Operations Using ARCMA

We now describe how ARCMA can be used for securing DCS messages: event-report, event-query, and query-response messages. The collaborative authentication described so far can be used for transmission of messages to remote locations, regardless of their type. However, different proofs of messages for CMAC evaluation are required for different types of messages.

Reporting Sensed Data Sensor nodes cooperate to identify events of interest; an event will typically be detected by multiple sensor nodes, and their sensor readings will be aggregated for detection accuracy. When such an event is reported to the corresponding storage nodes, the report message must be authenticated to guarantee its authenticity. ARCMA can be used for this purpose, and the information about the event can be used as the proof of the message. The node density of a WSN is determined so as to cover every region with more than a certain number of nodes [6] for detecting each event of interest with a desired level of accuracy since each sensor node has only limited sensing capability. By adjusting the sensor node density, we can, therefore, make an event detectable by at least k nodes which can participate in authenticating the event to be reported to the corresponding storage node. These nodes evaluate MAC only when the information in the event-report message is consistent with the information they have, i.e., they use the information about the event as a proof of the message. The id field of ARCMA messages is filled with the ID of the sensor node that composes each event-report message.

The storage node verifies the authenticity of each event-report message using its own AT, and stores only validated messages. Also, the sensor nodes in the storage node's AT may store the data to serve as replicas of the storage node.

Processing Queries Issued by Mobile Users An external mobile user may query the sensor network to retrieve the data stored in DCS. Such queries are routed via multiple hops to the storage nodes, and then the corresponding results are generated and sent back to the mobile user. In this scenario, both the query and the result should be verifiable.

To guarantee that only a legitimate mobile user can issue valid queries, we make the mobile user to be authenticated by multiple sensor nodes before issuing any query to the network. When authenticated, the mobile user will be issued a token, which is a security credential it can present to the sensor nodes that belong to the same group of the token-issuer. After collecting k tokens, the mobile user can issue a query by presenting tokens as the proof of event-query, i.e., the query is validated by the security credentials issued by k different sensor nodes.

Mobile node m can be authenticated by sensor node s by exchanging three-way handshakes as follows. First, m sends s its ID and a nonce, N_m. Second, when s receives such a message, it also generates a nonce, N_s. Then, s sends its own ID along with N_m and N_s after encrypting it with a pairwise key between s and m, $f_p(s, m)$. Note that both s and m can evaluate the pairwise key using the pairwise-polynomial after exchanging their IDs. Finally, m sends the nonce generated by s after encrypting it with $f_p(s, m)$. Now, s and m can authenticate each other.

After authenticating a mobile node m, s sends m a token, $T_g(m)$, if it belongs to group g. $T_g(m)$ is defined as $T_g(m) \equiv s|g|m|ts|\{s|g|m|ts\}_{f_g(g,m)}$. This token is valid only to the nodes in group g, and only m can present this token to the sensor nodes since only m can evaluate the pairwise-keys with arbitrary sensor nodes. Here, ts is a timestamp to indicate the time when the token is issued, and may be used by sensor nodes to reject stale tokens.

When m issues a query Q, it is handled as follows. First, m sends Q to a nearby sensor node s with the k tokens it has collected as the proof of message. Second, when a sensor node s receives Q with the tokens, it sends this information, the current timestamp, ts, and its location L_s to the nodes in its AT. Third, the nodes in AT^s verify the token, which is the proof of event-query, and sends s either MAC for the query message or *invalid* message. Here, AT^s_i evaluates MAC as $MAC(f_g(m, i), Q|ts|L_s)$. Fourth, when s receives all the MAC, i.e., all the tokens are verified without generating *invalid* message, it evaluates $CMAC$ by XORing collected MACs. Finally, s sends the query message to the storage node r after encrypting the message using spatial key with $f_s(L_s, L_r)$.

When a storage node r receives the query, it first verifies the message using AT^r, and then composes the query-response, R, for the valid query. The reply message is again authenticated using AT^r, and then encrypted with the pairwise-key between r and m, $f_p(r, m)$, which is evaluated using the pairwise-key polynomial so that the clear text of message is readable only by m. When m, the query issuer, receives this message, it evaluates CMAC using its share of the group-key polynomial, and accepts it only when a valid CMAC is concatenated to the reply message.

5. AUTHENTICATION TREE CONSTRUCTION

Since a global view is not available at any given sensor node, we present two simple heuristic algorithms to construct ATs without incurring too much overhead. The first heuristic, called $\mathcal{MIN}$, builds a minimum-depth tree by selecting the shortest-distance node from a certain group for the AT. This heuristic adds a small amount of information in a periodic beacon message, commonly used in sensor network applications like health monitoring, node discovery, and route discovery. In each periodic beacon message, a node advertises the groups it has discovered by using a k-bit vector. When a sensor node notices that some bits are not set in a neighbor's beacon while the corresponding bits are set in its own bit vector, i.e., its neighbor has not yet discovered the nodes in certain groups that it has already discovered, it announces the information about them in its next beacon message. When a sensor node detects such information in its neighbor's beacon, it updates its AT. Using these steps, a t-hops-away node can be discovered within t beacon periods, as in a distance-vector routing protocol.

The second heuristic, called $\mathcal{OPT}$, tries to minimize the number of transmissions in the optimized AT operation. For example, the $\mathcal{MIN}$ heuristic yielded the results in Fig. 1(b) for the topology in Fig. 1(a). This AT requires four transmissions since nodes 1, 3, 6, and 8 need to transmit a message for an AT operation. On the other hand, the AT shown in Fig. 1(c) for the same topology requires only three transmissions for an AT operation. Even though the nearest nodes are not chosen for groups 2 and 4, a transmission by node 5 can cover both groups in this example. The $\mathcal{OPT}$ heuristic works as follows. After discovering one's neighbor, each node keeps two sets of nodes; transmission set (T) and candidate set (C). The transmission set contains the nodes which will relay the message in AT operations. First, each node puts itself in T, and its neighbors to C after calculating the out-degree, which is defined as the number of new groups of each neighbor if it is added. At each iteration, the node with the maximum out-degree will be moved from C to T, and the neighbors of the node will be added to C. When there is no node that has a positive out-degree in C, a sensor node will request the route for the missing groups. Upon receiving such a request, a sensor node replies if it has access to the group.

6. EVALUATION

6.1 Security Analysis

We analyze the security of ARCMA against the various attacks described in Sec. 2.2. Since all the messages are encrypted with the relevant keys, the proposed protocol is safe from passive or active outsider attacks as long as a proven cryptography is used. Also, replay attacks are prevented since all the messages including tickets contain time-stamps. Thus, our analysis will focus on node-capture attacks, where valid keying materials are exposed to attackers.

An attacker should acquire k group keys to obtain a valid CMAC for the given messages. Since the security of a $(k-1)$-degree bivariate polynomial is preserved as long as no more than $k-1$ nodes are captured [1], the attacker should capture at least k nodes to obtain all the k keys to derive a valid CMAC. Therefore, the attacker cannot insert false data into storage, or make unauthorized data accesses by issuing queries, i.e., the system remains secure as long as less than than k nodes are captured.

Also, ID spoofing and Sybil attacks can be prevented by using the bivariate polynomials. Without knowledge of the pairwise-key polynomial, the attacker cannot generate the secret keys for the spoofed IDs. Therefore, the attacker cannot launch a Sybil attack with spoofed IDs.

However, denial-of-service (DoS) attacks are still possible. When a compromised node is asked to authenticate a valid mobile node, it may not respond at all, or may issue an invalid token. When this occurs, the performance of ARCMA may degrade. Similar DoS attacks could occur when a compromised node is asked to verify the query at the query location. This attack may be detected with NIDS using abnormal behavior activities, but it is beyond the scope of this paper.

6.2 Performance Evaluation

Memory Requirement A sensor node has only limited memory. Even though this may become a lesser problem in future due to the decreasing cost and increasing density/capacity of memory, the memory will still be a limited resource for sensor nodes or a larger memory may consume the battery power much faster.

Each sensor node needs to store three polynomials, each of which has k terms, each being a member in F_q, where q is a prime number large enough to hold the security keys. In ARCMA, only symmetric cryptographic functions are used, and a 128-bit key will be sufficient. Thus, each sensor node needs the memory of $3 \cdot k \cdot 128$ bits $= 384 \cdot k$ bits.

Each sensor node also maintains an AT. For each of the k groups, the ID of the node, the ID of the next-hop node, and the number of hops should be stored. When 16-bit node IDs are used and the maximum hop count is less than 255, the storage requirement for the AT at each node is $(2 \cdot 16+8) \cdot k$ bits $= 40 \cdot k$ bits.

The intermediate nodes in an AT need to store the information about the nodes in its own subtree. A k-bit vector should be stored for this purpose. Since each node will, on average, be an intermediate node of $k-1$ other nodes, the node needs $k \cdot (k-1)$ bits.

The total memory requirement for ARCMA is the sum of the above three requirements, resulting in $k^2 + 423 \cdot k$ bits. When $k = 20$ for example, the memory requirement is about 1.1 Kbytes.

CMAC Evaluation and AT Construction Costs We evaluate the two heuristics to build ATs in terms of the CMAC evaluation and AT construction costs. The CMAC evaluation cost is defined as the average number of transmissions to deliver a message to all the nodes of the AT. We average the number of transmissions by dividing it by the number of groups in an AT to show the cost of reaching a node (or group). The construction cost is defined as the average number of iterations and route request/reply messages per group to build the ATs.

We also show the cost of reconstructing the AT when a sensor node is removed from the network since it is either faulty or compromised. When such a problem node is removed from the network, all the nodes in the subtree rooted at the problem node should also be removed. Then, ATs should be reconstructed without using the problem node for the subsequent CMAC evaluations. We will adopt the number of nodes affected by the problem node and the number of iterations required to reconstruct the ATs as the evaluation metrics of the AT reconstruction cost.

The simulation parameters are set as follows. We assumed the transmission range (R) of the sensor node to be 100 m (equal to that of MicaZ). All simulations are conducted for a 1000 m $\times$ 1000 m coverage area. The number of nodes (N) in the area is varied to be 250, 500 and 1000, i.e., the average number of neighbors varies from 7.85 to 31.4. The number of groups, k, is varied from 2 to 50, and the pre-determined group assignment, $g = (s \bmod k) + 1$, is used. For each of N and k, 10 different topologies are generated using a uniform distribution. We also changed the network size while maintaining the same node density. Finally, all the simulation results are derived by averaging 10 simulation runs.

Figure 2 shows the CMAC evaluation cost of the two heuristics. When $k = 2$, the average CMAC evaluation cost is around 0.5, meaning that only 1 transmission is required, because at least one of a node's neighbors will likely belong to a different group than its own in a dense network. As k increases up to a certain point, the average cost decreases since all or most of groups are likely to be found within a single or a very small number of hops. However, the average cost slowly increases as k increases under the $\mathcal{MIN}$ heuristic, while it remains stable under the $\mathcal{OPT}$ heuristic. From these observations, we conclude that the $\mathcal{OPT}$ heuristic requires fewer transmissions, i.e., less energy consumption for an CMAC evaluation than the $\mathcal{MIN}$ heuristic. Also, a denser network is observed to incur a lower cost in both heuristics because a node will have more neighbors, i.e., a smaller number of hops will be required to reach a certain group in a denser network.

The costs of constructing ATs with the two heuristics are shown in Fig. 3. Fig. 3(a) shows the number of iterations to construct ATs. With both heuristics, a sparser network takes longer to build ATs than a denser network, as it will take more hops to reach all the groups in a sparser network. Figure 3(b) shows the average number of route request/reply messages. As expected, the required number of messages decreases as N increases. When comparing the two heuristics, we observe that the $\mathcal{MIN}$ heuristic exchanges fewer messages than the $\mathcal{OPT}$ heuristic in a sparser network. On

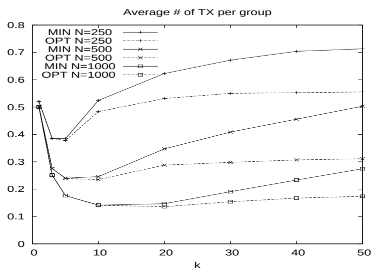
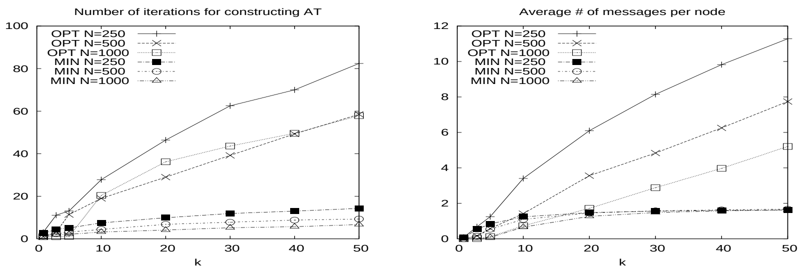


Figure 2: The average number of transmissions per group of $\mathcal{MIN}$ increases as k increases while that of $\mathcal{OPT}$ increases at a much slower pace.



(a) The number of iterations (b) The number of messages

Figure 3: These graphs show the cost of constructing ATs with $\mathcal{MIN}$ and $\mathcal{OPT}$ heuristics while varying k and node density. $\mathcal{MIN}$ takes less number of iterations and messages than $\mathcal{OPT}$ when constructing ATs.

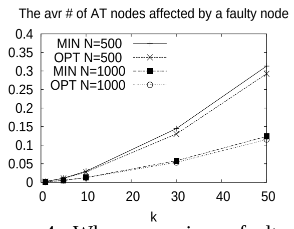


Figure 4: When removing a faulty node, the average number of affected nodes per AT increases as k increases for both heuristics.

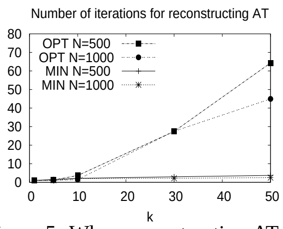


Figure 5: When reconstructing ATs, the number of iterations remain at a small number in $\mathcal{MIN}$, while it increases as k increases in $\mathcal{OPT}$.

the other hand, the $\mathcal{OPT}$ heuristic exchanges fewer messages in a denser network than the $\mathcal{MIN}$ heuristic.

Figures 4 and 5 plot the performance evaluation results when a node is removed from the network. Fig. 4 shows the average number of nodes removed for each AT upon occurrence of a problem node. There was only a small difference between the two heuristics. However, the number of affected AT nodes increases as the network gets sparser or k increases. In a sparser network, there are unlikely many alternative routes from a node to other nodes, i.e., a node is likely to be on more routes. Thus, one problem node can affect more nodes. When k increases, the number of alternative nodes of a certain group decreases. Fig. 5 shows the number of iterations to finish the reconstruction of ATs. $\mathcal{MIN}$ is found to take much fewer iterations than $\mathcal{OPT}$. This means that it will take much longer to construct ATs with the $\mathcal{OPT}$ heuristic when a node becomes faulty or compromised.

We also evaluated the performance of ARCMA for different network sizes while maintaining the same node density. However, we found that the network size does not affect the performance of ARCMA for the same node density, and thus, we did not repeat the result here.

7. CONCLUSION

In this paper, we proposed a novel security mechanism for DCS, called ARCMA, especially against node-capture attacks in WSNs. Under such attacks, securing DCS is very difficult since attackers have access to all the valid keying materials extracted from captured nodes. In ARCMA, sensor nodes collaboratively authenticate event-reporting and query-processing messages which typically travel multiple hops. Each sensor node belongs to one of k groups, and builds and maintains an AT with k nodes, each from a distinct group. With the help from the nodes in its AT, a sensor node can evaluate CMACs to authenticate messages.

When an event is identified by a set of cooperating sensor nodes, they generate an event-report with CMAC, which will then be forwarded to, and verified by, storage nodes. Before a mobile user issues a query, s/he needs to collect k tokens by contacting k sensor nodes, one from each distinct group. When a query is issued along with these tokens, it is authenticated as was done for an event-report message, and then forwarded to storage nodes. Finally, the storage node verifies and processes the valid message. Likewise, a response to the query can be authenticated.

ARCMA is shown to be $(k-1)$-collusion resistant, i.e., the same level of security is preserved if no more than $(k-1)$ nodes are captured. We also presented and evaluated two heuristics ($\mathcal{MIN}$ and $\mathcal{OPT}$) to build ATs. $\mathcal{MIN}$ builds ATs using the nearest nodes from a specific group, which constructs ATs within a much fewer iterations both when ATs are constructed from scratch and when ATs are reconstructed after revoking faulty or compromised nodes. On the other hand, $\mathcal{OPT}$ incurs a lower operation cost (i.e., less transmissions to send one message to all the nodes in an AT) and a slower growth in the operation cost, while it takes longer and more messages to build ATs.

8. REFERENCES

[1] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Proc. of CRYPTO '92*.

[2] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wirel. Netw.*, 7(6):609–616, 2001.

[3] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proc. of CCS '03*.

[4] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. of CCS '02*, pages 41–47, 2002.

[5] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proc. of SASN '04*.

[6] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *Proc. of MobiCom '04*.

[7] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proc. of SECON '04*.

[8] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proc. of WSNA '02*.

[9] M. Rudafshani and S. Datta. Localization in wireless sensor networks. In *Proc. of IPSN '07*.

[10] K. Sun, P. Ning, and C. Wang. TinySeRSync: secure and resilient time synchronization in wireless sensor networks. In *Proc. of CCS '06*.

[11] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. TinyPK: securing sensor networks with public key technology. In *Proc. of SASN '04*.

[12] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. *IEEE JSAC*, 23(4):839–850, 2005.

[13] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Proc. of the IEEE S&P '04*.