A Novel ID Anonymity Preserving Scheme (ID-APS) for Hierarchical Wireless Sensor Networks

Ahmed Al-Riyami, Ning Zhang and John Keane
School of Computer Science
The University of Manchester
Manchester, UK
Email:{ahmed.al-riyami,ning.zhang,john.keane}@manchester.ac.uk

***Abstract*—Node ID anonymity is a critical aspect of privacy in Wireless Sensor Networks (WSNs). Exposing node IDs can help adversaries to learn the underlying network infrastructure and to mount more serious attacks, such as attacks on important nodes (e.g., cluster heads). Therefore, providing ID anonymity should be an integral part of the solution to secure WSNs, and should not hinder other security properties such as accountability and intrusion detection. However, the latter requires that communication nodes be identifiable globally in a dynamic WSN where there is no fixed infrastructure support. Achieving these contradicting security properties effectively and efficiently is challenging. This paper proposes a method, termed ID Anonymity Preserving Scheme (ID-APS), to preserve node ID anonymity with a global node identification capability in hierarchical WSNs. Evaluation results show that ID-APS achieves these properties at a lower level of costs (computational, communication and memory overheads) than comparable methods.**

Keywords–Wireless Sensor Networks; privacy; ID anonymity; pseudonym.

I. Introduction

A critical aspect of privacy in Wireless Sensor Networks (WSNs) is node ID anonymity. By knowing the IDs of the nodes in a WSN, an adversary may obtain information about the communication relationships among the nodes, and infer the network topology which can help to launch successful attacks on important nodes such as cluster heads. Inference of network topology through analysis of node IDs is known as *ID analysis attack* [1].

Passive adversaries launch ID analysis attacks by eavesdropping on network communications; they overhear messages exchanged over wireless media and view source and destination IDs in the message headers. *Active* adversaries launch ID analysis attacks through node compromise; they physically capture a node and access data stored in its memory (e.g., clustering information), identifying the network topology.

Eavesdropping and node compromise attacks are hard to thwart due to the nature of WSNs, e.g., open wireless media and unattended nodes. Existing solutions address these attacks by using dynamic pseudonyms [2]–[5]. These solutions assign a unique pseudonym ID to each node in each transmission, disguising the node's real ID. However, there are a number of issues with these solutions. The first issue is that some solutions, e.g., [3], [4], only consider anonymising node IDs in unicast communications; communication among sensor nodes involves broadcasts as well. If ID anonymity is only provided in unicast communications, then an adversary may still be able to identify nodes through broadcasts. The second issue is how to reduce costs incurred in preserving ID anonymity. Take pseudonym updating and synchronization as an example, some solutions, e.g., [4], [5], require each pair of communication nodes to use acknowledgments to convey and synchronise the values of a set of pseudonym parameters after each packet transmission between the pair. This raises two further issues. One is that acknowledgments consume network bandwidth and sensor node energy, and increase network traffic levels causing congestions and packet collisions. The second is that acknowledgments are impractical for broadcast traffic, as the resulting acknowledgments will flood the entire network. However, if no mechanism exists to allow communicating nodes to synchronise broadcast pseudonyms, a corrupt or lost message could lead to unsynchronized parameter values hindering further anonymous communication.

Further, preserving node ID anonymity should be accompanied by a secure node global identification facility. This is because the system will also provide other important security properties, e.g., intrusion detection and node compromise detection where a monitoring node needs to report suspicious nodes to the base station. Providing node ID anonymity should enable, rather than hinder, these properties.

This paper proposes a secure solution, the ID Anonymity Preserving Scheme (ID-APS) for hierarchical WSNs. ID-APS offers anonymity and efficiency improvements over the Cryptographic Anonymity Scheme (CAS) [2], the most relevant existing solution. It provides a trade-off between ID anonymity level and computational costs, while minimising communication overheads and memory consumptions. Further, ID-APS also provides an additional built-in functionality to support controlled global node identifiability, i.e., each node in ID-APS can be globally identified by authorised entities. Such flexibility makes ID-APS highly suitable for a wider range of applications as compared with previous solutions.

The rest of the paper is structured as follows: related work is discussed in Section II; the CAS scheme is described in Section III; Section IV discusses how ID-APS improves CAS; Section V presents design preliminaries; the ID-APS scheme is described in Section VI; Section VII analyses the scheme and finally, Section VIII concludes the paper.

II. Related Work

Misra and Xue [2] address anonymity through dynamic pseudonyms in clustered WSNs. They proposed two anonymity schemes, Simple (SAS) and Cryptographic (CAS). In SAS, each node is assigned a randomly distributed set of pseudonym sub-ranges selected from a large space. The node chooses a different pseudonym to identify itself in each transmission. SAS is efficient as it does not involve complicated computations; however, it requires space to store the pseudonym sub-ranges. CAS reduces the high memory usage of SAS by generating pseudonyms on a per-message basis at run-time by key hashing a sequence number and a pre-defined seed shared between communicating nodes. CAS is more efficient than SAS in terms of memory consumption but at the cost of additional run-time computation. Nevertheless, if an adversary compromises a node in CAS, the adversary may compute the previous IDs of the node using the compromised secret key and the current sequence number.

To protect past used pseudonyms against node compromising attacks, Ouyang et al. [3] proposed the Hashing based ID Randomization (HIR) scheme. Similar to CAS, HIR uses a keyed hash function to generate dynamic pseudonyms. Each outgoing message is identified by a new pseudonym that is generated by key hashing the pseudonym used in the previous message. After each message transmission, the sender creates a new pseudonym and deletes the old one. So, if an adversary obtains the current pseudonym and the secret key, it is still difficult to derive the previous used pseudonyms because of the one way property of the hash function. However, the adversary can still compute the next pseudonym ID and hence impersonate the compromised node to send future messages once the key is captured. To resist such attacks on future messages, the same authors proposed the Reverse Hashing ID Randomization (RHIR) scheme. In RHIR, the one-way hash chain is used in the reverse order, i.e., pseudonyms are assigned backwards from the end of a pre-generated hash chain. This method is more secure against impersonation attack, however, RHIR consumes more memory to store the hash chain and provides each node with a limited number of pseudonyms. In addition, both HIR and RHIR only address anonymous unicast communication not broadcast.

Another scheme to support node ID anonymity is the Anonymous Path Routing (APR) scheme [4]. It hides node IDs using dynamic pseudonyms and uses per-message keys to encrypt each transmitted message. APR only provides node ID anonymity for unicast. For broadcast, (e.g., when a node broadcasts an anonymous path routing request), the broadcasters' IDs are exposed. Hence, an adversary can learn a source node ID from broadcast messages.

Chen and Fang [5] proposed the Efficient Anonymous Communication (EAC) scheme. Unlike other solutions, EAC supports both anonymous unicast and broadcast. Once a message is sent, the sender and receiver generate the next message pseudonym independently using a hash function. The hash function input includes the current message pseudonym and a parameter value shared between the two nodes. To synchronize the unicast pseudonyms used by the two communicating nodes, EAC uses anonymous acknowledgments. After receiving an acknowledgment sent by the receiver, the sender updates the next message pseudonym. However, this work does not mention how local broadcast pseudonyms, which are used between a sender and all its direct neighbours, are synchronised. Without a broadcast pseudonym synchronization solution, the use of anonymous broadcasting may not be practical in areas where message loss and transmission errors are probable.

III. The CAS Scheme

Pseudonyms in CAS are generated on a per-message basis using a keyed hash function and a number of parameter values. The parameters include keys, random seeds and message sequence numbers. Each node maintains a pseudonym table containing the parameter values for each other node in its neighbourhood. Two phases are involved: during the setup phase, nodes exchange and update the parameter values; during the operational phase, they use the values to compute and verify pseudonyms for anonymous unicast and broadcast communications.

A unicast message from node u to the base station via node v is composed as: $M_{uv} = SID \parallel RID \parallel EncryptedPayload \parallel seq_{uv}$ where SID is the end-to-end mutual pseudonym, $SID = Index_v \parallel H_{K_{Bu}}(a_{Bu} \oplus seq_{uv})$ and RID is the next-hop mutual pseudonym, $RID = Index_v \parallel H_{K_{uv}}(a_{uv} \oplus seq_{uv})$, $Index_v$ is the index used to index node u's parameter values in node v's pseudonym table, K_{Bu} and a_{Bu} are the hash pairwise key and the random seed shared between node u and the base station, respectively, K_{uv} and a_{uv} are the hash pairwise key and the random seed shared between node u and node v, respectively and seq_{uv} is the current message sequence number for mutual communication between nodes u and v. When v receives message M_{uv}, it retrieves $Index_v$ and searches its pseudonym table for a match. If a match is found, node v uses the corresponding values of K_{uv} and a_{uv} to compute $H_{K_{uv}}(a_{uv} \oplus seq_{uv})$. If the computed value equals the received value, node v accepts the message, otherwise it drops the message. When node v forwards the message to the base station, it includes the received sequence number seq_{uv}. When the message reaches the base station, the base station computes $H_{K_{Bi}}(a_{Bi} \oplus seq_{uv})$ where i refers to every node in the network. Then, it compares the computed hash values with the hash value received in the message to identify its original source.

A local broadcast from cluster head u to its neighbours is composed as: $M_{u*} = SID \parallel RID \parallel EncryptedPayload \parallel seq_{cu}$ where SID and RID are both pseudonyms used to identify the sender, u, $SID = Sentinel \parallel H_{K_{cu}}(a_{cu} \oplus seq_{cu})$ and $RID = Sentinel \parallel H_{K_{cu}}(b_{cu} \oplus seq_{cu})$, where $Sentinel$ is a special character indicating that the broadcast is sent by the cluster head, K_{cu} is the cluster key, a_{cu} and b_{cu} are random seeds shared between the cluster head and all other nodes in the cluster and seq_{cu} is the current message sequence number. Upon receiving the broadcast, a cluster member node identifies the $Sentinel$ to confirm that the message is a broadcast from the cluster head. The node then retrieves the parameter values related to the cluster head from its own pseudonym table and uses the values to compute new hash values $H_{K_{cu}}(a_{cu} \oplus seq_{cu})$ and $H_{K_{cu}}(b_{cu} \oplus seq_{cu})$. If the computed hash values match with the received ones, then the node confirms that the message is from the respective cluster head.

IV. Properties of ID-APS

CAS can be improved on three fronts: (i) integrate global node identifiability with ID anonymity, (ii) strengthen the ID anonymity protection level, and (iii) reduce overheads in achieving these properties. These improvements are accomplished using the following ideas:

- *For (i)*, this is achieved by a hybrid combination of fixed (static) and dynamic pseudonyms. Each node can be globally identified using a fixed pseudonym, while their communications are carried out by using per-message dynamic pseudonyms.

- *For (ii)*, two measures are used. The first is to break any association between a currently used pseudonym and the pseudonyms used in the past. So, the use of a random number attribute is introduced in dynamic pseudonym generations. In this way, if the current pseudonym is compromised, the previously used pseudonyms remain safe. The second measure is to hide any patterns or hints that may help adversaries to identify a particular node or a node performing a particular function. For example, CAS uses special characters ($Sentinel$) to identify cluster heads in broadcasts. These sentinels may be used to identify cluster heads as only cluster heads use them. To hide any patterns, we use the same mechanism (indexes) for unicasts and broadcasts and these indexes are randomly picked by the node concerned during the setup phase of the network. We also make the structure of the broadcast pseudonyms identical to that of the unicast pseudonyms, making it harder for an adversary to learn any identifiable information in the network.
- *For (iii)*, inspired by [6], the sequence number (i.e., message counter) is divided into two parts: one part is explicitly sent as part of the pseudonym in a message and the other part is hidden from transmission (i.e., stored in the node's memory). In this way, the sender and receiver's counters do not need to be tightly synchronized. As long as the number of consecutively lost messages does not exceed a certain threshold, the receiver can still verify the dynamic pseudonym, learn the message sender and synchronise its counter value with the sequence value received. This, along with the reduction in the number of overhead bits each message has to carry, can lead to both bandwidth and node battery savings.

V. PRELIMINARIES

This section presents the system model, assumptions and requirement specifications used in the design of ID-APS.

A. System Model

The WSN consists of a number of homogeneous resource-constrained static sensors and a single resource-rich base station (BS). All network links are assumed to be bidirectional. The network is partitioned into a set of clusters, each with an elected cluster head (CH). Sensor nodes periodically collect measured data and locally aggregate the collected data within each cluster before sending the aggregated data to the BS. Communication modes in this setting include:

- Broadcasts from the BS to all network nodes.
- Unicasts between the BS and any other network node.
- Broadcasts from a node to all its one-hop neighbours.
- Broadcasts from the CH to all cluster member nodes.
- Unicasts among nodes in the same cluster (or tier).

To maximise security, the principle of separation of duties is adopted to establish cryptographic (i.e., symmetrical) keys. In other words, different keys are used for different purposes as follows:

- *Network key* (k_N): This is the network-wide key shared between the BS and all nodes in the network. It is used for securing messages broadcast by the BS.
- *Individual Key* (k_{Bi}): Each node i shares a unique individual key with the BS. The key is used to secure pairwise unicasts between the node and the BS.
- *Pairwise Key* (k_{ij}): Each node i shares a unique pairwise key with each of its one-hop neighbours, j. This key is used by node i to secure unicast messages to node j and vice versa.
- *Broadcast Key* (k_i^*): Each node i shares a unique broadcast key with each of its one-hop neighbours. This key is used to secure local broadcasts by node i to its neighbours.
- *Cluster Broadcast Key* ($k_i^{\odot}$): This key is shared by a CH i and other member nodes in the cluster. It is used by the CH to secure all local broadcasts to cluster members.

k_N and k_{Bi} are generated by the BS and pre-loaded into each node prior to its deployment. k_{ij} and k_i^* are established using the Energy-efficient Distributed Deterministic Key management scheme (EDDK) [7]. In addition to key establishment, EDDK is also used for node discovery and periodic secure updates of k_{ij} and k_i^*. $k_i^{\odot}$ is established by the CH. The CH generates the key using a pseudorandom function and securely unicasts it to every cluster member using the pairwise keys.

Adversaries try to identify the nodes and the communication relationships among them. To do this, they try to access all available information by any means (passive or active attacks). The notations used in this paper are summarised in Table I.

B. Assumptions

- Each node $i \in \mathbb{N}$ has a unique identity, ID_i.
- The BS is always available, trustworthy and protected against physical attacks.
- The clocks of the BS and all the sensor nodes are synchronized.
- Broadcasts from the BS are done by using the μTesla authenticated broadcast method [8].
- Sensor nodes are able to obfuscate address fields in their Medium Access Control (MAC) layer header. This assumption is necessary to scope our work without losing generality. A solution to prevent sensor nodes from leaking their MAC level IDs is to use the dynamic pseudonyms at the MAC level as well.
- Each node i maintains a pseudonym table T_i. T_i stores the attribute values associated with the BS and each of the node's neighbours as shown in Table II.

C. Design Requirement Specifications

(a) Node ID anonymity:

- NIP-1: A passive adversary should be unable to learn the identity of the source or destination of a unicast message.
- NIP-2: A passive adversary should be unable to learn the identity of the source of a broadcast.
- NIP-3: An active adversary should be unable to learn the identities of uncompromised nodes through eavesdropping on their unicast communication.
- NIP-4: An active adversary should be unable to learn the past identities of a compromised node.

(b) Node Global Identification:

- GLB-1: A sensor node should be able to globally identify other network nodes without exposing the node IDs to an eavesdropping adversary.

(c) Message Security:

- MSE-1: All messages transmitted should be confidential.

TABLE I. ID-APS NOTATION

Notation	Definition
$\mathbb{N}$	Set of all nodes in WSN where total number of nodes $\lvert\mathbb{N}\rvert = N$.
T_i	The pseudonym table of node i.
N_i	Set of all nodes that have their details stored in T_i.
$\mathbb{A}_i$	Subset of N_i which includes all nodes that node i is allowed to communicate with based on node i's role.
α	Secret only known by the BS.
β	Shared secret known by the BS and all network nodes.
$H_k^{ID}(.)$	Keyed one-way hash function with output of l_{ID} bits long.
$H_k(.)$	Keyed one-way hash function with output of l_H bits long.
$\oplus$	Bitwise exclusive-or (XOR) operation.
ID_B	Real identity of BS. All IDs are l_{ID} bits long.
ID_i	Real identity of node i exclusively known to node i and BS.
$ID_i^{'}$	Fixed initial pseudonym assigned to node i.
$ID_i^{''}$	Fixed operational pseudonym created for node i.
$ID_{i\to B}$	Dynamic pairwise pseudonym computed by node i; it is used in unicast messages sent to BS.
$ID_{i\to j}$	Dynamic pairwise pseudonym computed by node i; it is used in unicast messages sent to node j.
$ID_{i\to *}$	Dynamic broadcast pseudonym computed by node i; it is used in broadcast messages sent to neighbours.
k_N	Network key shared by BS and all network nodes. All keys used are l_k bits long.
k_{Bi}	Individual key shared between node i and BS.
k_{ij}	Pairwise key shared between nodes i and j.
k_i^*	Broadcast key shared between node i and all its one-hop neighbours.
$k_i^{\odot}$	Cluster broadcast key shared between a CH node i and all member nodes in the cluster.
$ind_{i\to j}$	Unicast index shared between nodes i and j. All indexes are l_i bits long.
$ind_{i\to *}$	Broadcast index of node i shared between node i and its neighbours.
$H_{i\to j}$	Pairwise hash value used for the construction of dynamic pseudonyms in messages sent from node i to node j.
$H_{i\to *}$	Broadcast hash value used for the construction of dynamic pseudonyms in messages broadcast by node i to its neighbours.
$h_{i\to j}$	Implicit counter of messages sent by node i to node j. $h_{i\to j}$ is l_h bits long.
$s_{i\to j}$	Explicit counter of messages sent by node i to node j. $s_{i\to j}$ is l_s bits long.
$C_{i\to j}$	Unicast counter. This is the counter of messages sent by node i to node j. It consists of two parts: the implicit counter, $h_{i\to j}$, and the explicit counter, $s_{i\to *}$, (i.e., $C_{i\to j} = h_{i\to j} \parallel s_{i\to j}$).
$C_{i\to *}$	Broadcast counter of node i. It is the counter of messages broadcast by node i to its neighbours.
$C_{i\to\odot}$	Cluster broadcast counter of node i. It is the counter of messages broadcast by node i to other member nodes in the cluster.
$r_{i\to j}$	Random number generated by node i for the construction of dynamic pseudonyms in messages sent from node i to node j. The random number is l_r bits long.
$r_{i\to *}$	Random number generated by node i for the construction of dynamic pseudonyms in messages broadcast from node i to its neighbours.
MAC	Message authentication code.
S_T	Threshold value representing the number of lost messages that can be tolerated by the WSN.

TABLE II. THE PSEUDONYM TABLE (T_i) OF NODE i

	BS	CH	Other neighbours		
Fixed Pseudonym	$ID_B^{''}$	$ID_j^{''}$	$ID_u^{''}$	$ID_v^{''}$	…
Pairwise key (k)	k_{Bi}	k_{ij}	k_{iu}	k_{iv}	…
Broadcast key (k)	k_N	k_j^*	k_u^*	k_v^*	…
Cluster broadcast key (k)		$k_j^{\odot}$			
Outbound unicast index (ind)	$ind_{i\to B}$	$ind_{i\to j}$	$ind_{i\to u}$	$ind_{i\to v}$	…
Outbound unicast counter (C)	$C_{i\to B}$	$C_{i\to j}$	$C_{i\to u}$	$C_{i\to v}$	…
Inbound unicast index (ind)	$ind_{B\to i}$	$ind_{j\to i}$	$ind_{u\to i}$	$ind_{v\to i}$	…
Inbound unicast counter (C)	$C_{B\to i}$	$C_{j\to i}$	$C_{u\to i}$	$C_{v\to i}$	…
Broadcast index (ind)	$ind_{B\to *}$	$ind_{j\to *}$	$ind_{u\to *}$	$ind_{v\to *}$	…
Broadcast counter (C)	$C_{B\to *}$	$C_{j\to *}$	$C_{u\to *}$	$C_{v\to *}$	…
Cluster broadcast counter (C)		$C_{j\to\odot}$			

- MSE-2: The authenticity and integrity of a transmitted message should be ensured.
- MSE-3: A replay attack on a message should be detectable.

(d) Minimizing Overhead Costs:

- MOC-1: Minimize computational costs.
- MOC-2: Minimize communication overheads.
- MOC-3: Minimize memory requirements.

VI. THE ID-APS SCHEME

This section provides an overview of the design of ID-APS including different pseudonyms and protocols used.

A. Pseudonyms

As shown in Figure 1, ID-APS uses a hybrid combination of fixed (initial and operational) and dynamic (pairwise and broadcast) pseudonyms. These pseudonyms are discussed in more details below.

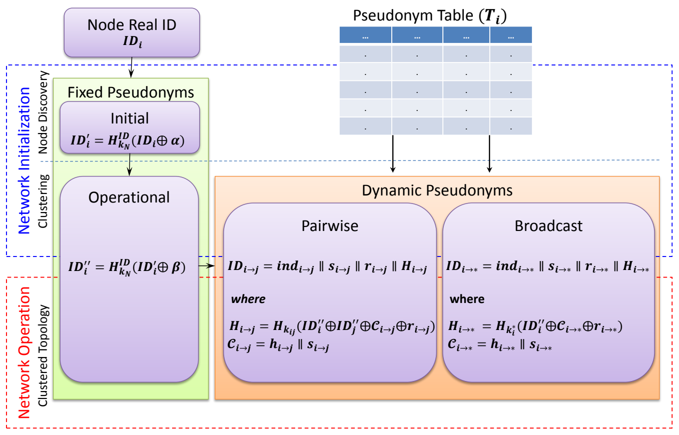
Node Real ID
ID_i
Pseudonym Table (T_i)
Network Initialization
Node Discovery
Clustering
Network Operation
Clustered Topology
Fixed Pseudonyms
Initial
$ID_i' = H_{k_N}^{ID}(ID_i \oplus \alpha)$
Operational
$ID_i'' = H_{k_N}^{ID}(ID_i' \oplus \beta)$
Dynamic Pseudonyms
Pairwise
$ID_{i\to j} = ind_{i\to j} \parallel s_{i\to j} \parallel r_{i\to j} \parallel H_{i\to j}$
where
$H_{i\to j} = H_{k_{ij}}(ID_i'' \oplus ID_j'' \oplus C_{i\to j} \oplus r_{i\to j})$
$C_{i\to j} = h_{i\to j} \parallel s_{i\to j}$
Broadcast
$ID_{i\to *} = ind_{i\to *} \parallel s_{i\to *} \parallel r_{i\to *} \parallel H_{i\to *}$
where
$H_{i\to *} = H_{k_i^*}(ID_i'' \oplus C_{i\to *} \oplus r_{i\to *})$
$C_{i\to *} = h_{i\to *} \parallel s_{i\to *}$

Figure 1. Pseudonym structure used in ID-APS.

(a) Fixed Pseudonyms

- *Initial pseudonym* $ID_i^{'}$: A unique identifier of the node used during the initial node discovery and key establishment phase. $ID_i^{'}$ is computed by the BS as $ID_i^{'} = H_{k_N}^{ID}(ID_i \oplus \alpha)$ and loaded into each node before its deployment.
- *Operational pseudonym* $ID_i^{''}$: A unique global identifier of the node used after the initial phase. It is used when the node should be identified by authorized entities. It is also used to generate dynamic pseudonyms for the node. The confidentiality of this pseudonym is always protected in transit. It is computed as $ID_i^{''} = H_{k_N}^{ID}(ID_i^{'} \oplus \beta)$.

(b) Dynamic Pseudonyms

- *Pairwise pseudonym* $ID_{i\to j}$: This is a dynamically generated pseudonym used as a per-message pseudonym identifying both node i (as a sender) and j (as a receiver) in each message flowing from node i to j. Similarly, node j will use $ID_{j\to i}$ for its unicast to node i. Only the intended recipient of the unicast can recognise this pseudonym. The pairwise pseudonyms are used in all unicast communications taking place after the initial phase. $ID_{i\to j}$ is constructed as $ID_{i\to j} = ind_{i\to j} \parallel s_{i\to j} \parallel r_{i\to j} \parallel H_{i\to j}$ where $H_{i\to j} = H_{k_{ij}}(ID_i^{''} \oplus ID_j^{''} \oplus C_{i\to j} \oplus r_{i\to j})$, $C_{i\to j} = h_{i\to j} \parallel s_{i\to j}$ and,

- $ind_{i \to j}$ is an outbound unicast index value shared between nodes i and j. It is also called the inbound unicast index of node j for messages from node i. This value is fixed for the life-time of node i. It is used by node j (the recipient) to verify the potential senders of a received message. This verification is designed to filter out any unauthentic messages as soon as they are received, i.e., before more accurate and expensive verifications need to be performed. A recipient only accepts a message if the index value contained in the message matches with one of those recorded in its pseudonym table. Node j will use another index $ind_{j \to i}$ to send unicasts to node i. These indexes are established as follows: assume that the length of $ind_{i \to j}$ is l_i bits, node i picks $ind_{i \to j}$ at random from the range of 2^{l_i} values and sends it to node j encrypted with the pairwise key. Node j does the same to establish $ind_{j \to i}$ with node i.
- $C_{i \to j}$ is used to generate sequence numbers to be carried in messages from node i to node j. It is called an outbound unicast counter by node i and inbound unicast counter by node j. It consists of two parts: an implicit counter $h_{i \to j}$ and an explicit counter $s_{i \to j}$, (i.e., $C_{i \to j} = h_{i \to j} \parallel s_{i \to j}$). The implicit counter $h_{i \to j}$ is not included in the pseudonym; rather it is one of the input items used in generating the keyed hash value $H_{i \to j}$. The explicit counter $s_{i \to j}$ is explicitly transmitted as part of the pseudonym. $s_{i \to j}$ increments with each message transmission while $h_{i \to j}$ increments each time $s_{i \to j}$ reaches its maximum value, at which point $s_{i \to j}$ is set to 0. $C_{i \to j}$ is maintained by both nodes in their pseudonym tables and its value is incremented by the sender after each message transmission and by the receiver after each message reception. Typically, the counters start from 0. However, to increase the anonymity level, each node establishes a random start to its counter value with the receiver. The purpose for using this counter is two-fold: (i) to ensure uniqueness of the pseudonyms carried in each message; (ii) to assure the freshness of a received messages (i.e., for detecting replayed messages).
- $r_{i \to j}$ is a random number generated by node i to break the link between the pseudonyms used in different messages sent from node i to j, i.e., to make it harder for an adversary to compute node pseudonyms used previously once the current pseudonym is compromised. $r_{i \to j}$ is regenerated before each message transmission and sent as part of the pseudonym but not stored in the pseudonym tables of the communicating nodes.
- $H_{i \to j}$ is an outbound hash value carried as part of the pseudonym in a unicast from node i to node j. It is used by the sending node to (i) further obfuscate the identities of the sender and receiver of the unicast; (ii) to ensure the authenticity of the pseudonym. Upon reception, this hash value will also be used by the recipient to verify the pseudonym.

To generate a pairwise pseudonym, the source node first retrieves k_{ij}, $ind_{i \to j}$ and $C_{i \to j}$ from its pseudonym table, picks a random number $r_{i \to j}$, computes $H_{i \to j}$ and constructs $ID_{i \to j}$. The node then increments the counter value to make it ready for the next message.

• *Broadcast pseudonym* $ID_{i \to *}$: This pseudonym is dynamically generated by node i for each message it broadcasts to its neighbours. $ID_{i \to *}$ can only be identified by nodes that have node i in their pseudonym tables and are located within the transmission range of node i. The generation of a broadcast pseudonym is similar to that of a pairwise pseudonym, except that a different set of attribute values are used based on whether the node is broadcasting within its own cluster or outside the cluster. If the node is broadcasting within the cluster, then the node uses its broadcast index $ind_{i \to *}$, cluster broadcast key $k_i^{\odot}$ and cluster broadcast counter $C_{i \to \odot}$ to construct $ID_{i \to *}$. The parameters, $ind_{i \to *}$, $k_i^{\odot}$ and $C_{i \to \odot}$ are shared among all the nodes in the cluster and maintained in each node's pseudonym table. $ID_{i \to *}$ is constructed as $ID_{i \to *} = ind_{i \to *} \parallel s_{i \to \odot} \parallel r_{i \to *} \parallel H_{i \to *}$ where $H_{i \to *} = H_{k_i^{\odot}}(ID_i^{''} \oplus C_{i \to \odot} \oplus r_{i \to *})$. However, if the node is broadcasting outside its cluster, then it uses its broadcast index $ind_{i \to *}$, broadcast key k_i^* and broadcast counter $C_{i \to *}$ to construct $ID_{i \to *}$ where k_i^* and $C_{i \to *}$ are shared among all the neighbouring nodes and maintained in each node's pseudonym table. $ID_{i \to *}$ is constructed as $ID_{i \to *} = ind_{i \to *} \parallel s_{i \to *} \parallel r_{i \to *} \parallel H_{i \to *}$ where $H_{i \to *} = H_{k_i^*}(ID_i^{''} \oplus C_{i \to *} \oplus r_{i \to *})$.

(c) Dynamic Pseudonym Verification (DP-Ver)

When a message is received by a node, say j, the node executes algorithm DP-Ver (Figure 2) to confirm the validity and authenticity of the pseudonym contained in the message. DP-Ver consists of three verifications, DP-Ver.1, DP-Ver.2 and DP-Ver.3. If the outcome of DP-Ver.1 is negative, node j will discard the message, otherwise it performs DP-Ver.2. If the outcome of DP-Ver.2 is positive, then the node confirms the validity of the received pseudonym, otherwise it performs DP-Ver.3. Similarly, if the outcome of DP-Ver.3 is positive then the node confirms the validity of the pseudonym, otherwise, it discards the message. The three verifications are detailed below.

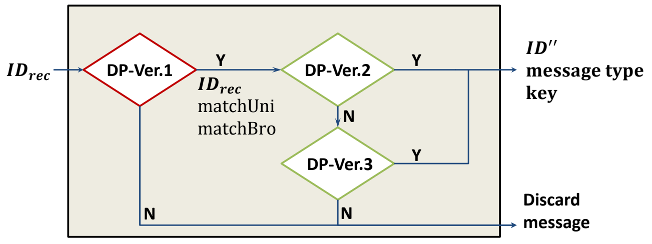

Figure 2. DP-Ver algorithm.

• *DP-Ver.1 - Index Value Verification*: Used to verify the received index value by searching for a match in the receiver's pseudonym table. DP-Ver.1 is a computationally cheap verification that is used to filter out unauthentic messages as soon as possible. Node j maintains a set of nodes with which node j is allowed to communicate, $\mathbb{A}_j$ where $\mathbb{A}_j \subset N_j$. Each node belongs to one cluster and is only allowed to communicate with the nodes (CH and other members) in the same cluster. Only the CH is allowed to communicate outside the cluster with other CHs. Therefore, when node j receives a message, it retrieves index ind_{rec} from the received message and checks it against the index values of the nodes belonging to $\mathbb{A}_j$. As node j does not know whether the received message is a unicast or broadcast, it has to check against the unicast and broadcast indexes of all nodes in $\mathbb{A}_j$. If matches are found within $\mathbb{A}_j$, then DP-Ver.1 returns two sets of potential senders, one for the unicasts ($matchUni$) and the other for broadcasts ($matchBro$). Otherwise, it returns empty sets, which means that node j is not the intended recipient of the message or the message is not authentic, in which case the node should discard the message.

- *DP-Ver.2 - Keyed hash value verification (no message loss)*: Used to verify the keyed hash value where there are no lost messages (i.e., the received explicit counter = the explicit counter maintained in the receiver's pseudonym table). If either of the two sets, $matchUni$ or $matchBro$ is not empty, then DP-Ver.2 is used to verify the received keyed hash value H_{rec}. To do this, the algorithm verifies the nodes in $matchUni$ first and if the result is negative it continue verifying the nodes in $matchBro$. For each $ID_i^{''} \in matchUni$, node j retrieves k_{ij} and $C_{i\to j}$ from T_j and checks if $s_{i\to j} = s_{rec}$. If they are equal, then node j computes $H^{'} = H_{k_{ij}}(ID_i^{''} \oplus ID_j^{''} \oplus C_{i\to j} \oplus r_{rec})$ and checks if $H^{'} = H_{rec}$. If they are equal, this means that node j is the intended recipient of the unicast message, the message is fresh (not replayed), and it is sent by node i. Node j then increments $C_{i\to j}$, updates T_j with this incremented value and terminates the algorithm returning the operational pseudonym of the sender $ID_i^{''}$, the type of the message (i.e., unicast) and the pairwise key. If the two keyed hash values are not equal, the execution of the algorithm continues by verifying the $ID^{''}$s in $matchBro$. The verification here is similar to that for $matchUni$. If this verification is negative, then node j proceeds to execute DP-Ver.3.
- *DP-Ver.3 - Keyed hash value verification (with message loss)*: Used to verify the keyed hash value where there are lost messages (i.e., the received explicit counter $\neq$ the explicit counter maintained in the receiver's pseudonym table). The maximum number of lost messages that can be tolerated equals to a threshold value, S_T. The algorithm starts with verifying the nodes belonging to $matchUni$. First, it retrieves k_{ij} and $C_{i\to j}$ from T_j for each $ID_i^{''} \in matchUni$, computes $S_D = |s_{i\to j} - s_{rec}|$ and checks if $S_D < S_T$. If true, then node j increases $C_{i\to j}$ based on S_D value, i.e., temporarily creates a new counter $C^{Temp} = C_{i\to j} + S_D$ and uses C^{Temp} to compute $H^{'} = H_{k_{ij}}(ID_i^{''} \oplus ID_j^{''} \oplus C^{Temp} \oplus r_{rec})$. Then, it checks if $H^{'} = H_{rec}$. If they are equal, node j confirms that it is the intended recipient of the unicast message. It increases $C_{i\to j}$ based on the computed number of lost messages and updates T_j with this value, thus, synchronising the counter value with the sender. Then, node j terminates the algorithm returning the operational pseudonym of the sender $ID^{''}$, the type of the message (i.e., unicast) and the pairwise key. If $H^{'} \neq H_{rec}$, then, node j continues verifying if the sender is one of the $ID^{''}$s in $matchBro$. If the outcome of this verification is negative, then node j discards the message.

B. Anonymous Communication Protocols

There are three communication protocols in ID-APS: anonymous one-hop unicast (AOHU), anonymous multi-hop unicast (AMHU) and anonymous local broadcast (ALBR).

(a) Anonymous one-hop unicast (AOHU)

This protocol consists of AOHU-Send and AOHU-Receive procedures. A node i sends data to its one-hop neighbour, node j, by executing AOHU-Send, for which node i generates a pairwise pseudonym $ID_{i\to j}$, encrypts the data using k_{ij}, generates a MAC of the pseudonym and the encrypted data, and constructs a message $m_{i\to j} = ID_{i\to j} \parallel Payload_{i\to j} \parallel MAC_{i\to j}$ where $Payload_{i\to j}$ is the encrypted data. Then, node i sends $m_{i\to j}$ to node j. Node j executes AOHU-Receive to receives $m_{i\to j}$, for which it verifies the pseudonym using DP-Ver to find the source of the message, verifies the MAC and decrypts the payload using k_{ij}.

(b) Anonymous multi-hop unicast (AMHU)

This protocol is built on AOHU. It consists of AMHU-Send, AMHU-Forward and AMHU-Receive procedures. When node i is to send data using a unicast to the BS multi-hops away (e.g., through an intermediate node j), node i executes AMHU-Send for which it generates an end-to-end pairwise pseudonym $ID_{i\to B}$, encrypts the data using k_{Bi} and generates a MAC of the pseudonym and the encrypted data. Next, node i constructs a message $m_{i\to B} = ID_{i\to B} \parallel Payload_{i\to B} \parallel MAC_{i\to B}$. As the BS is not a direct neighbour of node i, node i checks its routing table for the next hop (i.e., node j) in the route to the BS. Next, node i uses the AOHU-Send procedure to send a unicast, which wraps $m_{i\to B}$ to the one-hop neighbour j. Upon receiving the one-hop unicast message from node i, node j executes AMHU-Forward to forward node i's message to the BS. First, it retrieves $m_{i\to B}$ using the AOHU-Receive procedure. Then, it checks its routing table for the next hop (i.e., the BS). After that, it uses the AOHU-Send procedure to forward node i's message to the BS. When the BS receives the one-hop unicast from node j, the BS executes AMHU-Receive procedure to retrieve the data sent by node i. This is done by executing AOHI-Receive first to retrieve $m_{i\to B}$. After that, the BS verifies $ID_{i\to B}$ using DP-Ver to find the original source of the message. Then, it verifies the received MAC and decrypts the payload using k_{Bi} to retrieve the data sent by node i.

(c) Anonymous local broadcast (ALBR)

This protocol consists of ALBR-Send and ALBR-Receive procedures. When node i is to send a broadcast to its one-hop neighbours, it executes the ALBR-Send procedure. In this procedure, node i first reads the respective broadcast key from its memory, depending on whether it is broadcasting within the cluster or outside it. Then, node i generates the broadcast pseudonym $ID_{i\to *}$, encrypts the data and generates a MAC of the pseudonym and the encrypted data. Next, node i broadcasts the message $m_{i\to *} = ID_{i\to *} \parallel Payload_{i\to *} \parallel MAC_{i\to *}$ to the neighbours. Upon receiving the broadcast by a neighbouring node j, node j executes ALBR-Receive for which it verifies the pseudonym using DP-Ver to find the source of the broadcast, verifies the MAC and decrypts the payload to retrieve the data sent by node i.

C. ID-APS in Action

There are three phases in ID-APS: sensor node initialization, network initialization and network operation.

(a) Sensor Node Initialization

Prior to deployment of the network, each sensor node is initialized with specific parameter values and functions. For each node $i \in \mathbb{N}$, the BS generates the following parameter values: ID_i, $ID_i^{'}$, β, k_N, k_{Bi}, $ind_{B\to i}$, $ind_{i\to B}$, $ind_{B\to *}$, $C_{B\to i}$, $C_{i\to B}$ and $C_{B\to *}$. The indexes and starting counter values are picked at random for each node. The installed functions are two keyed hash functions, $H_K^{ID}(.), H_k(.)$ and a pseudorandom function.

(b) Network Initialization

In this phase, nodes discover their neighbours, establish pairwise and broadcast keys and perform clustering, preparing for normal network operation. In the node discovery and key establishment phase, the nodes use EDDK methods to discover their neighbours, and establish broadcast and pairwise keys. In this process, the nodes identify their neighbours using the initial fixed pseudonyms and populates a pseudonym table that includes the initial fixed pseudonym and the established keys for each neighbour. The nodes also pick the indexes $(ind_{i\to *}, ind_{i\to j})$ and starting counter values $(C_{i\to *}, C_{i\to j})$ at random and exchange these values with their neighbours. Then, each node generates an operational pseudonym for itself and for each neighbour in the node's pseudonym table and deletes the initial pseudonyms. By the end of this process, each node should have updated its pseudonym table and be ready for the clustering process to be performed. During clustering, nodes organise themselves into clusters and elect CHs. To ensure that the clustering process is done in a secure and anonymous way, we utilise the Private Cluster Head Election (PCHE) [9] protocol. This protocol is designed to hide the identities of the CHs from an adversary who can monitor the clustering process. To further increase the protection level of the ID anonymity, we use our ALBR protocol for all broadcast messages involved in the PCHE. Upon completion of clustering, each node should know if it is a CH or a cluster member. After that, the elected CH in each cluster generates a cluster broadcast key and picks a cluster broadcast counter at random and unicasts both values to each member in its cluster. It is worth noting that the elected CHs will also need to establish their routes to the BS. ID-APS is independent of the routing mechanism; however, ID-APS requires that each CH uses the dynamic pseudonyms when sending the route request and route update messages. Once clustering is complete and the routes to the BS established, the nodes are ready to perform normal network operation.

(c) Network Operation

In this phase, the nodes start to carry out their normal functional operations such as data collection and aggregation. Cluster member nodes can only communicate with the CH and other cluster members (intra-cluster communication). Communication outside the cluster (inter-cluster communication) is carried out via the CH.

In intra-cluster communication, each node senses the physical phenomenon at a pre-defined sampling period and sends its reading to the CH. The CH waits for the readings to be received from different cluster members before applying an aggregation function to compute an aggregate value for the readings. Intra-cluster communication can either be unicast or broadcast among cluster members as follows:

- Intra-cluster unicast is carried out using the AOHU protocol and the pairwise keys to secure the communication.
- Intra-cluster broadcast is carried out using the ALBR protocol and the cluster broadcast key to secure the communication.

In inter-cluster communication, each CH forwards the locally aggregated data within its cluster to the BS through other CHs. CHs which are closer to the BS may collect the aggregated data from other CHs that are further away and aggregate the received data before sending them to the BS. The inter-cluster communication encompasses the following:

- Inter-cluster unicast between a CH and the BS is carried out using the AOHU protocol if the CH is a direct neighbour of the BS or the AMHU protocol if the CH is multi-hops away from the BS. In both protocols the pairwise keys are used to secure the communication.
- Inter-cluster unicast among neighboring CHs is carried out using the AOHU protocol and the pairwise keys to secure the communication.
- Inter-cluster broadcast from the BS is done through the authenticated broadcast protocol μTesla [8].

VII. Analyses and Evaluation

The effectiveness and efficiency of ID-APS are analysed based on the parameter values presented in Table III.

TABLE III. ID-APS EVALUATION PARAMETER VALUES

Sensor mote	CrossBow TelosB [10]
Block cipher	RC5 [11] (CBC mode)
Message authentication	CBC-MAC-RC5
Key length l_k	8 bytes
ID length l_{ID}	8 bytes
Index length l_i	1 byte
Counter length $(l_h + l_s)$	(3+1) bytes
Pseudonym random number length l_r	2 bytes
Pseudonym Hash value length l_H	4 bytes

A. ID Anonymity

NIP-1: As described earlier, a single pairwise pseudonym $ID_{i\to j} = ind_{i\to j} \parallel s_{i\to j} \parallel r_{i\to j} \parallel H_{i\to j}$ is used to identify both the sender (node i) and the receiver (node j) of a unicast message, where $H_{i\to j} = H_{k_{ij}}(ID_i^{''} \oplus ID_j^{''} \oplus C_{i\to j} \oplus r_{i\to j})$. It is hard for an adversary to gain any identity information regarding the source or destination of the message from the hash value. This is because (i) it is computationally hard to reverse the hash value due to the one-way property of the hash function and (ii) to compute a matching hash value, the adversary needs to know the pairwise key that was used in the computation, which is only known by the message sender and receiver, and in addition, the adversary needs to guess the correct values of $ID_i^{''}, ID_j^{''}$ and $h_{i\to j}$ to succeed. The probability of guessing the correct values of all these parameters is: $(1/2^{l_K+2l_{ID}+l_h} = 2^{-216})$, which is sufficiently low if the values of l_K, l_{ID} and l_h are sufficiently large.

NIP-2: The analysis of NIP-2 follows that of NIP-1. The hash value in a broadcast is computed as $H_{i\to *} = H_{k_i^*}(ID_i^{''} \oplus C_{i\to *} \oplus r_{i\to *})$. The probability of guessing the correct values of $k_i^*, ID_i^{''}$ and $h_{i\to *}$ is $(1/2^{l_K+l_ID+l_h} = 2^{-158})$.

NIP-3: For a neighbourhood of n nodes where an active adversary is present, we analyse the impact on ID anonymity when the adversary compromises m nodes where $m \leq n-3$. Let us consider the worst case scenario when the adversary compromises all the nodes except three nodes a, b and c (i.e., there are $n-3$ colluding nodes). The adversary may learn $ID_a^{''}, ID_b^{''}$ and $ID_c^{''}$ assuming that each of the non-compromised nodes lies in the transmission range of at least one compromised node. When node b for example sends a unicast to node c, node b constructs the dynamic pseudonym as $ID_{b\to c} = ind_{b\to c} \parallel s_{b\to c} \parallel r_{b\to c} \parallel H_{b\to c}$ where, $H_{b\to c} = H_{k_{bc}}(ID_b^{''} \oplus ID_c^{''} \oplus C_{b\to c} \oplus r_{b\to c})$. The additional information available to the adversary through overhearing the message is $s_{b\to c}$ and $r_{b\to c}$. However, the adversary does not know k_{bc} and $h_{b\to c}$ which are only known to nodes b and c. Therefore, from the adversary's point of view, each of the three non-compromised nodes has equal probability to be the sender or receiver of the message. To learn the exact sender or receiver of the message, the adversary needs to guess the values of k_{bc} and $h_{b\to c}$ and apply the keyed hash function to get the correct $H_{b\to c}$. The probability to get the correct hash value is $1/2^{l_K+l_h} = 2^{-88}$.

NIP-4: ID-APS uses a random number, r, to break the link between different pseudonyms used by the same node for the same destination. If a node i is compromised, the adversary may obtain all the necessary items (i.e., crypto keys, indexes, operational fixed pseudonyms and counters) to compute node i's pseudonyms. Let us denote the current pseudonym of node i in its unicast communication to node j as $ID^m_{i\to j}$. If $ID^m_{i\to j}$ is compromised, then to learn the pseudonym used in the previous unicast message to node j, i.e., $ID^{m-1}_{i\to j}$, the adversary has to decrement the current counter value and guess the random number used in the previous message. As the length of $r_{i\to j}$ is l_r bits, the probability of guessing the correct random number used by the node in the previous message is $1/2^{l_r} = 2^{-16}$. If higher anonymity is required, the length of $r_{i\to j}$ can be set to a higher value.

B. *Degree of anonymity vs. computational costs analysis*

The anonymity level of ID-APS is analysed and quantified based on the concept of degree of anonymity [12]. The entropy of a system is calculated after the attack and compared with the maximum entropy of the system. This provides a measure of information gained by the adversary after the attack. Assuming a WSN with a total number of N nodes distributed uniformly in a field, if an eavesdropping adversary is present while a message is being sent, then the only fixed part of the pseudonym that the adversary can identify a sender by is the index value. However, this index value may also be used by several other nodes while sending messages to their neighbours. As the index values are picked at random, the probability of a node i picking a certain index (ind_a) to identify itself as a sender to one of its neighbours is $1/2^{l_i}$. Let the average size of $|N_i| = n$, then the node will pick n indexes at random to identify itself in unicast communications and another index for broadcast communications. As broadcasts and unicasts messages have similar structure, the total probability, P_{ind_a}, of a node using index ind_a to identify itself in any message sent by the node is given by (1).

$$P_{ind_a} = 1 - \left(\frac{2^{l_i}-1}{2^{l_i}}\right)^{n+1} \tag{1}$$

This also implies that approximately NP_{ind_a} nodes in the network will be using ind_a to identify themselves when communicating with other nodes (assuming large network). Through continuous monitoring of the index values appended to messages exchanged in the network, the adversary may assign certain probability to each node as being the sender of an eavesdropped message. Let X be the discrete random variable with probability mass function $P_i = P(X = i)$, where i represents a single sender in a set of potential senders, N. Let us assume that, in the worst case scenario, the adversary is able to identify all the nodes that are using ind_a as identifier. So, the adversary assigns a probability of $(1/NP_{ind_a})$ to NP_{ind_a} nodes as being the senders of a message that has an index value $= ind_a$ and a probability of 0 to the rest of the nodes in the network. The probability P_i is given by (2).

$$P_i = \begin{cases} \frac{1}{NP_{ind_a}} & \text{for } NP_{ind_a} \text{ nodes} \\ 0 & \text{for } N\left(1 - P_{ind_a}\right) \text{ nodes} \end{cases} \tag{2}$$

From [12], the entropy of the system after the attack is computed as $E(X) = -\sum_{i=1}^{N} P_i log_2(P_i)$, hence:

$$E(X) = log_2\left(\left(1 - \left(\frac{2^{l_i}-1}{2^{l_i}}\right)^{n+1}\right)N\right) \tag{3}$$

The maximum entropy of the system E_M is achieved when the adversary assigns a probability $P_i = 1/N$ to each node in the network as being the sender of a message, i.e., $E_M = log_2(N)$. Hence, the degree of anonymity d is estimated as:

$$d = \frac{E(X)}{E_M} = \frac{log_2\left(\left(1 - \left(\frac{2^{l_i}-1}{2^{l_i}}\right)^{n+1}\right)N\right)}{log_2(N)} \tag{4}$$

We see that the degree of anonymity depends on three parameters N, n and l_i. As N and n are fixed values, then the only parameter that can affect the anonymity of the nodes is l_i. Figure 3 shows the impact of l_i for a WSN with $N = 1000$ and $n = 100$. It is clear from the figure that as l_i increases, the degree of anonymity decreases. According to [12], to achieve an acceptable level of anonymity, the degree of anonymity should be greater than 0.8. Therefore, l_i should be selected as (≤ 8 bits) in the above example.

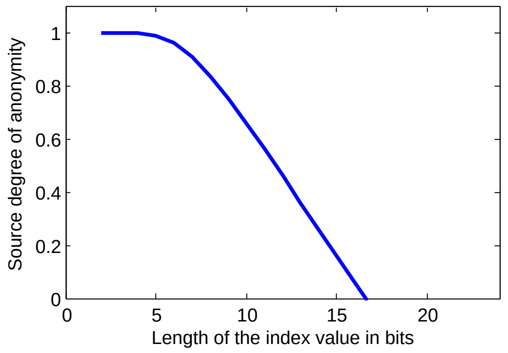

Figure 3. The impact of the length of the index value on the source degree of anonymity for N=1000 and n=100.

Let us now analyse the computational costs of receiving a message. Depending on the number of matches of the received pseudonym with those contained in a receiver's pseudonym table, the receiver may need to execute more than one keyed hash operation before successfully confirming the validity of the pseudonym. Suppose that each node i has an average size of $|\mathbb{A}_i| = m$. The value of m is directly proportional to the size of the cluster. The pseudonym table of node i, T_i, includes an inbound unicast index and a broadcast index for each node in T_i. Therefore, when node i receives a message, it has to find a match within $(2m)$ indexes in T_i. As described earlier, the inbound unicast indexes and broadcast indexes are picked at random during the node discovery and key establishment phase. Based on this, there is a probability that more than one index in T_i could match the index of the received dynamic pseudonym. The probability of getting exactly r indexes in T_i that matches the index of a received pseudonym (ind_{rec}) is given by (5).

$$P_{match}(r) = \binom{2m}{r}(P_{ind})^r(1-P_{ind})^{2m-r} \qquad (5)$$

Figure 4 shows the probability $P_{match}(r)$ against different r values for $m = 100$ nodes and $l_i = 8$ bits. It can be seen from the figure that the probability beyond 3 matches is very low. In other words, in 99% of the cases, the pseudonym matches occurs in less than 3 matches. Therefore, in 99% of the cases, the verification of a pseudonym involves less than 3 keyed hash operations.

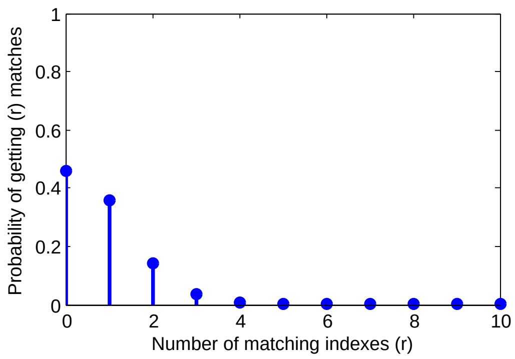

Figure 4. The probability of getting exactly r indexes in T_i that matches ind_{rec} for $m = 100$ and $l_i = 8$.

This computational cost may be further reduced by tuning the parameter values in (5) as follows:

- Reduce the size of the cluster and therefore the value of m drops: This will result in a reduction in the list of potential indexes. Figure 5 depicts the three cases where $m = 100, 50$ and 10 nodes, respectively. As can be seen, reducing the cluster size from 100 to 10 nodes reduces the likelihood of matching indexes from 3 to 1 in 99% of the cases.
- Increase the size of the index: Figure 6 shows the probability of getting r matches when l_i is set to 8, 10 and 12, respectively, where $m = 100$. As shown in the figure, the probability $P_{match}(r)$ drops when l_i increases from 8 to 12 bits. For $l_i = 12$, the receiver need to compute less than 2 hash values in 99% of the time.

From the above we can see that by reducing the cluster size and/or increasing the bit length of the index, we may reduce the computational costs incurred as a result of anonymity protection. However, increasing the bit length of the index reduces the degree of anonymity as illustrated by (4). Therefore, for an acceptable level of anonymity, we may cut computational costs by reducing the size of a cluster.

C. Performance Evaluation

In this section, the performance of ID-APS is evaluated in terms of computation, communication and memory consumption and compared to the performance of CAS and EAC.

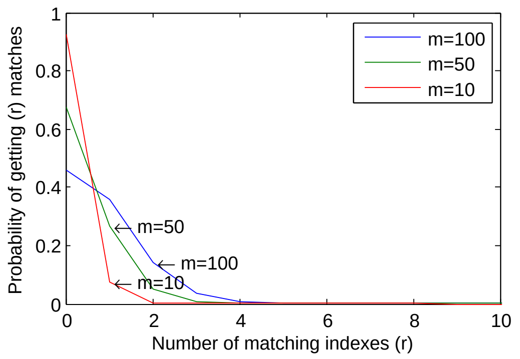

Figure 5. The probability of getting exactly r indexes in T_i that matches ind_{rec} for $l_i = 8$ and different m values.

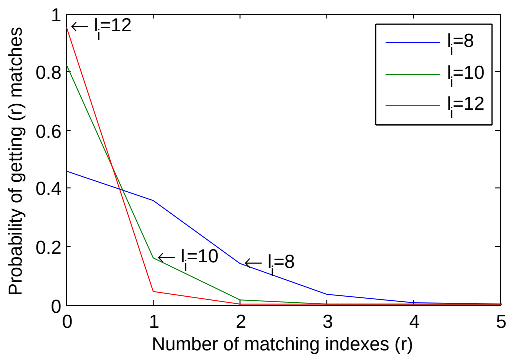

Figure 6. The probability of getting exactly r indexes in T_i that matches ind_{rec} for different l_i values and $m = 100$.

MOC-1: In ID-APS, each message construction prior to transmission involves the following computational costs: one random number generation, one dynamic pseudonym generation using keyed hash function, one payload encryption and one MAC generation protecting the pseudonym and the encrypted payload using a keyed hash function. Based on the experiments with RC5 execution time for TelosB motes [13], constructing a message in ID-APS with a maximum payload size of 29 bytes, as suggested by TinyOS [14], takes less than 20 ms. According to [8], sensor nodes support a maximum throughput of twenty 30-bytes messages/s with the microcontroller being idle for about 50% of the time. Therefore, our scheme is able to anonymize the node IDs, encrypt and generate a MAC for every message sent by the sensor node. The computational costs of receiving the message at the receiver side are comparable to that of the sender. Table IV shows a comparison of computational costs with CAS and EAC.

TABLE IV. COMPUTATIONAL COSTS COMPARISON

Scheme	Sender	Receiver	Other neighbours
CAS	One keyed hashing operation	One keyed hashing operation	One keyed hashing operation
EAC	Two hashing operations	One hash operation	-
ID-APS	One random number generation and one keyed hashing operation	One keyed hashing operation	-

As can be seen from Table IV, the computational costs of ID-APS are comparable to that of EAC and better than CAS.

This is because with the optimization of l_i and cluster size in ID-APS, there are negligible computational costs involved by other nodes in the neighbourhood.

MOC-2: A pseudonym in ID-APS is 8 bytes in size and it is used to identify both the sender and the receiver of a message. Table V shows the comparison between the three schemes in terms of message overheads. It can be seen that ID-APS exhibits the least communication costs in comparison with the CAS and EAC schemes. EAC has the highest message overheads as the receiver in EAC is required to send an acknowledgment to the sender to synchronise the pseudonym parameters.

TABLE V. MESSAGE OVERHEADS COMPARISON

Scheme	Sender	Receiver	Total
CAS	18-byte (ID+Index+Seq)	-	18-byte
EAC	32-byte (2 IDs)	20-byte (Ack)	52-byte
ID-APS	8-byte (ID)	-	8-byte

MOC-3: The memory requirements for the parameters used in ID-APS are shown in Table VI in comparison with CAS and EAC. In this table, n denotes the size of the node neighbourhood. Figure 7 shows a comparison of memory consumption between the three schemes for different n sizes.

TABLE VI. MEMORY CONSUMPTION COMPARISON

Scheme	Parameters	Memory consumption (byte)
CAS	Key size $K = 8$ bytes Other parameters size $P = 8$ bytes Index size $I = 2$ bytes Sequence number size $S = 8$ bytes	$2K + 3P + S + n(2K + 2I + 3P + 2S)$
EAC	Pseudonym size $ID = 16$ bytes Key size $K = 16$ bytes Other parameters size $P = 16$ bytes Link direction size $L = 0.25$ byte	$3ID + 2K + 2P + n(3ID + 2K + 2P + L)$
ID-APS	ID/pseudonym size $ID = 8$ bytes Key size $K = 8$ bytes Index size $I = 1$ byte Counter size $C = 4$ bytes	$3ID + 4K + 4I + 5C + n(ID + 3K + 3I + 4C)$

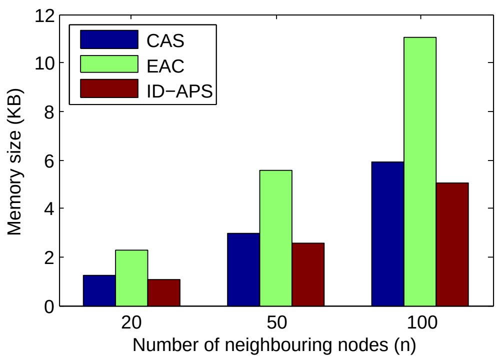

Figure 7. Comparison of memory consumption for different n values.

There are two observations from Figure 7. The first is that the memory requirements of ID-APS can be easily accommodated in the sensor memory. This is because TelosB mote has an architecture where the entire memory (48KB of flash memory, 10KB of RAM and 1MB of external EEPROM) is accessible for code and data. The second observation is that ID-APS consumes least memory space compared to the other two schemes.

VIII. CONCLUSION AND FUTURE WORK

This paper has presented the design and evaluation of a novel ID anonymity scheme, ID-APS. The scheme is designed to achieve a high level of ID anonymity with as less overhead costs as possible in terms of computation, communication and memory space. ID-APS provides ID anonymity in unicast and broadcast communication against passive adversaries and against attacks on past used pseudonyms by active adversaries. Theoretical evaluation of the anonymity level provided by the scheme has been carried out using the concept of the degree of anonymity, and the scheme has also been compared with two most relevant schemes in literature. The evaluation and comparison results have shown that ID-APS achieves a good level of ID anonymity protection with least level of overhead costs in comparison with related solutions. However, with ID-APS, if a node is compromised, the adversary may be able to impersonate the node and compromise the privacy of other nodes. Our future work will examine this issue in depth.

REFERENCES

[1] X. Luo, X. Ji, and M.-S. Park, “Location privacy against traffic analysis attacks in wireless sensor networks,” in Information Science and Applications (ICISA), 2010 International Conference on. IEEE, 2010, pp. 1–6.

[2] S. Misra and G. Xue, “Efficient anonymity schemes for clustered wireless sensor networks,” International Journal of Sensor Networks, vol. 1, no. 1, 2006, pp. 50–63.

[3] Y. Ouyang et al., “Providing anonymity in wireless sensor networks,” in Pervasive Services, IEEE International Conference on. IEEE, 2007, pp. 145–148.

[4] J.-P. Sheu, J.-R. Jiang, and C. Tu, “Anonymous path routing in wireless sensor networks,” in Communications, 2008. ICC’08. IEEE International Conference on. IEEE, 2008, pp. 2728–2734.

[5] J. Chen, X. Du, and B. Fang, “An efficient anonymous communication protocol for wireless sensor networks,” Wireless Communications and Mobile Computing, vol. 12, no. 14, 2012, pp. 1302–1312.

[6] M. Gouda, Y.-r. Choi, and A. Arora, “Antireplay protocols for sensor networks,” Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks,(ed. Jie Wu), CRC, 2005.

[7] X. Zhang, J. He, and Q. Wei, “Eddk: energy-efficient distributed deterministic key management for wireless sensor networks,” EURASIP Journal on Wireless Communications and Networking, vol. 2011, 2011, p. 12.

[8] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. E. Culler, “Spins: Security protocols for sensor networks,” Wireless networks, vol. 8, no. 5, 2002, pp. 521–534.

[9] L. Buttyán and T. Holczer, “Private cluster head election in wireless sensor networks,” in Mobile Adhoc and Sensor Systems, 2009. MASS’09. IEEE 6th International Conference on. IEEE, 2009, pp. 1048–1053.

[10] Crossbow, “Telosb datasheet,” April 2014. [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[11] R. L. Rivest, “The rc5 encryption algorithm,” in Fast Software Encryption. Springer, 1995, pp. 86–96.

[12] C. Diaz, S. Seys, J. Claessens, and B. Preneel, “Towards measuring anonymity,” in Privacy Enhancing Technologies. Springer, 2003, pp. 54–68.

[13] D. He et al., “An enhanced public key infrastructure to secure smart grid wireless communication networks,” IEEE NETWORK, vol. 28, no. 1, 2014, pp. 10–16.

[14] P. Levis et al., “Tinyos: An operating system for sensor networks,” in Ambient intelligence. Springer, 2005, pp. 115–148.