Efficient anonymity schemes for clustered wireless sensor networks

Satyajayant Misra and Guoliang Xue*

Department of Computer Science and Engineering,
Arizona State University,
Tempe, Arizona, USA
E-mail: satyajayant@asu.edu E-mail: xue@asu.edu
*Corresponding author

Abstract: In this paper, we propose two simple and efficient schemes for establishing anonymity in Clustered Wireless Sensor Networks (CWSNs). The first scheme Simple Anonymity Scheme (SAS), uses a range of pseudonyms as identifiers for a node to ensure concealment of its true identifier (ID). After deployment, neighbouring nodes share their individual pseudonyms and use them for anonymous communication. The second scheme Cryptographic Anonymity Scheme (CAS), uses a Keyed Cryptographic one way Hash Function (KCHF) for ID concealment. In CAS, neighbouring nodes securely share the information used by the KCHF to generate pseudonyms. Even when many nodes in a neighbourhood are compromised and are colluding, our schemes guarantee complete anonymity to non-compromised nodes during their mutual communication. Our schemes have reasonably low memory and computation costs. They can be embedded into any wireless sensor setwork routing protocol to ensure anonymity and privacy during node discovery, routing and data delivery.

Keywords: Wireless Sensor Networks (WSNs); clustering; authentication; confidentiality; anonymity; privacy; pseudonym.

Reference to this paper should be made as follows: Misra, S. and Xue, G. (2006) 'Efficient anonymity schemes for clustered wireless sensor networks', *Int. J. Sensor Networks*, Vol. 1, Nos. 1/2, pp.50–63.

Biographical notes: Satyajayant Misra is a PhD student in the Department of Computer Science and Engineering at Arizona State University (ASU), Tempe, Arizona, USA. He received an Integrated MSc (Hons.) in Physics and an MSc (Tech.) in Information Systems from Birla Institute of Technology and Science (BITS), Pilani, India in June 2003. His research interests include identifying security, privacy and reliability issues in wireless networks and formulating efficient solutions to handle them.

Guoliang Xue received a PhD in Computer Science from the University of Minnesota in 1991 and is currently a Professor in the Department of Computer Science and Engineering at Arizona State University. His research interests include efficient algorithms for optimization problems in networking. He has published over 140 papers in these areas. His research has been continuously supported by federal agencies including NSF and ARO. He is the recipient of an NSF Research Initiation Award in 1994 and an NSF-ITR Award in 2003. He is an Associate Editor of Computer Networks and an Associate Editor of the IEEE Network Magazine.

1 Introduction

Large scale distributed Wireless Sensor Networks (WSNs) are becoming increasingly common in a variety of applications, ranging from military to civilian in nature (Akyildiz et al., 2002). Despite significant improvements in the robustness of the Sensor Nodes (SNs), they are still hugely constrained, having limited power, memory, and computing abilities (Karlof and Wagner, 2003). The available redundancy and inherent energy scarcity of a sensor network encourages the use of aggregation of data while on its way to the base station from the point of stimuli.

Clustering of the WSNs vastly improves this aggregation ability. In a Clustered Wireless Sensor Network (CWSN), the nodes in a neighbourhood organise themselves into a cluster, with one node designated as the Cluster Head (CH) (Bandyopadhyay and Coyle, 2003; Younis and Fahmy, 2004). The CH gets information regarding a stimulus from the SNs in its neighbourhood and uses an aggregation scheme to aggregate this information. It then sends the information to a neighbouring CH in the direction of the Base Station (BS). This neighbouring CH may aggregate the information further and send it ahead.

In many applications of the WSN, identity of the nodes sending data from a locality to the BS might be extremely sensitive information. In a mobile wireless ad hoc network, identity of a node is generally given by a tuple, {location, identifier, time}(Delakouridis et al., 2005). In a static WSN, in general, the identifier (ID) is sufficient for unique node identification. An intelligent adversary

analysing the traffic in the network may obtain access to this identity information. Such an informed adversary can infer and destroy/compromise the identified SNs, rendering the network ineffective. The problem of traffic analysis becomes even more critical in a CWSN. Identity information of the CH in a region can allow an adversary to compromise the CH, effectively compromising the complete cluster and ensuring that the BS gets no information from the cluster's locality. To solve the problem of traffic analysis in the CWSN, design and deployment of effective anonymity solutions is essential. Anonymity solutions allow the SNs to use dynamic pseudonyms during the communication, thus reducing the scope of traffic analysis significantly.

In this paper, we propose two anonymity solutions for a CWSN, where the SNs in a neighbourhood share pairwise symmetric keys, generated using exchange of predeployed information (Blom, 1985). We assume that clustering is done in the network using the clustering scheme proposed by Younis and Fahmy (2004). We also assume that the Tiny OS beaconing scheme (Karlof and Wagner, 2003) is used for creating inter-cluster routes for communication of a CH with the BS. The first scheme is a simple yet effective scheme that provides the SNs with dynamic pseudonyms for use instead of their true identity during communication, ensuring complete anonymity. The SNs are given pseudonym ranges that are non-contiguous and chosen uniformly at random from a pseudonym space. Our scheme guarantees complete anonymity to a communicating SN even when several of its neighbours are compromised and are colluding.

The second scheme has better memory efficiency and uses keyed hash functions to generate anonymous pseudonyms for SNs to use instead of their true identity during communication. The SNs in a neighbourhood securely share parameters immediately after deployment and use these parameters to key a publicly known cryptographic hash function that generates the anonymous IDs. Keyed cryptographic hash functions, also known as strongly collision resistant hash functions (Menezes et al., 1996), are popularly used in the creation of Message Authentication Codes (*MACs*) in WSNs. The key used for keying the hash function is a secret, shared between the SNs that are involved in the communication. It is difficult for an adversary that is not a part of the communication to identify the key and hence be able to forge the message. In our scheme, we further strengthen the output from the hash function against forgery (as compared to MACs) by using an input with a secret component. Hence, our scheme guarantees complete anonymity to the sender and receiver. Further, the scheme ensures that the probability of an adversary being able to identify the true ID of the sender and receiver, or forge an authentic sender-receiver ID pair for a communication in its neighbourhood, is very low.

The rest of this paper is organised as follows. In Section 2, we briefly survey related work in the areas of clustering, anonymity and privacy in wireless sensor and ad hoc networks. In Section 3, we give the problem statement and define the models, security assumptions and requirements for anonymity in a CWSN. In Section 4, we describe the framework for our proposed anonymity schemes. In Section 5, we propose our first scheme, SAS, and analyse the protocol. Section 6 contains our proposition of the second scheme, CAS and accompanying analyses of the protocol. In Section 7, we compare our two anonymity schemes on the basis of their performance in terms of memory and computation requirements. In Section 8, we present our conclusions and scope of future work.

2 Related work

Ibriq and Mahgoub (2004) specified the design criteria and challenges for cluster based WSNs. An on-demand distributed clustering algorithm for ad hoc networks was proposed by Chatterjee et al. (2004). A hybrid, energy efficient and distributed clustering protocol (HEED), was proposed by Younis and Fahmy (2004), which does not depend on the network topology or size, nor makes any assumptions on the node degree. Bandyopadhyay and Coyle (2003) proposed a distributed and randomised clustering algorithm that generates a hierarchy of CHs.

Security in WSNs has been a topic of intensive study in the last few years. Karlof and Wagner (2003) considered routing security in WSNs. They identified attacks on WSNs and proposed countermeasures and security design considerations. Zhu et al. (2003) proposed a key management protocol for WSNs that supports in-network processing. The protocol also restricts the impact of node compromise to the local neighbourhood. Liu and Ning (2003) presented a general framework for establishing pairwise keys between sensors, on the basis of a polynomial-based key pre-distribution protocol. Du et al. (2005) proposed a novel secret key predistribution scheme that substantially improves resilience of the network.

Anonymity and security in clustered wireless sensor networks have not been studied in great detail, although there have been significant work in ad hoc networks that may be applied to these sensor networks. Poosarla et al. (2004) used both public and symmetric key cryptography to provide security in the cluster based routing protocol for ad hoc networks. Delakourdis et al. (2005) presented a novel architecture that provides location anonymity to a mobile node by splitting the identification information among the entities in the network. Zhu et al. (2003) proposed the ASR protocol that provides a form of identity anonymity and location privacy. Wu and Bhargava (2005) proposed an on-demand position based private routing protocol for an ad hoc network. Kong and Hong (2003) proposed an anonymous on-demand routing protocol for mobile ad hoc networks deployed in hostile environments. To our best knowledge, no research on anonymity in wireless networks has addressed the problems we are studying in this paper.

Message authentication codes have been used as a common mechanism for achieving message authentication in all kinds of communication networks. Many techniques such as CBC-MAC, universal hash functions, hashed MAC (HMAC), XOR MAC (XOR-MAC) and NMAC have been proposed for generation of MACs. Bellare et al. (1996a,b) proposed the NMAC and HMAC schemes for MAC. Both the schemes are proved to be secure if the underlying hash function used has a good cryptographic strength. Bellare et al. (1994) showed that the CBC-MAC construction is secure if the underlying block cipher on which it is based

is secure. Preneel and Oorschot van (1995) proposed a generic construction (MDx-MAC) for transforming any secure hash function of the MD4-family into a secure MAC. Bellare et al. (1995) proposed the XOR-MAC scheme that has the desirable properties of parallelisability and incrementability while also being provably secure. Black et al. (1999) presented a secure MAC algorithm named UMAC that was an order of magnitude faster than other MAC algorithms by virtue of being highly parallelisable. The second anonymity scheme that we propose uses a chosen keyed cryptographic hash function, which may be similar to one of the MAC generation schemes to create anonymous IDs for communicating nodes in the neighbourhood.

3 Problem statement

3.1 System model

We consider a wireless sensor network composed of a large number of similar, small, low cost, and immobile sensors. These sensors are assumed to have unique IDs. They have limited power, memory and computation abilities, and are not tamper resistant. The network is partitioned into clusters. The links in the network are assumed to be bidirectional. The SNs send sensed data to the elected CH. The role of a CH rotates between the SNs in a neighbourhood. We assume the neighbourhood of an SN consists of all other SNs within its transmission range and also includes itself.

There are many clustering algorithms that have been proposed in the literature (Bandyopadhyay and Coyle, 2003; Chatterjee et al., 2002; Younis and Fahmy, 2004). We assume that the network uses some clustering mechanism. We propose schemes for establishing anonymity in a given CWSN. Our anonymity schemes can work on top of any clustering scheme. The CH aggregates data in its cluster and sends it to the BS using a multi-hop path created using intermediate CHs. The intermediate CHs may also aggregate data from their neighbouring CHs before sending it towards the BS. The BS acts as the interface for the sensor network to the internet or a wired network. It is assumed to have unlimited power source and computation ability, that is, orders of magnitude higher than the sensor nodes themselves. We assume that the BS is secure and is not compromised by any malicious user.

We assume that the clustering algorithm, HEED (Younis and Fahmy, 2004) is used for cluster formation, with an added assumption that the SNs are static after deployment. These SNs have the ability to transmit at several discrete power levels. The highest power level is used for inter-cluster communication. The lower power levels are used for intra-cluster communication and are called the *cluster power levels* (Younis and Fahmy, 2004). Clustering and CH election are done on the basis of residual energy, while the *average minimum reachability power* as proposed by Younis and Fahmy, (2004) is used to break ties. Each node declares itself a CH with a probability that is dependent on its residual energy. The process of CH selection goes through many iterations. At the end of the iterations, a node that is neither a CH nor part of any cluster declares itself as a CH. To ensure the inter-cluster connectivity, we assume, that the routing mechanism used in Tiny OS, namely, *base station beaconing* (Karlof and Wagner, 2003), is used. The base station beaconing protocol constructs a breadth first spanning tree of the network, rooted at the BS. The BS broadcasts a route update beacon periodically. All nodes receiving the BS's beacon designate it as their parent and forward the beacon using their ID as the sender ID. A node receiving this beacon designates its parent as the node whose ID is in the sender field. This algorithm continues recursively till it terminates at the periphery of the network. In our scheme only the CHs forward the BS beacon message, thus creating a connected network of CHs from the BS to the periphery of the network. Given the higher order transmit power used for inter-cluster communication, we can assume, as assumed by Ye et al. (2003) and Younis and Fahmy (2004), that complete network connectivity is guaranteed.

3.2 Security assumptions

The BS acts as the key server and also shares a key with every SN in the network for authentic and confidential communication. The SNs are identical to the current generation TelosB motes (Polastre et al., 2005) in their computation, communication and power resources. We assume that the SNs have adequate memory for storing up to hundreds of bytes of keying material to be used by our anonymity schemes. During initial set-up and neighbourhood discovery, the SNs in a neighbourhood exchange their identity information for key set-up. For the complete mechanism of such a set-up, we refer the readers to the work of Du et al. (2005). After the key set-up, each SN can communicate securely with every other SN in its neighbourhood and authenticate messages using the shared pairwise keys. Also, each SN shares a secret cluster key with every other SN in its neighbourhood. These cluster keys may be generated by each SN and sent securely to every neighbour using the shared pairwise secret key. The cluster key is used by an SN for secure intra-cluster communication in its neighbourhood as the CH. Zhu et al. (2003) assumed that in a WSN there exists a lower bound on the time interval ($T_{\min}$) that is necessary for an adversary to compromise an SN. The initial set-up for our first anonymity scheme, which involves exchange of a few encrypted range messages between neighbours, is possible in time much less than $T_{\min}$. Also, the set-up phase of the second scheme, which requires the exchange of some parameters between the neighbours, can be easily completed in time much less than $T_{\min}$.

3.3 Characteristics of CWSN

In this paper, we propose two anonymity and privacy solutions for a CWSN that are characterised by the following attributes:

1 All nodes in the CWSN are loosely time synchronised.

2 All SNs outside the range of transmission of a given SN cannot comprehend its transmission and treat it as noise.

3 The SNs have enough compute power to generate pseudo-random numbers.

4 SNs do not communicate among themselves other than during the initial set-up or the CH election phase. All other communications happen between the CH and the SNs or between the CHs, and are of broadcast or unicast nature.

5 Similar to the assumption in Kong and Hong (2003), we assume that the SNs have the ability to obfuscate address fields in their medium access control layer header. This ensures that an SN's medium access control layer header does not give out its identity to a compromised SN in the neighbourhood.

3.4 Adversary and threat model

Generally two types of attackers are considered for a wireless sensor network: a *sensor* class attacker, also known as the *inside attacker*, and the high power *laptop* class attacker, also known as the *outside attacker*. The inside attacker might consist of more than one compromised SN that can mount a concerted attack on the network. On the other hand, a laptop class attacker has higher capacity than the SNs in the network and can jam or eavesdrop on the entire network, or create wormholes or sinkholes (Karlof and Wagner, 2003). In this paper, we assume an adversary that is much stronger than the sensor nodes in the network. The adversary is capable of both insider and outsider attacks, but has bounded computing and traffic analysing abilities.

Communication of an SN with the BS and pairwise one-hop neighbours is confidential and authenticated. The adversary would not be able to decrypt any communication until it compromises the nodes in the network. However, it can identify the centres of stimuli in the network by looking at the source and destination IDs in the packet headers. As a direct consequence, it can identify the clusters sending important information and infer the IDs of the CHs in the clusters. Knowing the ID of the CH, the adversary can infer its location and compromise/destroy it, in turn rendering all communications in the cluster compromised.

A compromised CH closer to the BS shall also allow the adversary to monitor any communication happening through it. Furthermore, an adversary can obtain routing information from the compromised nodes or by eavesdropping. By analysing this information, it can obtain knowledge of the network's topology, thus, becoming equipped to disrupt the network.

3.5 Requirements for anonymity in a CWSN

Based on Section 3.2, we can define the anonymity requirements of a CWSN consisting of the following:

1 Every SN can communicate with any other SN in its neighbourhood and the BS in an anonymous and a secure manner.

2 Routing of messages is anonymous. The CHs that are in the forwarding path of a CH to the BS cannot infer its true ID.

3 The nodes in a cluster are indistinguishable. A malicious agent that is not a part of the CWSN should not be able to identify the nodes involved in communication.

4 SNs outside the neighbourhood of a cluster cannot figure out the CH of the cluster. This entails that when the CH communicates in its cluster, any other node not in the cluster cannot identify that it is the CH. Furthermore, when a CH communicates with a neighbouring CH, no other SN can identify it.

5 In essence, any anonymity solution in a CWSN environment should provide the following three kinds of privacy (Zhu et al., 2003):

- *Identity privacy:* this ensures that neither the destination node knows the true identity of the source node of a packet nor do the source and destination know the true identity of the intermediate forwarding nodes.
- *Location privacy:* a node in the network should not know the location of any other node in the network.
- *Route privacy:* an adversary, irrespective of whether it is on the route of a packet it has received or not, cannot trace the source of the packet or the intermediate nodes the packet has traversed.

We note that the final destination of all inter-cluster packets is the BS. This can be inferred from the final destination field in the packet. However, this does not aid in traffic analysis for identifying the source of the packet, as all packets are destined to the BS.

In a key exchange scheme such as that suggested by Blom (1985), a neighbouring SN needs to identify the sender of a packet to be able to use the correct pairwise key to decrypt the packet's contents. The *identity privacy* requirement has to be enforced in a way that the receiver recognises the sender to be able to select the correct key for decryption. However, it should not be able to determine the sender's true identity. We address this important aspect in our solution. We do not consider the requirement of *location privacy*, as the clustering and routing schemes we use do not exchange any location information. We address the issue of *route anonymity* but do not propose any new routing algorithm. Our scheme can be used with any existing routing algorithm, to ensure that node discovery, route requests, and route replies use pseudonyms and the true identity of a node is kept private.

4 Framework for the anonymity schemes

In this section, we present the basic framework and definitions we use to build our anonymity schemes. Table 1 gives a list of notations used and their meaning. We shall define some of the notations that are not implicitly understandable at the place of their use.

Defined below are a few terms that we are going to use in the rest of the paper.

Definition 1: *A node u in the network is said to have complete anonymity, if no node v that captures packets sent by u has*

any way to identify that the sender is u, in spite of having knowledge of the IDs in the packets.

Table 1 Notations table

Notation	*Explanation*
N	number of nodes in the network
M	number of neighbours of a node in the network
K	number of bits used for the pseudonym space
u, v	nodes that we shall use in our illustrations
k'_{uc}	secret key used by SN u for intra-cluster communication as a CH
k'_{uv}	mutually shared secret key used by SNs u and v for encrypting communication
k_{uc}	shared secret hash key used by SN u for anonymous intra-cluster communication
k_{uv}	mutually shared secret hash key used by SNs u and v for anonymising the header
F	is a family of pseudo-random functions
H	a set of hash functions s.t. for each $k \in \mathbf{K}$ there is a hash function $h_k \in \mathbf{H}$. Each h_k: $\mathbf{X} \longrightarrow \mathbf{Y}$
(X, Y, K, H)	defined as a hash family
$2^{\|\ell\|}$	pseudonym subrange each SN assigns to each neighbour in the SAS
$\|\|$	the concatenation operator

Definition 2: *The neighbourhood set N_i of a node i is the set containing all its neighbours including itself. The common neighbourhood set, S_C, of a group of colluding compromised nodes C, is defined as the intersection of the neighbourhood sets of the compromised nodes, excluding the compromised nodes themselves. If the set of colluding nodes is defined as, $C = \{1, 2, \ldots, C\}$, then, $S_C = \{N_1 \cap N_2 \cap \ldots \cap N_C\} \setminus \{C\}$.*

In the following two definitions, we define some terms we shall use in the design of our second anonymity scheme. They pertain to some basic cryptographic primitives as specified by Stinson (2002).

Definition 3: *Let $F^{X,Y}$ denote the set of all functions from X to Y, where $|X| = N$ and $|Y| = M$. Then $|F^{X,Y}| = M^N$. Any hash family $F \subseteq F^{X,Y}$ is termed an (N, M) hash family.*

Definition 4: *A keyed hash family is a four tuple (X, Y, K, H), where for each $k \in K$, there is hash function $h_k \in H$ where $h_k : X \rightarrow Y$. A (N, M) hash family in this context is defined as one in which each $h_k : X \rightarrow Y$, where $|X| = N$ and $|Y| = M$.*

The two schemes we propose ensure that a node in the network has complete anonymity during the communication with uncompromised nodes, even when colluding compromised nodes exist in its neighbourhood. Our interpretation for anonymity is that if an SN's true ID is not known to other nodes in the network, they cannot infer it, hence it shall be anonymous. Hence, an SN should use a pseudonym to identify itself. However, use of a static pseudonym is as bad as using true ID, as it renders the SN open to traffic analysis by the adversary. The idea of using the shared symmetric keys to encrypt the true ID and using the encrypted ID as a pseudonym has the same drawback as well. Also, if the ID is encrypted, the receiver does not know which node has sent the message. Hence, in the worst case, to identify the sender, it might have to decrypt the encrypted pseudonym, with the symmetric keys it shares with each SN in its neighbourhood, which is computationally expensive. A better solution is for the SNs to use a range of indistinguishable pseudonyms while communicating with other SNs in the network. However, we still have to ensure that the receiver is able to identify the sender in order to select the proper key to decrypt the mutual communication. To identify the sender, a receiver may store a mapping of the relevant pseudonym ranges of the sender with the mutually shared key. We shall describe this in more detail in Section 5. At the time of set-up and neighbourhood discovery, the SNs in a neighbourhood exchange the information to recognise each other's pseudonyms during the later communication. Once this information is exchanged, the SNs delete the information needed to decipher each other's true identity. We assume as specified in Section 3.2, that is, in this initial period of exchange of the pseudonym ranges as required by our first scheme or parameters as required by our second scheme, the SNs are not compromised (Zhu et al., 2003, 2004a,b), and follow the respective outlined procedures correctly.

5 Simple anonymity scheme

We shall propose here the Simple Anonymity Scheme (SAS). For anonymity, we use a 'K' bit pseudonym scheme for all the nodes in the network. Hence, the pseudonym space range for the K bits pseudonyms is, $0 - 2^K - 1$, a total of 2^K pseudonyms. We refer to it as the *pseudonym space*. Further, we assume that a sensor node u uses, $L = 2^{|\ell|}$ pseudonyms for communicating with each neighbour (excluding itself). This pseudonym sub-range for each neighbour is contiguous. Each packet sent by u to neighbour v has a pseudonym chosen randomly from the corresponding subrange. This usage of dynamic pseudonyms by u for sending messages renders traffic analysis ineffective.

SAS is a simple scheme for ensuring ID anonymity and privacy for an SN even when a significant number of its neighbours is compromised and are colluding. We describe this scheme in two broad phases in Sections 5.1 and 5.2.

5.1 Predeployment phase

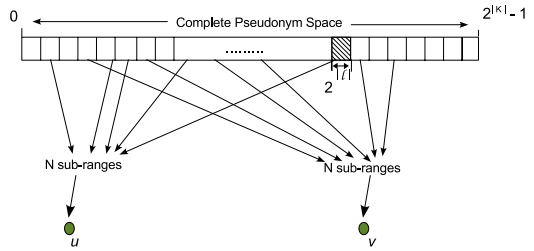
Before deployment, the predeployment authority:

1 Divides the IDs from the pseudonym space, uniformly into subranges of size $2^{|\ell|}$ each. The value of ℓ is chosen in such a way that the pseudonym space can be divided into at least N^2 subranges.

2 Assigns each SN u, N randomly chosen subranges, distributed uniformly in the pseudonym space. Figure 1 illustrates such an assignment. Hence, node u shall have N ranges of size $2^{|\ell|}$ to use in place of its true ID.

3 Creates a table at the BS that stores the pseudonym ranges of each node u. This ensures that when the BS receives packets from u it is able to figure out the correct key to decrypt and authenticate the message.

Figure 1 Pseudonyms ID space assignment



5.2 *Postdeployment phase*

After the SNs are deployed:

1 Each SN u randomly chooses one subrange from its N subranges to ensure anonymity while forwarding the BS beacons. The beacons forwarded by u identifies node u as the CH, as only the CHs forward the beacons.

2 The beacon subrange are also used by CH u when broadcasting messages in its cluster. SNs outside the boundary of the cluster cannot identify these messages as they are sent using only the cluster power level.

3 Each neighbour v of u is also assigned a pseudonym sub-range, chosen uniformly at random from the remaining $N - 1$ subranges.

4 Each SN u has a pseudonyms table it uses to store the subranges for communication with other SNs in its neighbourhood. The table maps the pseudonym subranges that u uses to communicate with a neighbour v and the subranges v uses for u, to the corresponding pairwise key shared between them.

5 To each neighbour v, u securely communicates the beaconing subrange and the pseudonym subrange that u has assigned for mutual communication with v. To prevent cases of simultaneous communication, we assume that the SN with higher true ID starts the communication. Here, we assume $u > v$.

6 SN u also sends the index in its pseudonym table where it shall store the range information for v, in the same message. We shall explain the need for the index information later.

7 When node v receives the range message from u, it selects a random subrange from its pseudonym subranges for communication with u. It then stores this information about the subranges for mutual/beacon communication with u and u's index along with the mutual key in its pseudonym table. Then v sends its beacon sub-range and the sub-range for mutual communication with u along with the index of the information in its pseudonym table to u. Further, v *deletes* the true ID of u. Now, v can only identify u by the pseudonyms u shares with it.

8 When u receives the message from v, it stores the subrange and index information in the appropriate position in its pseudonym table and *deletes* the true ID of v. Figure 2 shows the pseudonym table of u with the entry for SN v, containing the stored subranges, index of v and the mutual secret key.

9 When v wants to communicate with u, it chooses an ID, (ID_{vu}), randomly from the pseudonym subrange, $\mathrm{ID}_{vu1} - \mathrm{ID}_{vu2}$, it shares with u and another random ID, (ID_{uv}), from the subrange, $\mathrm{ID}_{uv1} - \mathrm{ID}_{uv2}$, u shares with it, for mutual communication. The sender ID and receiver ID are generated as follows: Sender ID = $\mathrm{Index}_u||\mathrm{ID}_{vu}$, where Index_u = Index where u stores information about v, and Receiver ID = $\mathrm{Index}_u||\mathrm{ID}_{vu}$.

10 When node u receives the message, it checks the sender ID and uses the index (Index_u), to index into its pseudonym table and compare the sender ID with the subrange of v it has stored in the table. If the pseudonym is in the subrange, it identifies that the packet is from the correct source. Note that our reference to nodes u and v is simply for illustration. Node identification is based solely on the pseudonym ranges.

11 When CH u wants to communicate with the BS through a neighbour CH v, it uses the same sender ID for both node v and BS. The BS has information about the ranges of all nodes. It disregards the index information and uses the pseudonym itself to identify the source using a stored reference table.

12 A CH forwards the BS beacon using a pseudonym from its beacon subrange as sender ID. Nodes in its neighbourhood check their table for pseudonym match and identify the CH. They mark the CH's index in their pseudonym table for future communication with the CH. When a CH communicates in its cluster neighbourhood, it uses a special sentinel character as index. When a cluster SN gets this message, it uses the index it has stored as the CH's index to identify the pseudonym.

Use of index will not aid traffic analysis, as the same index will be used by different SNs in a neighbourhood for mutual communication. Furthermore, the IDs used by an SN are dynamic. Thus, knowledge of the index cannot help the adversary to correctly infer the communicating SNs.

Figure 2 Pseudonym table for node u

Index	u's range	Neighbor's Range	Neighbor's Beacon Range	Neighbor's Index	Shared Key
$Index_u$ . . .	ID_{uv1} - ID_{uv2}	ID_{vu1}-ID_{vu2}	ID_{bv1}-ID_{bv2}	$Index_v$	K_{uv}

5.3 Node revocation

In this subsection, we shall discuss the issue of node revocation and illustrate how it is handled by the SAS. The issue of revocation is a practical one in a WSN. In the event of identification of one or more compromised SNs, it is essential that they be revoked. In this paper, we do not deal with the issue of how one or more compromised SNs are detected or how the revocation procedure is effected. We assume that there exist mechanisms to detect and revoke compromised SNs. Using these existing mechanisms, our scheme works on top to provide anonymity during the communication, given the presence of one or more compromised/revoked SNs in the network.

To ensure complete revocation of an SN u in the anonymous network, it is essential that u is unable to identify the source of any anonymous intra-cluster communication in its neighbourhood. To this effect, all the SNs in the neighbourhood of u need to share new pseudonym sub-ranges to generate anonymous IDs for intra-cluster communication. Generally, the neighbourhood size of a node is smaller than the total size of the network. Thus, in general, each SN has several free pseudonym subranges from its N subranges. When the SNs in a neighbourhood identify a compromised SN, by whichever mechanism, they can exchange new pseudonym subranges among themselves and use them for anonymous beacon/cluster communication. The compromised SN has no means of identifying these ranges, as it was not involved in these exchanges. Mutual communication between two SNs is not compromised in any way given the presence of compromised SNs. This is because the compromised SNs have no idea of the subranges used by other non-compromised SNs for their communication.

5.4 Anonymity analysis of SAS protocol

Node anonymity in SAS is due to each SN using randomly chosen IDs from the corresponding pseudonym subranges when it wants to communicate with a given neighbour or broadcast the BS beacon. Given some communication between two non-compromised nodes in a neighbourhood, a group of colluding, compromised neighbours cannot infer the source/destination of the communication. This is because an SN chooses the pseudonym subrange for each neighbour randomly. These chosen pseudonym sub-ranges are non-overlapping and non-contiguous. Thus, there is no way, given the knowledge of the sub-ranges of an SN, the colluding nodes can figure out the other sub-ranges the SN is using. Thus, our scheme ensures that a sender is guaranteed complete anonymity in communication with an uncompromised SN despite the existence of compromised, colluding SNs in the neighbourhood. We give below a theorem that further illustrates the level of anonymity provided by SAS.

Theorem 1: *Let, S denote the common neighbourhood of k compromised nodes that are colluding. Assume that a unicast communication between two uncompromised nodes in S is heard by the k colluding nodes:*

a) If $S > 1$, then there is no way for the k colluding nodes to identify the sender/receiver of the message.

b) The probability that the k compromised colluding nodes will guess the sender correctly is, $1/|S|$.

Proof: a) The N subranges for each node are chosen randomly. Further, each node selects a subrange randomly for each of its neighbours and the beacon. Each neighbour of u shares two pseudonym subranges with it, the beacon subrange being common to all. The common neighbourhood is formed by the k nodes sampling only the packets accessible to all of them. The k colluding nodes know only $k + 1$ subranges of a node u in this neighbourhood. If $|S| = 1$, the k nodes shall be able to identify the communication is from the only node in S. If $S > 1$, the k nodes cannot identify the sender/receiver of the message, as they cannot infer the pseudonym ranges.

Proof: b) The compromised nodes know $k + 1$ of the N sub-ranges of each common neighbour. A pseudonym that belongs to any of these subranges will be recognised by the compromised nodes with probability 1, because it is addressed to one of them. Any pseudonym from outside these subranges is chosen uniformly from the remaining $\{N|S| - (k + 1)|S|\}$ subranges. The probability that the pseudonym belongs to a node, $u \in S$, is $1/|S|$. Hence, if the colluding nodes try to guess the sender, the chance that their guess is right is only $1/|S|$.

According to Theorem 5.4(b), even if there are only 2 uncompromised nodes in the neighbourhood, the probability that the compromised neighbours can even guess correctly which uncompromised node is transmitting is only $1/2$. The larger the number of uncompromised nodes in a neighbourhood, the lower the probability that the compromised nodes can guess the sender of a packet correctly.

5.5 Memory and computation requirements

In the SAS, each node u stores 2 subranges for each of its neighbours. u also stores its N pseudonym subranges. Consider a K bits pseudonym space and the neighbourhood size of a node upper bounded by M. Each node has to store $4KM + 2KN$ bits for the ranges. For example, for an ID space of 64 bits, with a CWSN of 1000 nodes, and an average neighbourhood size of 100 nodes, the memory requirement is $4 \times 64 \times 100 + 2 \times 64 \times 1000 = 153.6$ KBits $= 19.2$ KB. Assuming 2 bytes for storing the index per entry in the table, the total memory requirement is 19.4 KB. In a CWSN of 10, 000 nodes, and average neighbourhood size of 1000 nodes, total memory requirement should be 192.2 KB. The

TelosB motes (Polastre et al., 2005) we use have an external flash memory of size 1 MB along with the internal RAM of 48 KB. These motes use the Von Neumann architecture, wherein the complete memory space is accessible for code. Hence, the range information could be stored in the external flash as dynamically loadable modules.

The computation involved in deciphering the sender pseudonym involves indexing into the pseudonym table using the index prepended to the sender pseudonym, and checking if the pseudonym falls in the sender's subrange. Hence, the total computation complexity for pseudonym checking is O(1). There are ways of improving the memory utilisation by using hashing, compression, etc.; however, we do not discuss them in this paper.

6 Cryptographic anonymity scheme

In this section, we propose our second scheme, named the Cryptographic Anonymity Scheme (CAS). To start with, we define the term computational resistance property for a given keyed hash function.

Definition 5: *Given a keyed one way hash family (X; Y; K; H), as defined in definition 4, with the domain* $|X| = |Y| = K$ *and a hash function* $h_k \in H$*, the computational resistance (CR) property implies that an adversary that has no prior knowledge of the key k, cannot compute a new text-MAC pair* $(x, h_k(x))$ *for some text* $x \neq x_i$*, given its knowledge of one or more pairs* $(x_i, h_k(x_i))$*.*

In CAS, we use the above definition of the CR property (Menezes et al., 1996, Chapter 9) of Keyed Hash Function (*KHF*) and further augment it to generate unrelated pseudonyms for a given SN. Hence, the pseudonyms cannot be identified nor generated by an outside adversary even if it captures previous messages sent by the SN. The CR property is used in Message Authentication Codes (MAC) to generate the MAC value, $h_k(x_i)$ for any message x_i, while precluding the possibility of an adversary being able to generate the same. In our construction, a part of each x_i is a secret, which is unknown to an outside adversary, making it even more difficult for the adversary to be able to generate a valid pseudonym.

Similar to the SAS, in CAS we use a K bit pseudonym with the pseudonym space lying between 0 and $2^K - 1$. The CAS ensures that an SN is anonymous and its ID remains private in spite of several compromised colluding SNs in the neighbourhood. Sections 6.1 and 6.2 detail and analyse the scheme in depth.

6.1 Predeployment phase

Before the SNs are deployed, the predeployment authority:

1 Generates and assigns to each SN u a pseudo-random function $f^u \in$ F, where F is the pseudo-random function family (Goldreich et al., 1986). SN u uses this function to generate keys for the keyed hash function used to generate the pseudonyms.

2 Loads on each SN u the parameters required by u to anonymously communicate with the BS. The parameters are:

- The hashed key k_{Bu} that u uses to anonymise its ID when sending a message to the BS. $k_{Bu} = f_{K'_{Bu}}(u)$, where f is a pseudo-random function used by the predeployment authority, and K'_{Bu} is the mutually shared key between the BS and u for secret communication.
- A randomly generated variable a_{Bu} that is used by u as a seed to generate anonymous sender ID. The procedure is defined in Section 6.2.

6.2 Set-up phase

In the set-up phase, the SNs in a neighbourhood exchange required parameters to set up the mechanism to create pseudonyms, for anonymous unicast or broadcast communication in a neighbourhood. For an SN u to communicate anonymously with a neighbour v, they require to share the following three parameters:

- the hash key k_{uv}.
- a randomly generated seed, a_{uv} used by each node to generate the receiver ID.
- the starting sequence number seq_{uv}. Without loss of generality, we assume that all sequence numbers used in our scheme start at 1. Each packet contains the sequence number, whose value is incremented by 1, by the sender after transmission of the packet and the receiver after reception of the packet.

Each SN also shares four parameters with all its neighbours, for anonymous cluster communication when it performs the role of the CH. For instance, SN u shares hash key k_{cu}, two seeds a_{cu} and b_{cu}, and the sequence number seq_{cu} with all its neighbours. For our illustrations, we assume that the ID of SN $u > v$. Then the parameters exchange takes place in the following manner.

The set-up phase consists of the following steps:

1 After the initial neighbour discovery is completed, SN u generates the hash key k_{uv} that it shall share with SN v. k_{uv} is generated using the pseudo-random function f^u as, $k_{uv} = f^u_{k'_{uv}}(v)$, where k'_{uv} is the secret key shared by SN u and v during the pairwise key set-up for secret mutual communication, and v is the ID of node v.

2 SN u also generates the random seed a_{uv}.

3 Before exchanging the parameters for mutual anonymous communication with its neighbours, each node u generates the parameters, it requires for anonymous broadcast communication as a CH. u generates the hashed key $k_{cu} = f^u_{k'_{cu}}(u)$, where k'_{cu} is the cluster key u shares with all its neighbours. And u also generates the random seeds a_{cu} and b_{cu}.

4 For each SN v in its neighbourhood that has an ID less than itself, u generates the information needed for anonymous communication and stores it in a pseudonym table. Also, u securely communicates this information for mutual anonymous communication to

each neighbour v, along with the information for v to decipher cluster communication when u is the CH. In the same message, u also sends the index in the pseudonym table where it stores this information.

5 SN v responds by sending a message to u containing the information needed for u to decipher the anonymous cluster messages sent by v as the CH, and also the index in the pseudonym table where it stores all the information corresponding to u. Figure 3 illustrates the messages communicated between u and v.

6 After the secure mutual exchange of the information needed for set-up, the pseudonym table entries of each node u stores the information required for mutual communication and intra-cluster communication, with each of its neighbours. u also stores the indexes in the pseudonym table of its neighbours where its information is stored. A representative entry in SN u's pseudonym table corresponding to neighbour v is illustrated in Figure 4. Further, u deletes the true ID of the neighbours and also the pseudo-random function f^u used to generate the keys for the hash.

Deleting f^u prevents an adversary that compromises u to figure out the true IDs of the neighbours of u. This is possible if the adversary uses the brute-force method for creating the keys for all possible IDs using f^u, and compares it with the already existing key for a neighbour.

Figure 3 Set-up phase message exchanges between u and v

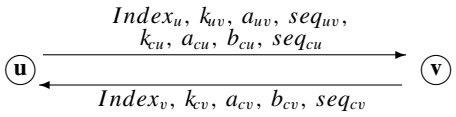


Figure 4 Pseudonym table for node u in CAS

Index	Mutual Exchange		Nhbr's Cluster Exchg			Nhbr's Cluster	Nhbr's Hash	Shared	Shared Hash	Nhbr's
	a	seq	a	b	seq	Key	Key	Key	Key	Index
$Index_u$	a_{uv}	seq_{uv}	a_{cv}	b_{cv}	seq_{cv}	k'_{cv}	k_{cv}	k'_{uv}	k_{uv}	$Index_v$

Similar to SAS, the index information is used by the receiving SN to identify the parameters to use for deciphering, if it is the intended receiver. Without the use of the index, it becomes computationally intensive for the receiver to figure out which parameters to use to identify if it is the receiver and further, decrypt the message if it is the intended receiver. As we have noted before, use of index in the message does not aid traffic analysis as many messages in a neighbourhood will have the same index.

6.3 Operation phase

In this subsection, we shall illustrate how the CAS provides anonymity during communication between the SNs in the network. For illustration we shall use the two SNs, u and v as before. For mutual pairwise communication (here we assume without loss of generality that u starts the communication) the two SNs follow the procedure given below.

6.3.1 Mutual communication procedure

1 Any message from u to v (with BS as the final destination) is composed as, $M_{u\to v}$: SID || RID || EncryptedPayload || seq_{uv}, where the value of seq_{uv} is the current sequence number value for mutual communication between u and v. The sender ID is, SID = Index_v $||h_{k_{Bu}}(a_{Bu} \oplus \text{seq}_{uv})$, the receiver ID is, RID = Index_v $||h_{k_{uv}}(a_{uv} \oplus \text{seq}_{uv})$, and $\oplus$ is the XOR operator. Let us define, sid = $h_{k_{Bu}}(a_{Bu} \oplus \text{seq}_{uv})$ and rid = $h_{k_{uv}}(a_{uv} \oplus \text{seq}_{uv})$ for later use. u increments the value of seq_{uv} by 1 for the next message. The SID generated is used by u as the sender ID for both the forwarding neighbour v and the BS.

2 When v receives $M_{u\to v}$, it uses Index_v in the message to index into its pseudonym table and obtain the values, k_{uv} and a_{uv}. Using the obtained values and seq_{uv} from the message, it constructs $\text{rid}' = h_{k_{uv}}(a_{uv} \oplus \text{seq}_{uv})$. If $\text{rid}' = \text{rid}$, then v has verified the sender and may use the corresponding key k'_{uv} to decrypt the payload. If $\text{rid}' \neq \text{rid}$, v drops the packet as it is not the intended receiver. At the end of this procedure, if v received a packet that passed the verification test, v increments the value of seq_{uv} by 1.

3 When v forwards the message it receives from u, the payload of the message consists of EncryptedPayload|| seq_{uv}, as the BS needs the seq_{uv} value to identify the source of the packet. Any SN on the path to the BS uses this payload.

4 When the BS receives the message sent from u via some SN w in its neighbourhood, it decrypts the payload with the mutually shared secret key k'_{Bw} and gets the sequence number in the payload sent by u. It then uses the stored values of the random seed for all the SNs and the sequence number obtained from the message to identify the sender. It follows the same procedure as u's neighbour v. When BS verifies u to be the sender it uses the shared key to decrypt the message.

For the computationally powerful BS, which might be a PC class machine, the process of finding a match for the sender ID in a 1000 node network for a 160 bit ID generally takes less than 0.7 ms (Black et al., 1999). For the 64-bit or 32-bit pseudonym implementations of the schemes suggested in this paper, the time taken shall be much less.

Even in the case of loss of many consecutive packets during communication between two SNs, the receiver is able to correctly receive and identify a packet intended for it, and is even able to deduce the number of packets that were lost, which is equal to $\text{seq}_{\text{current}} - \text{seq}_{\text{previous}}$. In this context, $\text{seq}_{\text{current}}$ = sequence number in the currently received packet and $\text{seq}_{\text{previous}}$ = sequence number of the last correctly received packet.

A node w that is in the neighbourhood of u and v and not the recipient of the message $M_{u\to v}$, follows the same

procedure as v to identify if its the receiver, and discards the packet when it finds that $\text{rid}' \neq \text{rid}$.

The intra-cluster message sent by a CH may either be a broadcast of the beacon message or some information for the SNs in the cluster. For any such intra-cluster communication originating from a CH, say u to the SNs in the cluster, the procedure adopted is defined below.

6.3.2 Intra-cluster communication

1 The intra-cluster message $M_{u\to *}$, sent by CH u is composed of the following: $M_{u\to *}$: SID || RID || EncryptedPayload|| seq_{cu}, where SID = Sentinel$||h_{k_{cu}}(a_{cu} \oplus \text{seq}_{cu})$ and RID = Sentinel$||h_{k_{cu}}(b_{cu} \oplus \text{seq}_{cu})$. Here we define $\text{sid} = h_{k_{cu}}(a_{cu} \oplus \text{seq}_{cu})$ and $\text{rid} = h_{k_{cu}}(b_{cu} \oplus \text{seq}_{cu})$. The sentinel is for the receiving SNs to identify that the communication is from the CH. This mechanism is the same as the one used in SAS. u then increments seq_{cu} by 1.

2 When an SN v in the cluster receives the first intra-cluster message from the CH, it follows the same procedure as in SAS. v goes through all neighbours in its pseudonym table and uses the a_{cx} value corresponding to each neighbour x to calculate $\text{sid}' = h_{k_{cx}}(a_{cx} \oplus \text{seq}_{cx})$. It checks for which neighbour, $\text{sid}' = \text{sid}$ in the received packet. That neighbour is the CH. Then v marks the corresponding index as the CH for future communication from the CH.

3 For all the packets sent by the CH after the first one, each SN in the cluster neighbourhood uses the stored index of the CH to index into its pseudonym table and verify the sender pseudonym and also decrypt the payload of the message.

For a communication between two pairs of nodes in a neighbourhood, there is a chance that the index values used in the sender/receiver IDs is the same. In addition, given the pseudo-random nature of the pseudonyms generated, there is a possibility that the IDs generated by the two simultaneously communicating nodes in the neighbourhood might match, leading to an ID collision. We prove below that the probability of such an occurrence is very low. It is understood that the pseudonym generated by one or more nodes in a neighbourhood are mutually independent. These pseudonyms appear as pseudo-random sequences for a node not a party to a given communication, due to the avalanche effect (Fiestel, 1973) of the hash function, and also due to the fact that the hash key as well as a part of the input are unknown to it. Given the use of a K-bit pseudonym, the following theorem shows that two IDs generated by two different transmitting nodes in a neighbourhood, where M nodes are transmitting, is the same with very low probability.

Theorem 2: *If there are M simultaneously transmitting nodes in a given neighbourhood, the probability that the sender or the receiver pseudonyms collide for any two simultaneous communication originating from two different nodes in the neighbourhood is,* $2\begin{pmatrix} M \\ 2 \end{pmatrix} 1/2^K$.

Proof: The choice of two simultaneously communicating nodes in a neighbourhood consisting of M simultaneously communication nodes is given by, $\begin{pmatrix} M \\ 2 \end{pmatrix}$. Given a choice of two communicating nodes, let one of the nodes generate a sender and a receiver pseudonym. Given the pseudo-random and mutually independent nature of the pseudonym generation procedure, the probability that the sender pseudonym generated by the other node is the same is only $1/2^K$, as all the IDs in the ID space are equally likely. The case for the receiver ID may be similarly argued. Hence, the probability that there is a collision of either the sender or the receiver ID for the two communicating nodes is $2 \times 1/2^K$. Thus, the probability that the sender or the receiver pseudonyms collide for any two simultaneous communication in a neighbourhood is given by, $2\begin{pmatrix} M \\ 2 \end{pmatrix} 1/2^K$.

For example, if we assume that we are using 64-bit pseudonyms, the probability that there shall be an ID collision in a neighbourhood of size 100 is, $2\begin{pmatrix} M \\ 2 \end{pmatrix} 1/2^{64} = 5.3668 \times 10^{-16}$. This implies that there is a probability of an ID collision of 1 in every 1.8633×10^{15} packets, a very low probability. Given a 250-kbps channel capacity, as supported by TelosB motes, a possible collision may happen in 7020 years, which is much more than the supposed lifetime of any current generation WSNs.

6.4 Node revocation

In this subsection, we shall discuss the issues of node revocation and addition of new nodes post network deployment and illustrate how they are handled by the CAS. As we have explained while considering node revocation in the SAS, we do not deal with the issue of how one or more compromised SNs are detected, or how the revocation procedure is effected. We assume there exist mechanisms to detect and revoke compromised SNs. To ensure complete revocation of an SN u in the anonymous network, it is essential that u is unable to identify the source of any anonymous intra-cluster communication in its neighbourhood. To this effect, all the SNs in the neighbourhood of u need to share new keys and random seeds, to be used by the hash function to generate anonymous IDs for intra-cluster communication. However, as specified in Section 6.2, each SN deletes its pseudo-random function to ensure that if it is compromised by an adversary, the adversary cannot identify the other SNs in the neighbourhood. SNs in the compromised neighbourhood elicit the help of the BS in this scenario. Each uncompromised SN v in the neighbourhood requests the BS for a pseudo-random function. The BS generates and sends a different pseudo-random function $f^{v'}$ to each v, in a secure manner. Using $f^{v'}$ and its own ID, v generates a new key for the cluster hash function and also new values for a_{cv} and b_{cv} and securely communicates these parameters to each of its non-revoked neighbours as a unicast message.

After the exchange of parameters, each SN in the neighbourhood updates its pseudonym table. With the use

of the new parameters for intra-cluster communication, the compromised SNs have no means of identifying the CH sending the intra-cluster message. Mutual communication between two non-compromised SNs in a neighbourhood is still uncompromised as the compromised SNs still have no way of identifying the source/destination pseudonyms.

6.5 Anonymity analysis of CAS protocol

In CAS, SN anonymity results from each SN using a keyed hash function, which is strongly forgery resistant, to generate anonymous sender and receiver IDs. A group of colluding nodes in a neighbourhood have no means of finding out the source or destination of an anonymous communication between two uncompromised nodes in the neighbourhood. Despite of the fact that the hash function is known to all SNs, the compromised SNs have no idea of the key used by the hash function nor the starting parameters used in the hash function. The only information that the compromised nodes have is the sequence number.

It might appear that since a part of the input to the keyed hash function is constant, it may be forged or guessed by the adversary. However, in reality this is not the case. There are two reasons to support this argument. First, the keyed hash function used by us and also the MAC schemes, such as Cipher-Block-Chaining (CBC-MAC) or Hashed MAC (HMAC), are strongly pseudo-random (Bellare et al., 1996a,b), which provides them the CR property. As a result, these functions exhibit a property called the *avalanche effect* (Fiestel, 1973). Avalanche effect ensures that even if two inputs to the hash function differ by atmost a single bit, the corresponding outputs obtained from the hash function are totally unrelated. In general, an adversary cannot find a relation between the two inputs and the corresponding outputs in time to compromise the system, even when it knows the inputs. Second, in CAS we further augment the unforgeability of the scheme by keeping the seeding value a secret, which is unknown to any node other than the two communicating nodes. This arrangement ensures that the adversary cannot mount any of the attacks possible on MAC, such as known-text attack, chosen-text attack, or adaptive chosen-text attack (Menezes et al., 1996, Chapter 9). The adversary has no mechanism to interact with the sender to have a choice in the text used. In this paper, we use CBC-MAC that has a block of size $K/2$, where K is the bit-size of the pseudonyms, as the keyed hash function. CBC-MAC is provably secure for fixed length input as specified by Karlof et al. (2004). An improved variant of CBC-MAC may also be used as specified by Bellare et al. (2000), if enhanced security is desired. To be able to forge a pseudonym, the adversary has to correctly guess the key as well as the seed value and use the correct sequence number, to input to the pseudonym generation procedure. The probability that the adversary can get all the steps right is very low.

Thus, we conclude that the partial information with the adversary is inadequate for identifying the communicating SNs. Hence, the CAS guarantees a pair of uncompromised nodes complete anonymity during communication, despite the existence of many compromised and colluding SNs in the neighbourhood.

Below we present a theorem that further strengthens our claim of anonymity.

Theorem 3: *Let S denote the common neighbourhood of k compromised nodes that are colluding. Assume that a unicast communication between two uncompromised nodes in S is heard by the k colluding nodes:*

a) If $|S| > 1$, then there is no way for the k colluding nodes to identify the sender/receiver of the message.

b) The probability that the k compromised colluding nodes will guess the sender correctly is, $1/|S|$.

c) Given that each parameter used by CAS is a K-bit integer, the probability that the k compromised colluding nodes are able to guess a parameter is $1/2^K$, and the probability that they can defeat the scheme by forging a valid sender and a receiver pseudonym for a communication in the neighbourhood is $1/2^{4K}$.

Proof: a) The common neighbourhood is formed by the k nodes sampling only the packets accessible to all of them. Each uncompromised SN u in the common neighbourhood uses the keyed hash function $h_{k_{Bu}}$, the secret seed a_{Bu}, and the sequence number, seq_{uv}, for creating the anonymous sender ID, and $h_{k_{uv}}$, a_{uv} and seq_{uv} to create the receiver ID, to communicate with the corresponding neighbour v, which is on its path towards the BS. The colluding SNs in the neighbourhood do not know the values of the keys nor the other parameters being used for generating the anonymous IDs. Hence, they have no means of identifying the source or the destination of the communication. As a result, if $|S| = 1$, the k nodes shall be able to identify the communication is from the only node in S. However, if $|S| > 1$, the k nodes cannot identify the sender/receiver of the message, as they cannot infer the pseudonym ranges.

b) Each compromised node knows only the parameters that each non-compromised node in the neighbourhood shares with it. Together the k compromised nodes only know the parameters that the non-compromised nodes share with them for cluster or mutual communication, this is the only combined knowledge the compromised nodes possess. Given this knowledge they can in no way guess the parameters of the SNs communicating nor verify the sender pseudonym or receiver pseudonym of a captured packet. Thus, in a given common neighbourhood, if there are $|S|$ common nodes, then the probability that the node guessed as the sender by the compromised nodes is the right sender, is indeed $1/|S|$.

c) Given that a parameter is a K-bit integer, since the colluding adversaries have no idea of the value of the parameter, the only option available is to guess the value. The probability that the guessed value is the correct value is obviously only $1/2^K$. For the colluding nodes to correctly generate the sender and receiver pseudonyms for a communication between two nodes u and v in the common neighbourhood (final destination of the message is the BS), they have to correctly guess the values of the four parameters, the hashed keys k_{Bu} and k_{uv}, and the seeds a_{Bu} and a_{uv}. Each of the four values are mutually independent from each other; hence, can be chosen in 2^K ways each. Hence, the probability

that the colluding nodes may be able to generate a valid source and destination pseudonym for a communication in the common neighbourhood by guessing the values of the parameter is $1/2^{4K}$. It is also easy to see that the probability of generation of just one pseudonym correctly is $1/2^{2K}$. Thus, the likelihood of successful forgery in this case is highly improbable.

The CAS ensures that when two non-compromised SNs are communicating in a neighbourhood consisting of compromised nodes that are colluding, there is no way that the nodes can identify the sender and receiver of the communication. Further, given the common knowledge that the SNs use hash function to generate the IDs, it is still very difficult for the compromised nodes to identify the communication, or even generate packets that would appear authentic to a receiver. As the compromised nodes have no knowledge of the mutual key used in the hash function nor do they know the value of the random numbers used to seed the hash function. Hence, the CAS is guaranteed to be completely anonymous.

6.6 Memory and computation requirements

In this subsection, we shall illustrate the memory required at each node of a given WSN to store the parameters needed to communicate anonymously in it neighbourhood and with the BS. We also identify the computation requirements at each node to identify if the packet is destined for it or otherwise. Consider a K-bit pseudonym space, and the neighbourhood size of a node upper bounded by M. Each node u, stores the four parameters, k_{Cu}, a_{Cu}, b_{Cu} and seq_{Cu} for creating anonymous ID when it is the CH. Further, u has to store the two variables, k_{Bu} and a_{Bu}, to generate the sender ID and for each of its neighbour v, u stores three parameters, k_{uv}, a_{uv} and seq_{uv}, for pairwise anonymous sender and receiver ID generation. It also has to store the parameters required to identify cluster communication originating from any of its neighbours v performing the role of the CH. Hence, the four parameters, k_{Cv}, a_{Cv}, b_{Cv} and seq_{Cv}.

Hence, the total memory requirement of a node is, $4K + 2K + 3KM + 4KM = 6K + 7KM$. For instance, for an ID space of 64 bits, in a CWSN containing 1000 nodes network, and an average neighbourhood size of 100 nodes. The memory requirement shall be, $6 \times 64 + 7 \times 64 \times 100 = 45,184$ bits = 5648 bytes = 5.5 KB. Assuming 2 bytes for index, the total memory requirement is 5.7 KB. For a CWSN with 10,000 nodes and a neighbourhood size of 1000 nodes, the memory requirement is, $6 \times 64 + 7 \times 64 \times 1000 = 448,384$ bits = 56, 048 bytes = 54.7 KB, with 2 bytes for index, the total memory requirement is 56.7 KB. There is no extra code added to the code base, as the same code that is used to generate MAC for messages may be used for generating the pseudonyms.

In this paper, we assume that the underlying block cipher used for the hash function (either CBC-MAC or HMAC) is skipjack (Brickell et al., 1993). In the CAS, the sender has to generate two pseudonyms. If we assume the use of skipjack for the underlying cipher block generation, generating a 64 bits block cipher takes less than 0.42 ms (Karlof et al., 2004) on a mica mote (Hill et al., 2000). Hence, for two pseudonyms in a packet, the total time is less than 0.84 ms. Most SNs in a typical WSN transmit a packet per minute, hour, or day. In such a scenario, the CAS serves to provide a pseudonyms generation mechanism with acceptable latency per packet. An SN receiving the packet has to be able to identify if it is the intended receiver of the packet, and also the correct key to use for decrypting the packets. For this, the receiver has to use the index prepended to the receiver ID to index into the pseudonym table and obtain the sequence number, the stored seed value and the mutually shared hash key. Then, using the mutually shared hash key, it shall obtain the hash value and compare with the value of the receiver ID in the received packet. If there is a match, then the receiver knows that the packet is intended for it. This requires indexing into the table, a $O(1)$ operation, and the hash value calculation, which depends on the kind of hash algorithm used. Use of the skipjack as the underlying block cipher for the pseudonym generation process shall take 0.42 ms; the comparison can be done in constant time. A node that is not the intended receiver of the message has to follow a similar process and discard the packet when the receiver ID that it calculates does not match the one in the received packet.

7 Discussion

In this section, we do a comparative analysis of the SAS and the CAS protocols. In terms of memory requirements, the CAS performs better; it has a memory saving of more than 230% in comparison to the SAS. Also, the CAS does not require any extra code size, the code that is used for MAC generation can be reused for pseudonym generation. In terms of computation, the SAS is faster than the CAS, as it requires only a constant time for indexing and comparison. However, the CAS has a higher computation requirement, as generation of a pseudonym is dependent on the size of the cryptographic block used, and the complexity of the underlying cryptographic keyed hash function used, which is not a constant running time algorithm. In CAS, each node receiving a packet has to generate a receiver pseudonym at its end and compare it with the receiver pseudonym in the packet to identify if it is the intended receiver. This results in the CAS having a higher computation requirement than SAS for each packet, in general. However, the computation and time requirements for the CAS are well within reasonable bounds as shown in Section 6.6.

We believe that the CAS is computationally more secure than the SAS. The probability of an adversary being able to successfully forge a pseudonym in the CAS is lower than the probability of similar success of forgery in the SAS. This is because, to forge the pseudonyms in CAS, the adversary has to actually identify the parameters that form the input to the keyed hash function and also the current sequence number value for the given communication between two SNs, which is computationally difficult as proved in Theorem 3. In addition, the probability that the adversary guesses the correct value for either pseudonym (sender/receiver) is extremely low as illustrated in the same theorem.

In the SAS, on the other hand, the adversary may be able to identify the range of the communicating SNs in its neighbourhood by capturing several transmitted packets. By observing and analysing these packets in its neighbourhood

the adversary may be able to create partial pseudonym subranges for the corresponding communicating nodes. However, we note here that this requires intensive analysis and also capture of significant number of packets. Once the adversary has some knowledge of the subranges used in its neighbourhood, it can generate and transmit fake packets, with forged sender and receiver pseudonyms. We do not quantitatively analyse the possibility of success of such an endeavour; however, it suffices to say that the possibility, although small, does exist. Another possible means of forgery in SAS is that an adversary may be able to inject false packets in the network, using the same sender and destination ID as that of a previous packet it has captured in the neighbourhood. This kind of a replay attack can, however, be contained by extending the SAS to use a monotonically increasing sequence number in any mutual communication between two communicating nodes.

Despite the chance of a possible forgery, we note here that in any event of forgery by one or more compromised nodes, the identity of any uncompromised communicating node(s) in the network is guaranteed to remain uncompromised in either of our schemes. Thus, our schemes continue to guarantee complete anonymity to uncompromised communicating nodes in a CWSN, even in the event of existence of malicious nodes in the neighbourhood.

8 Conclusions and future work

In this paper, we have discussed the requirements for anonymity in a CWSN. We have proposed two memory and computation efficient anonymity schemes that preserve node identity and privacy and ensure complete anonymity. From the analysis, we observe that the first scheme requires lesser computation while the second one is better at memory utilisation. Both the schemes can be easily implemented and used in a WSN environment without much resource penalty and are useful for anonymous communication. A WSN testbed implementation comparing the efficacy of the two schemes in different scenarios is an exciting future direction of work.

Acknowledgement

This research was supported in part by ARO grant W911NF 04-1-0385 and NSF grants CNS-0524736 and CCF 0431167. The information reported here does not reflect the position or the policy of the federal government.

References

Akyildiz, I., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'Wireless sensor networks: a survey', *Computer Networks*, Vol. 38, No. 4, pp.393–422.

Bandyopadhyay, S. and Coyle, E. (2003) 'An energy efficient hierarchical clustering algorithm for wireless sensor networks', *22nd Conference of the IEEE Communication Society, INFOCOM*, Vol. 3.

Blom, R. (1985) 'An optimal class of symmetric key generation systems', *EUROCRYPT 84, Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, France, pp.335–338.

Bellare, M., Canetti, R. and Krawczyk, H. (1996b) 'Keying hash functions for message authentication', *Advances in Cryptology-CRYPTO '96, Lectures Notes in Computer Science*, Vol. 1109, pp.1–15.

Bellare, M., Guérin, R. and Rogaway, P. (1995) 'XOR MACs: new methods for message authentication using finite pseudo-random functions', *Advances in Cryptology-CRYPTO '95, Lectures Notes in Computer Science*, Vol. 963, pp.15–28.

Bellare, M., Kilian, J. and Rogaway, P. (1994) 'The security of cipher block chaining', *Advances in Cryptology-CRYPTO '94, Lectures Notes in Computer Science*, Vol. 839, pp.340–358.

Bellare, M., Kilian, J. and Rogaway, P. (2000) 'The security of the cipher block chaining message authentication code', *Journal of Computer and System Sciences*, Vol. 61, No. 3, pp.362–399.

Black, J., Halevi, S., Krawczyk, H., Krovetz, T. and Rogaway, P. (1999) 'UMAC: fast and secure message authentication', *19th Annual International Cryptology Conference (CRYPTO'99)*, Santa Barbara, USA, Vol. 1666, pp.216–245.

Brickell, E.F., Denning, D.E., Kent, S.T., Mahler, D.P. and Tuchman, W. (1993) 'SKIPJACK review', *Interim Report*, 28, July, p.8.

Chatterjee, M., Das, S. and Turgut, D. (2002) 'WCA: a weight based clustering algorithm for mobile ad hoc networks', *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, Vol. 5, pp.193–204.

Delakouridis, C., Kazatzopoulos, L., Marias, G.F. and Georgiadis, P. (2005) 'Share the secret: enabling location privacy in ubiquitous environments', *Lecture Notes in Computer Science*, Vol. 3479, pp.289–305.

Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J. and Khalili, A. (2005) 'A pairwise key predistribution scheme for wireless sensor networks', *ACM Transactions on Information Systems Security*, Vol. 8, No. 2, pp.228–258.

Fiestel, H. (1973) 'Cryptography and computer privacy', *Scientific American*, Vol. 228, No. 5, pp.15–23.

Goldreich, O., Goldwasser, S. and Micali, S. (1986) 'How to construct random functions', *Journal of the ACM*, Vol. 33, No. 4, pp.792–807.

Hill, J., Szewczyk, A., Woo, A., Hollar, S. Culler, D., Pister, K. (2000) 'System architecture directions for networked sensors', *Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November, pp.93–104.

Ibriq, J. and Mahgoub, I. (2004) 'Cluster-based routing in wireless sensor networks: issues and challenges', *Proceedings of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS)*.

Karlof, C. and Wagner, D. (2003) 'Secure routing in wireless sensor networks: attacks and countermeasures', *Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, Vol. 1, Nos. 2–3, pp.293–315.

Karlof, C., Sastry, N. and Wagner, D. (2004) 'TinySec: a link layer security architecture for wireless sensor networks', *Second International Conference on Embedded Networked Sensor Systems, SenSys'04*, Baltimore, MD, pp.162–175.

Kong, J. and Hong, X. (2003) 'ANODR: ANonymous On Demand Routing with untraceable routes for mobile ad-hoc networks', *Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc*, pp.291–302.

Liu, D. and Ning, P. (2003) 'Establishing pairwise keys in distributed sensor networks', *Tenth ACM Conference on Computer and Communications Security*, pp.52–61.

Menezes, A., Oorschot van, P. and Vanstone, S. (1996) 'Handbook of applied cryptography', *A CRC Press Company*, Boca Raton, FL.

Polastre, J., Szewczyk, R. and Culler, D. (2005) 'Telos: enabling ultra-low power wireless research', *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors, IPSN/SPOTS '05*, pp.364–369.

Poosarla, R., Deng, H., Ojha, A. and Agarwal, D.P. (2004) 'A cluster based secure routing scheme for wireless ad hoc networks', *IEEE, IPCCC*, pp.171–175.

Preneel, B. and Oorschot van, P. (1995) 'MDx-MAC and building fast MACs from hash functions', *Advances in Cryptology-CRYPTO '95, Lectures Notes in Computer Science*, Vol. 963, pp.1–14.

Stinson, D.R. (2002) 'Cryptography: theory and practice', *A CRC Press Company*, Boca Raton, FL.

Wegman, M. and Carter, L. (1981) 'New hash functions and their use in authentication and set equality', *Journal of Computing System Science*, Vol. 22, pp.265–279.

Wu, X. and Bhargava, B.K. (2005) 'AO2P: ad hoc on-demand position-based private routing protocol', *IEEE Transactions on Mobile Computation*, Vol. 4, pp.335–348.

Younis, O. and Fahmy, S. (2004) 'Distributed clustering in ad hoc sensor networks: a hybrid, energy-efficient approach', *23rd Conference of the IEEE Computer and Communications Societies*, pp.629–640.

Ye, F., Zhong, G., Lu, S. and Zhang, L. (2003) 'PEAS: a robust energy conserving protocol for long-lived sensor networks', *International Conference on Distributed Computing Systems (ICDCS)*, pp.28–29.

Zhu, S., Setia, S. and Jajodia, S. (2003) 'LEAP: efficient security mechanisms for large-scale distributed sensor networks', *Tenth ACM Conference on Computer and Communications Security*, pp.62–72.

Zhu, B., Wan, Z., Kankanhalli, M.S., Bao, F. and Deng, R.H. (2004a) 'Anonymous secure routing in mobile ad-hoc networks', *IEEE 29th International Conference on Local Computer Networks (LCN'04)*, pp.102–108.

Zhu, S., Setia, S., Jajodia, S. and Ning, P. (2004b) 'An interleaved hop-by-hop authentication scheme for filtering false data injection in sensor networks', *IEEE Symposium on Security and Privacy*, pp.259–274.