4-2009

Achieving better privacy protection in wireless sensor networks using trusted computing

Yanjiang YANG
Singapore Management University, yjyang@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Jianying ZHOU
Institute for Infocomm Research, Singapore

Ying QIU
Institute for Infocomm Research, Singapore

DOI: https://doi.org/10.1007/978-3-642-00843-6_33

Information Security Practice and Experience: 5th International Conference, ISPEC 2009
Xi'an, China, April 13-15, 2009: Proceedings. pp 384-395.
http://dx.doi.org/10.1007/978-3-642-00843-6_33

Achieving Better Privacy Protection in Wireless Sensor Networks Using Trusted Computing

YANG, Yanjiang; DENG, Robert H.; Zhou, Jianying; QIU, Ying

Abstract

A wireless sensor network (WSN) is an ad-hoc wireless network composed of small sensor nodes deployed in large numbers. Sensor nodes are usually severely resource limited and power constrained. They are often deployed in accessible areas and interact closely with their physical surroundings and people. Security enforcement in WSNs is thus a challenging task. In this paper we propose a clustered heterogeneous architecture for WSNs, where high-end cluster heads are incorporated, and the cluster heads are further equipped with trusted computing technology (TC). As such, the cluster heads act as trusted parties, and are expected to help effectively address privacy issues in WSNs. As concrete examples, we discuss in details how user query privacy and source location privacy can be better protected in such TC-enabled WSNs.

1 Introduction

Emerging as an important new technology, wireless sensor networks (WSNs) have a wide range of potential applications, especially in the realtime monitoring scenarios, such as battlefield surveillance, wildlife tracking, healthcare monitoring, emergency response and earthquake monitoring. A WSN consists of a large number of sensor nodes collecting environmental data. The sensor nodes communicate wirelessly and self-organize after being deployed in an ad hoc manner. The nodes are usually severely constrained in computation, storage, communication and power resources.

When deployed in critical applications, mechanisms must be in place to secure a WSN. Security issues associated with WSNs can be categorized into two broad classes [26]: content-related security, and contextual security. *Content-related security* deals with security issues related to the content of data traversing the sensor network such as data secrecy, integrity, and key exchange. Numerous efforts have recently been dedicated to content-related security issues, such as secure routing [17, 18, 24, 37], key management and establishment [5, 8, 10, 27, 28, 29, 30, 39, 45], access control [43, 46], and data aggregation [6, 16, 31, 38]. In many cases, it does not suffice to just address the content-related security issues . Suppose a sensitive event triggers a packet being sent over the network; while the content of the packet is encrypted, knowing which node sends the packet reveals the location where the event occurs. *Contextual security* is thus concerned with protecting such contextual information associated with data collection and transmission.

It is commonly acknowledged that the resource-constrained nature of sensor nodes makes security enforcement in WSNs a challenging task. The majority of the above mentioned efforts attempted to solve security issues in *homogeneous* WSNs where all sensor nodes have the same capabilities. However, both theoretical and empirical studies have concluded that homogeneous WSNs are not scalable. Through a theoretical analysis it is shown in [13] that the throughput of

each sensor node decreases rapidly as the numbers of nodes increases; in addition, it is demonstrated via simulations in [9] that, as the traffic becomes heavy, the overhead due to routing and control consumes a large portion of the available bandwidth.

Our Contribution To improve the effectiveness of security enforcement, we present a trusted computing (TC)-enabled clustered heterogeneous WSN architecture, composed of not only resource constrained sensor nodes, but also a number of more powerful high-end devices acting as *cluster heads*. Compared to sensor nodes, a high-end cluster head has higher computation capability, larger storage, longer power supply, and longer radio transmission range, and it thus does not suffer from the resource scarceness problem as much as a sensor node does. A distinct feature of our heterogeneous architecture is that cluster heads are equipped with trusted computing (TC) technology, and in particular a TCG-compliant TPM (Trusted Platform Module) [41] is embedded into each cluster head. According to the TCG specifications, TPM is a tamper-resistant, self-contained secure coprocessor, capable of performing cryptographic functions. A TPM attached to a host establishes a trusted computing platform that provides *sealed storage*, and measures and reports the integrity state of the platform. More discussions on TCG/TPM are provided in Section 2. In our architecture, the TC-enabled cluster heads act as online *trusted parties*; security enforcement is thus expected to be substantially simplified and improved. We further substantiate the above assertion by demonstrating how trusted cluster heads help to provide elegant solutions for two important contextual security problems, user query privacy [7] and source location privacy [26, 32, 34, 36], both under a strong adversarial model.

Organization In Section 2, as preliminary knowledge we give a brief overview of the TCG/TPM. We then present a TC-enabled heterogeneous architecture for WSNs in Section 3. In Section 4, we show how to efficiently achieve user query privacy and source location privacy in TC-enabled WSNs. We review related work in Section 5. Section 6 concludes the paper.

2 Preliminaries: Overview of TCG/TPM

The latest effort in trusted computing is represented by the Trusted Computing Platform specifications defined by Trusted Computing Group (TCG) [41]. The specifications aim to provide hardware based roots of trust through a tamper resistant coprocessor, Trusted Platform Module (TPM). A TPM is attached to a host machine and acts as the *root of trust* of the host platform, given its tamper resistance property. TPM is capable of performing cryptographic functions such as random number generation, SHA1 hash function, and RSA encryption and digital signature.

A core functionality provided by TPM is integrity measurement and storage, and reporting of the state of the host platform. Integrity measurement metrics that represent the state of the underlying platform are stored by a set of Platform Configuration Registers (PCRs), internal to TPM. Each PCR value is a 20-byte SHA1 hash digest of a number of measured platform integrity metrics. Altogether the PCRs record the integrity status of the host platform from booting to OS loading to loading of the protected applications. An update to a PCR value is through what is termed *extending the PCR*, which is described as

$$PCR[i] \leftarrow SHA1(PCR[i]||newly\ measured\ value)$$

where i is the index of the PCR being updated. Since a PCR value is a *digest* of the platform state (which results from a series of state altering events), it is meaningless by itself. The data that complements PCRs in providing semantics is Stored Measurement Log (SML). The SML stores the complete event history for all the PCRs, and each PCR has corresponding entries in the SML that records the series of events leading to the current PCR value . The SML is stored unprotected outside the TPM. This however does not compromise integrity as the corresponding digests are stored in PCRs, and "extending a PCR" can only be performed by TPM protected capabilities. The PCR values, together with the corresponding entries of the SML, are used as evidence to attest to the current state of the host platform.

Upon request, TPM can report the state of its underlying platform to a remote challenging entity through *attestation*. In particular, TPM has a number of key pairs called Attestation Identity Keys (AIKs), which are used as aliases of the unique Endorsement Key (EK). The attestation protocol proceeds as follows. (1) The challenging entity issues a challenge message, indicating that it wants to inspect one or more PCR values. (2) A Platform Agent collects the related SML entries corresponding to the requested PCR values. (3) TPM sends the Platform Agent the requested PCR values signed by the private key of an AIK. (4) The Platform Agent sends the signed PCR values, together with the relevant SML entries and the AIK certificate to the challenging entity. (5) The challenging entity verifies the replied data - the AIK certificate is validated, the measurement digest is computed from the SML entries and compared with the signed PCR values.

Another security function provided by TPM is *Sealed Storage*, which encrypts sensitive data with integrity measurement values. In particular, the data to be protected is encrypted/sealed together with one or more PCR values. Subsequently, TPM releases an encrypted data only if the current PCR values match those stored during encryption. In other words, if the state of a platform is modified, the encrypted data in the sealed storage under that state will not be decrypted/unsealed. The encryption key is protected either by the Storage Root Key (SRK) internal to TPM, or by a key protected by the SRK.

3 A TC-enabled Heterogeneous Architecture for WSNs

3.1 The Architecture

We partition a WSN into a number of *clusters*. A high-end device is placed into each cluster, acting as the *cluster head*. In contrast to sensor nodes, high-end cluster heads have relatively higher computation capability, larger storage size, and longer radio range. They also have longer power supply, and in some circumstances they can even be line-powered, e.g., when a WSN is deployed to monitor a building, the cluster heads can easily tap on the electricity lines to get power supply. Therefore unlike sensor nodes, cluster heads do not drastically suffer from the resource scarceness problem. The introduction of high-end cluster heads into a WSN makes the once homogeneous network *heterogeneous*. The general heterogeneous architecture is depicted in Figure 1.

Downlink communication (from base station to sensor nodes) and uplink communication (from senor nodes to base station) in the architecture are asymmetric. Messages broadcast by the base station can directly reach sensor nodes, whereas messages sent by a sensor node need to be forwarded by its corresponding cluster head. As a result, uplink communication follows a hierarchical manner and consists of intra-cluster and inter-cluster communications, respectively. At the level of *intra-cluster communication*, a cluster head acts as a gateway for the sensor nodes within the

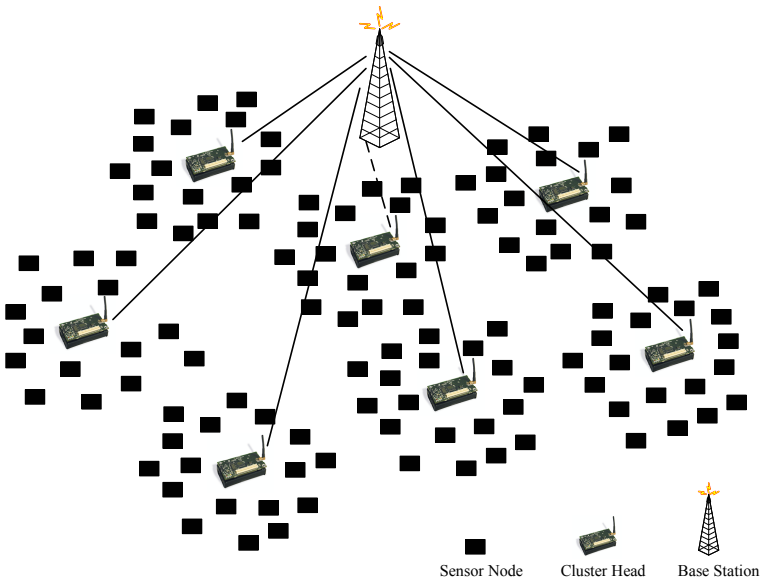

Figure 1: Heterogeneous Wireless Sensor Network

cluster; a sensor node can reach the cluster head directly, or through a *short* multi-hop channel. A fundamental criteria for cluster partition is that the number of intermediary hops of a multi-hop channel must be small. *Inter-cluster communication* is concerned with communication among cluster heads and the base station. Since cluster heads are not severely constrained by resources and power supply, inter-cluster communication does not suffer from the limits upon sensor nodes, and it can utilize more advanced communication infrastructure, e.g. 802.11 or even wired network, depending on applications. Consequently, the heterogeneous architecture is expected to enormously improves the overall system performance and lifetime of the network.

A feature distinguishing our architecture from other similar heterogeneous ones (e.g., [19, 20, 44]) is that we equip each cluster head with a TPM, so as to simplify and improve security enforcement in WSNs. The rationale is that under the auspices of TPM, cluster heads can act as online *trusted parties* in enforcing security mechanisms. Relevant entities, e.g., the base station or users who query the network (see Section 4.1), can ascertain the trustfulness of cluster heads by means of attestation. We notice that the authors of [14, 23] also discussed the idea of applying trusted computing technology to equipping more powerful cluster heads within WSNs, and there should be no distinction in the network architecture between ours and theirs. However, in their proposals, sensor nodes are responsible for verifying the platform state of the cluster heads through attestation, whereas our proposal is that sensor nodes never challenge their cluster heads for attestation (we believe they actually do not afford to do so), and they simply trust their respective cluster heads. Our argument relies on the fact that it is the end users (or the owner) of the network who should be concerned with the trustfulness of the cluster heads and the network.

Advantages The TC-enabled heterogeneous architecture has a number of advantages.

1. Partitioning a network into clusters makes management of the network scalable. Incorporating high-end cluster heads simplifies management of sensor nodes, as the base station can delegate management and adminstration of sensor nodes to their respective cluster heads. This enables easy

node dynamics such as node join and node departure, because node dynamics within a cluster are managed by the corresponding cluster head, which is closer to the sensor nodes.

2. High-end cluster heads help to reduce the amount of data that must traverse the network, thereby enhancing the over system throughput. For example, a cluster head is aware of the topology of the cluster where it resides, thus it can easily take charge of routing information within the cluster. Sensor nodes no longer need to discover routing paths by themselves, which is an energy-consuming process; they simply obtain routing information from their cluster head.

3. Cluster heads amortize the workload of security enforcement, preventing the base station from becoming the system bottleneck (virtually all existing work depends on the base station for security enforcement). Further, the use of multiple cluster heads avoids the base station as the single point of vulnerability as in homogeneous networks.

4. Cooperation of cluster heads makes it possible to provide resilient security solutions, e.g., using threshold cryptographic techniques.

3.2 Configuration of Cluster Head

Depending on application scenarios, hardware capabilities of cluster head may vary from that comparable to a bluetooth device to that of a high end PDA. The TCG is currently working on the specifications for Trusted Mobile Platforms, whose core element is Mobile Trusted Module (MTM), similar to TPM for PCs [42]. Prototype implementation of MTM were already available (e.g., [40]). Hence, we believe that there exists no technical barrier to implement our envisioned TC-enabled cluster head, although for the moment no off-the-shelf such devices are available.

A trusted computing platform can be implemented as a restricted system or an open system. The former runs a small set of protected applications, while the latter runs both protected and unprotected applications. We choose to design the cluster head as a restricted trusted computing platform due to its specialized functionality and application in WSNs. A reference platform configuration of cluster head is shown in Figure 2. The platform runs the sole ClusterH application. At

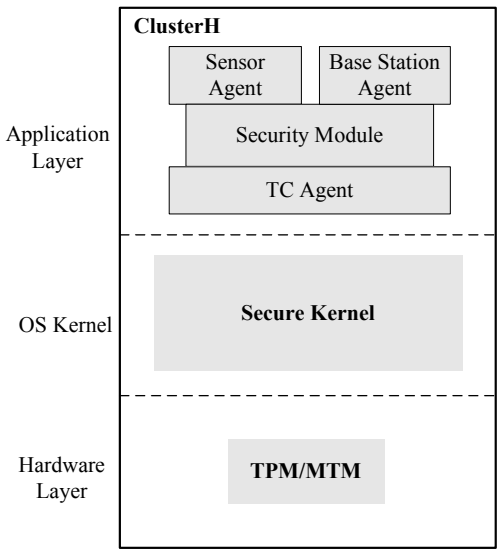

Figure 2: A Reference Cluster Head Platform Configuration

the application layer, the ClusterH program includes four main components. The *trusted computing agent* (or TC agent) is the interface that accesses the functionalities provided by the underlying TPM/MTM such as sealed storage and integrity reporting mechanisms. The *security module* is dedicated to implementing the designated WSN security mechanisms (e.g., the algorithms to implement user query privacy and source location privacy). The *sensor agent* is the communication interface with sensor nodes, while the *base station agent* is the interface with the base station or other cluster heads. The OS layer implements the secure kernel, bridging between the application layer and the hardware layer. The hardware layer includes a TCG-compliant TPM/MTM, providing hardware based root of trust.

4 Addressing Privacy Issues in Wireless Sensor Networks

Equipped with TPM/MTM, the secure kernel and the ClusterH software, cluster heads act as online trusted parties. To show the effect of the trusted cluster heads on security enforcement, in this section we present solutions to two important contextual security problems in WSNs: user query privacy and source location privacy. Compared to existing solutions in [7, 26, 32, 34, 36], our schemes achieve better privacy and higher efficiency.

Adversary Model For the schemes presented below, we assume a *global eavesdropper* who eavesdrops on the entire network. In particular, the adversary is able to watch all the traffic traversing the sensor network. An adversary of this nature poses a great threat, especially to contextual security in WSNs.

The global adversary in our consideration is much stronger than the *local eavesdropper* assumed by other work such as [7, 26, 34, 36]. A local eavesdropper only has knowledge on the sensor node it stays with, and it can only trace communication hop by hop. We must stress that the privacy offered by the schemes in [7, 26, 34, 36] is immediately broken under the global adversary, because through global eavesdropping the adversary can always know which node(s) initiates the targeted data transmission.

4.1 Achieving User Query Privacy

4.1.1 Problem Statement

WSNs are often deployed to provide services to other users than the network owner [7]. Users are allowed to query a network to get sensed data from particular areas. In such a scenario, a user may wish to protect her "areas of interest" from being disclosed to other users or even the network owner. User query privacy is thus concerned with the following problem: suppose a user queries the network, intending to get the sensed data in cluster c_i, a user query privacy scheme ensures that the user ends up getting the desired data, but the adversary does not learn c_i by observing the communication.

4.1.2 Our Algorithm

Network Model We support roaming users querying a wireless sensor network. The network follows the heterogeneous architecture proposed earlier: the whole network is partitioned into a set of n clusters, c_1, c_2, ..., c_n, where c_i is the identifier of the ith cluster; each cluster is grouped

around a TC-enabled cluster head and we denote ch_i the cluster head in c_i. A user who desires to query the network first contacts the nearest cluster head within her proximity, through which she will issue queries. This cluster head is called *access point*. Taking Figure 3 as an example, ch_1 of c_1 is the access point for the user.

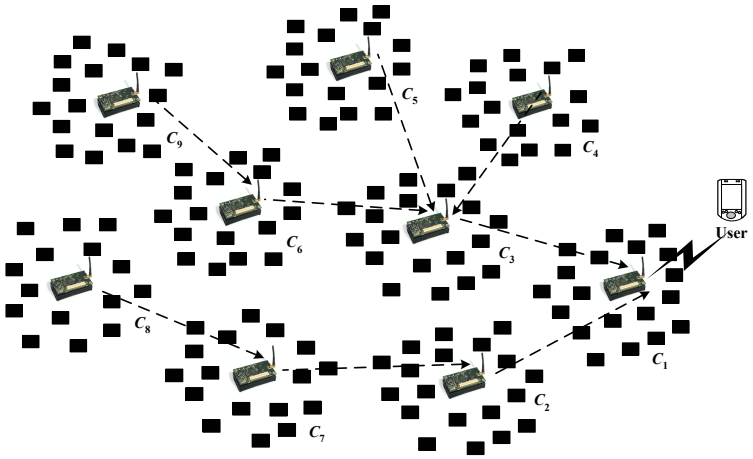

Figure 3: A User Uses the Cluster Head of c_1 as Her Access Point

Once the access point is determined, all the other cluster heads need to dynamically establish routing pathes to the access point. As cluster heads do not drastically suffer from limited resources, some well studied routing techniques for networking could be applied here for the purpose of routing path construction. It is our recommendation to use the single-path routing presented in [26] due to its high efficiency and simplicity. The single-path routing uses a greedy algorithm where each node chooses one of its neighboring nodes that is nearest to the sink (i.e., the access point in our case) as its *forwarding node* in data transmission, and the node itself is a *dependent node* of the forwarding node. As a result, the routing pathes form a tree rooted at the access point. For the example in Figure 3, the routing pathes are shown in dotted lines. After the routing pathes are formed, every cluster head knows which node is its forwarding node, and which nodes are its dependent nodes.

Assumption To obtain services from a WSN, a user is assumed to have a certain means to authenticate to the WSN. Also, a TC-enabled cluster head can authenticate to users using the AIK of its embedded TPM/MTM. Therefore, the access point and the querying user can accomplish mutual authentication, based on which we assume the two entities share a secret key for semantic secure symmetric encryption. Further, it is also easy for each cluster head to get this secret key from the access point, since they can clearly authenticate each other with the help of their respective AIKs. We denote $\mathcal{E}_{ch_i}(.)$ and $\mathcal{D}_{ch_i}()$ the encryption and decryption, respectively, by ch_i using this secret key. We also assume each cluster head shares a secret *cluster key* (for semantic secure symmetric encryption) with all sensor nodes in its cluster. Several well studied key exchange schemes [29, 30] can achieve this objective. $\mathcal{CE}_{c_i}(.)$ and $\mathcal{CD}_{c_i}()$ denote the encryption and decryption, respectively, using the cluster key of c_i.

Algorithm Overview A straightforward way to achieve query privacy is that every cluster head sends encrypted data to the access point, who then forwards only the data desired by the user.

This however unnecessarily wastes communication bandwidth among cluster heads. Even for this straightforward method, we should still be very cautious not to leak information about the queried cluster from the size of the data returned to the user. More specifically, clusters normally have different number of sensor nodes, so data from different clusters are likely to have different lengthes. Let us suppose the data from each sensor node forms a packet for simplicity. The total number of packets from a cluster equals the number of sensor nodes. Without privacy treatment, the number of packets eventually returned to the user by the access point would clearly indicate the cluster from which the data originates. A method to fix this problem is that regardless of which cluster is queried, the access point returns a fixed-number of packets, corresponding to the biggest cluster size. We use l to denote this number thereafter. For a cluster whose size is smaller than l, *dummy* packets are generated.

The basic idea of our approach is to only transmit packets from the queried cluster to the access point. However, to hide the target cluster, all the remaining clusters have to generate *fake* data transmission of the same pattern as that of the queried cluster. To better convey our idea, let us continue to use Figure 3 as an example. Suppose c_6 is the target queried cluster. The data transmission starts with ch_9, ch_8, ch_5 and ch_4, who are the tails of the respective routing pathes. Let us however only explain the transmission involving the target cluster c_6, as others follow the same manner. The cluster head ch_9 generates and sends l dummy packets $\mathcal{E}^l_{ch_9}(dummy\_data)$ to ch_6. We use $\mathcal{E}^l_{c_i}(data)$ to denote l packets of encryption of $data$ by ch_i. ch_6 knows that its cluster is being queried, so he discards the data from ch_9, and generates and passes $\mathcal{E}^l_{ch_6}(sensed\_data)$ to ch_3. At this point of time, ch_3 may have already received $\mathcal{E}^l_{ch_5}(dummy\_data)$ and $\mathcal{E}^l_{ch_4}(dummy\_data)$ from its dependent nodes ch_5 and ch_4, respectively. If not, ch_3 waits until it gets the data from all its dependent nodes. The reason why a node does not proceed until receives from all its dependent nodes is to avoid disclosure of information from timing patterns. Then ch_3 decrypts $\mathcal{E}^l_{ch_6}(sensed\_data)$ and re-encrypts $sensed\_date$ to yield $\mathcal{E}^l_{ch_3}(sensed\_data)$, it then passes the data to the access point ch_1. The access point ch_1 waits until it gets data from all other routing pathes, and then decrypts $\mathcal{E}^l_{ch_3}(sensed\_data)$ and re-encrypts $sensed\_date$ to yield $\mathcal{E}^l_{ch_1}(sensed\_data)$. Finally, ch_1 sends $\mathcal{E}^l_{ch_1}(sensed\_data)$ to the querying user as the query result.

In our approach, every cluster head sends out l packets. Due to the semantic security of encryption, re-encryptions of the same data are not distinguishable. Therefore, the adversary watching the network cannot tell if the l packets sent out by a cluster head originate from the cluster head itself or from its dependent nodes.

Algorithm Details A complete description of the algorithm in pseudo codes is shown in Algorithm 1, where u denotes the querying user and ap denotes the access point.

To start, the user contacts the access point by sending a *hello* message, including a *nounce* that will be used in the ensuing attestation process (Step 1). The access point then informs all the other cluster heads to form routing pathes using the method described earlier (Step 2). Before sending a query, the user must have assurance of the trustfulness of the cluster heads. This is achieved by means of attestation (Step 3). Note that it is unnecessary for the user to check the status of all cluster heads, which is quite expensive; it suffices to adopt the strategy of "chained attestation" along the established routing pathes. In particular, referring to Figure 3, u only verifies the access point: ch_9 is verified by ch_6; ch_4, ch_5 and ch_6 are verified by ch_3; ch_8 is verified by ch_7 who is in return verified by ch_2; ch_2 and ch_3 are verified by the access point.

Algorithm 1 Achieving User Query Privacy.

1: $u \rightarrow ap$: *hello*
2: $ap \leftrightarrow \{ch_i\}_i$: establish routing pathes.
3: $u \leftrightarrow ap$: attestation
4: $u \rightarrow ap$: $e_u = \mathcal{E}_u(c)$ /*c *is the identifier of the target cluster**/
5: $ap \rightarrow \{ch_i\}_i$: e_u
6: **for** EACH ch_i **do**
7: $c = \mathcal{D}_{ch_i}(e_u)$
8: $\{sensor\} \rightarrow ch_i$: $data_{c_i} = \{\mathcal{CE}_{c_i}(sensed\_data)\}$
9: $\{dependent\_node\} \rightarrow ch_i$: $\{packets\}$ /*ch_i waits until gets packets from all its dependent nodes**/
10: **if** $ch_i \in c$ **then**
11: /*if ch_i *is cluster head of the target cluster* c **/
12: $packets = \mathcal{CD}_{c_i}(data_{c_i})$
13: $packets = head||content = \mathcal{E}_c(c)||\mathcal{E}^l_{ch_i}(packets)$
14: **else**
15: $foundqueriedcluster$ = FALSE
16: **for** *packets* from EACH *dependent_node* **do**
17: $c' = \mathcal{D}_{ch_i}(head)$
18: **if** $c' == c$ **then**
19: $content' = \mathcal{D}_{ch_i}(content)$
20: $packets = head||content = \mathcal{E}_{ch_i}(c)||\mathcal{E}^l_{ch_i}(content')$
21: $foundqueriedcluster$ = TRUE
22: break
23: **end if**
24: **end for**
25: **if** $foundqueriedcluster ==$ FALSE **then**
26: $packets = head||content = \mathcal{E}_{ch_i}(c_i)||\mathcal{E}^l_{ch_i}(dummy\_date)$
27: **end if**
28: **end if**
29: $ch_i \rightarrow forward\_node$: *packets*
30: **end for**
31: $ap \rightarrow u$: *packets*

Once attestation is successful, the user sends to the access point the query e_u, which is the encryption of the identifier c of the target cluster using the shared secret key (Step 4). The access point broadcasts the query to all other cluster heads (Step 5), each decrypting the query and knowing which cluster the user is querying (Step 7). Each cluster head then collects sensed data (encrypted using the cluster key) from the sensor nodes of its cluster (Step 8).

Before sending out l packets of data to its forwarding node (Step 29), a cluster head must wait until it receives packets from all its dependent nodes (Step 9). Afterwards, if the cluster itself is the target cluster (Step 10-13), the cluster head simply ignores the packets from its dependent nodes, and encrypts the sensed data from its cluster. Note that every set of l packets consists of *head* and *content*, where the *head* is used to inform cluster heads enroute the origin of the l packets while without decrypting the *content*. For a cluster that is not the target one (Step 14-27), the cluster head checks whether one of its dependent nodes sends in the data of the target cluster. If yes, the cluster head re-encrypts the data (Step 16-22); otherwise, the cluster head generates l dummy packets (Step 25-27). Eventually, the access point passes the l packets of the target cluster to the querying user (Step 31).

Security Analysis We can argue security of our algorithm from two aspects: data communicated and communication patterns. For the first aspect, it is somewhat straightforward to see that due to the use of semantic secure encryption, data communicated do not divulge the content of messages. More formal proof can be constructed based on simulation: define *view* of a query to be all data communicated across the network to answer the query. It can be proven that for any query q, any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{A}^*$ such that $|\Pr[\mathcal{A}(view) = f(q)] - \Pr[\mathcal{A}^*(struc) = f(q)]|$ is negligible, as long as the encryption scheme is a pseudo-random permutation, where $f(q)$ is any function on the result of query q, and *struc* is the structure of the underlying sensor network. For the second aspect, it is clear that every query results in the same communication pattern, i.e., every cluster head reads sensed data from the sensor nodes in its cluster; every cluster head sends out l packets to its forwarding node after receiving data from all its dependent nodes. Altogether, observing communication in the network does not in any way help the adversary to figure out which cluster is the query target.

4.1.3 Improvement

In the above scheme, to answer a user query all the sensor nodes are asked to send data to their respective cluster heads. This may shorten the lifetime of the network because of excess energy consumption. To mitigate this problem, We can alternatively trade off data freshness for energy efficiency, especially when queries come in at a high rate. In particular, sensor nodes *periodically* provide the sensed data to their respective cluster heads, who cache the data. The cluster heads then handle user queries using the cached data rather than collecting realtime data from the sensor nodes.

4.1.4 Comparison

The only earlier work we are aware of studying user query privacy is [7]. The two-server approach in [7] uses a routing scheme for data transmission similar to onion routing [15]. The routing path is constructed dynamically among the sensor nodes for each query. We list the comparison results between our approach and that in [7] in Table 1. To be more specific, our scheme assumes a global

	Adversary model	Sensor storage	Sensor computation	Sensor communication	User registration
Our scheme	*Global eavesdropper*	*Constant*	*Constant*	*Constant*	*No*
Scheme in [7]	*Local eavesdropper*	*Linear to the total number of users*	*Related to routing path*	*Related to routing path*	*yes*

Table 1: Comparison Results

eavesdropper, so achieving better privacy. A sensor node in our scheme only needs to store a secret cluster key, while each sensor in [7] is required to store secret data linear to the total number of users authorized to query the network. On computation and communication, each sensor node in our scheme needs to encrypt its sensed data and sends to its cluster head, or is required to relay data from other nodes in the same cluster; as the routing pathes must be short within a cluster, the computation and communication are thus constant. In contrast, computation and communication

for a sensor node in [7] depend on the routing path from the target cluster to the querying user. Finally, in [7] every authorized user is required to register to the servers, which in turn store secret data for the user's virtual name on each sensor node; sensor nodes in our scheme on the contrary have nothing to do with user registration. To conclude, the comparison results show that our scheme outperforms [7] in all aspects.

4.2 Achieving Source Location Privacy

Source location privacy is concerned with *not letting the adversary know which sensor node sends data to a sink (the base station)*. Most existing approaches, e.g., [26, 34, 36], consider local eavesdroppers who can only trace transmission hop by hop. An exception is [32], which assumes a global eavesdropper. The commonly used methods for achieving source location privacy are *random walk*, *source simulation*, or a combination of the two. The random walk method generates a random routing path from the source node to the sink for each transmission, while the source simulation method generates a number of fake source nodes simulating the actual source sensor, so as to confuse the adversary.

It appears that the random walk method is not effective under the global adversary overseeing the entire network, since the adversary always knows from which node packets originated. Our heterogeneous network architecture facilitates a much more efficient implementation of source simulation. In particular, to simulate the actual source node sending a packet to its cluster head, every other cluster head randomly chooses a sensor node in its cluster to send a fake packet. Afterwards, cluster heads send their packets to the sink. Intuitively, source location privacy can be considered as a special case of user query privacy in our architecture. In order to keep the paper compact, we omit the details of the scheme, but it should not be difficult to specify, given the earlier scheme for user query privacy.

Source simulation in our architecture is more efficient than that in [32], due to the shortened routing pathes and reduced control messages resulting from the use of high-end cluster heads.

5 Related Work

Partitioning a WSN into clusters were proposed by several authors for achieving scalability and better performance [2, 3, 11, 21]. In these schemes, one or more sensor nodes within a cluster are chosen as cluster head (i.e., homogeneous clustered WSNs). On the other hand, considering the limited capabilities of sensor nodes, studies from both the research community and the industry sector have tried to enhance network performance by incorporating a number of more powerful nodes in WSNs (i.e., heterogeneous clustered WSNs). A detailed theoretical analysis on the effect of adding powerful nodes to WSNs was given in [44]. It is concluded that only a modest number of reliable, long-range backhaul links and line-powered nodes are required to have a significant effect, and if properly deployed, heterogeneity can triple the average delivery rate and a 5-fold increase in the lifetime of a large battery-powered sensor networks. Intel has an on-going experimental effort [20] to incorporate Intel XScale® based nodes into WSNs. The experiment indicated that data traversing across a network are routed biased towards the XScale® nodes over simple sensor nodes, thereby indeed enhancing the overall system performance. [12, 35, 44] are among the work to study security issues in the homogeneous or heterogeneous clustered WSNs.

Our study of TC-enabled WSNs is motivated by the above idea of network clustering and heterogeneity, and TCG's trusted computing technology. The heterogeneous architecture we proposed differs from the existing heterogeneous networks such as [4, 20, 35, 44] in that the high-end cluster heads in our WSNs are TC-enabled. This makes our WSNs not only have improved network performance, but also greatly facilitate security enforcement, as we have demonstrated earlier.

We realized that we are not the first to propose the idea of equipping some more powerful cluster heads in a WSN with trusted computing technology. [14, 23] also discussed similar ideas. More specifically, [23] proposed lightweight attestation techniques that can help regular sensor nodes to check the status of a TC-enabled cluster head's platform, and [14] studied key establishment and management between sensor nodes and the base station, with the help of TC-enabled cluster heads. There seems no essential difference in the network architecture between our proposal and that of [14, 23]. However, the techniques proposed in [14, 23] are intended for the sensor nodes to perform attestation so as to check the platform state of the cluster heads. In contrast, in our architecture the sensor nodes never attempt to check the trustfulness of the cluster heads, and they simply trust their respective cluster heads. Our justification for this is that it is the responsibility of the end users of the network to concern about the trustfulness of the cluster heads. Other difference between our work and [14, 23] is that different security issues in WSNs were addressed.

Trusted computing technology has also been proposed to protect privacy in RFID systems (the RFID tags forming a RFID system are also quite weak in capabilities), where a TPM is attached to the RFID reader [33] in order to make the reader act according to the established privacy policies. Another work to protect RFID privacy is [22] which proposes to incorporate powerful devices into a RFID system, and the devices are designed to simulate the behavior of the RFID tags and the tags themselves are made dormant. The devices are simply assumed to be trusted, but how to make the assumption practically true is unclear. RFID systems have quite different working mechanisms from WSNs, thus the security concerns, and in turn the challenges in enforcing security, in the two are significantly different.

In this work, we focused on privacy protection in WSNs, and provided better solutions to two important privacy issues than existing proposals [7, 26, 32, 34, 36]. Another privacy issue in WSNs discussed in the literature is temporal privacy [25], which is concerned with the fact that an adversary, by simply monitoring the arrival of packets at the sink, can infer the temporal patterns of interested events. While we did not address the temporal privacy issue for lack of space, we believe our TC-enabled heterogeneous architecture is in a better position to solve the problem, because the *trusted* cluster heads should be able to perform timing synchronization more reliably and easily.

6 Conclusion and Future Work

Due to stringent resource limitations of sensor nodes, security enforcement is extremely challenging in wireless sensor networks. To solve this problem, we proposed to render a wireless sensor network heterogeneous, by incorporating TC-equipped high-end devices into clusters of the network, acting as cluster heads. We demonstrated how the TC-enabled cluster heads can effectively address privacy issues in WSNs.

This study is still in the preliminary stage. We are preparing to implement proof-of-the-concept TC-enabled WSN architecture, and further experiment with the architecture in certain real world wireless sensor network settings.

References

[1] T. Arnold, and L. V. Doorn. *The IBM PCIXCC: A New Cryptographic Coprocessor for the IBM EServer*. IBM Journal of Research and Development 48 (May 2004).

[2] S. Banerjee, and S. Khuller. *A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks*, Proc. IEEE INFOCOM'01.

[3] S. Basagni. *Distributed Clustering Algorithm for Ad-Hoc Networks*, Proc. International Symposium on Parallel Architectures, Algorithms, and Networks, 1999.

[4] M. Bohge, and W. Trappe. *An Authentication Framework for Hierarchical Ad Hoc Sensor Networks*, Proc.ACM workshop on Wireless security, WiSE'03, pp. 79 - 87, 2003.

[5] H. Chan, A. Perrig, and D. Song. *Random Key Pre-distribution Schemes for Sensor Networks*, Proc. IEEE Symposium on Security and Privacy, pp. 197-213, 2003.

[6] H. Chan, A. Perrig, and D. Song. *Secure Hierarchical In-Network Aggregation in Sensor Networks*, Proc. ACM Conference on Computer and Communications Security, CCS'06, 2006.

[7] B. Carbunar, Y. Yu, L. Shi, M. Pearce, and V. Vasudevan, *Query Privacy in Wireless Sensor Networks*, Proc. 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON '07, pp. 203-212, 2007.

[8] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. *A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks*, Proc. ACM Conference on Computer and Communication Security, CCS'03, pp. 42-51, 2003.

[9] S. Das, C. Perkins and E. Royer. *Performance Comparison of Two On-demand Routing Procotols for Ad Hoc Networks*, Proc. of IEEE INFOCOM 2000, Vol 1, pp. 3-12, IEEE Press, 2000.

[10] L. Eschenauer, and V. D. Gligor. *A Key-Management Scheme for Distributed Sensor Networks*, Proc. ACM Conference on Computer and Communication Security, CCS'02.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. *Next Century Challenges: Scalable Coordination in Sensor Networks*, Proc. ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM'99.

[12] A. C. Ferreira et al. *On the Security of Cluster-based Communication Protocols for Wireless Sensor Networks*, Proc. International Conference on Networking, ICN'05, pp. 449-458, 2005.

[13] P. Gupta and P.Kumar. *The Capacity of Wireless Networks*, IEEE Transactions on Information Theory, Vol 46(2), pp. 388-404, 2000.

[14] S. Ganeriwal, S. Ravi, and A. Raghunathan. *Trusted Platform Based Key Establishment and Management for Sensor Networks*, Under Review.

[15] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. *Hiding Routing Information*, Proc. 1st International Workshop on Information Hiding, LNCS 1174, pp. 137-150, 1996.

[16] L. Hu, and D. Evans. *Secure Aggregation for Wireless Networks*, Proc. 2003 Symposium on Applications and the Internet Workshops, SAINT'03, pp. 384-394, 2003.

[17] Y. Hu, D. Johnson, and A. Perrig. *SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks*, Ad Hoc Networks Journal, Vol. 1(1), pp. 175-192, 2003.

[18] Y. Hu, A. Perrig, and D. Johnson. *Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks*, Wireless Networks Journal, Vol. 11(1), 2005.

[19] J. Ibriq and I. Mahgoub. *A Hierarchical Key Establishment Scheme for Wireless Sensor Networks*, Proc. 21st International Conference on Advanced Networking and Application, AINA'07, 2007.

[20] http://www.intel.com/research/exploratory/heterogeneous.htm.

[21] N. Jain, and D. P. Agrawal. *Current Trends in Wireless Sensor Network Design*, International Journal of Distributed Sensor Networks, Vol 1(1), pp. 101-122, 2005.

[22] A. Juels, P. Syverson, and D. Bailey. *High-Power Proxies for Enhancing RFID Privacy and Utility*, Proc. Privacy Enhancing Technologies (PET) Workshop, pp. 210-226. 2005.

[23] C. Kraub, F. Stumpf, and C. Eckert. *Detecting Node Compromise in Hybrid Wireless Sensor Networks Using Attestation Techniques*, Proc. 4th European Workshop on Security and Privacy in Adhoc and Sensor Networks, ESAS 2007, LNCS 4572, pp. 203-217.

[24] C. Karlof, and D. Wagner. *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasurements*, Proc. 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[25] P. Kamat, W. Xu, W. Trappe, and Y. Zhang. *Temporal Privacy in Wireless Sensor Networks*, Proc. 27th International Conference on Distributed Computing Systems, ICDCS'07, pp. 23-30, 2007.

[26] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. *Enhancing Source-Location Privacy in Sensor Network Routing*, Proc. 25th IEEE International Conference on Distributed Computing Systems, ICDCS'05, pp. 599-608, 2005.

[27] D. Liu, and P. Ning. *Location-based Pairwise Key Establishement for Relatively Static Sensor Networks*, Proc. ACM Workshop on Security of Ad hoc and Sensor Networks, 2003.

[28] D. Liu, and P. Ning. *Improving Key Pre-distribution wih Deployment Knowledge in Static Sensor Networks*, ACM Transactions on Sensor Networks, 2005.

[29] D. Liu, P. Ning, and W. Du. *Group-based Key Pre-distribution in Wireless Sensor Networks*, Proc. ACM Workshop on Wireless Security, 2005.

[30] D. Liu, P. Ning, and K. Sun. *Efficient Self-Healing Group Key Distribution with revocation Capability*, Proc. ACM Conference on Computer and Communication Security, CCS'03, 2003.

[31] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. *TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks*, Proc. 5th Annul Symposium on Operating Systems Design and Implementation, OSDI'02, 2002.

[32] K. Mehta, D. Liu, and M. Wright. *Location Privacy in Sensor Networks Against a Global Eavesdropper*, Proc. IEEE International Conference on Network Protocols, ICNP'07, pp. 314-323, 2007.

[33] D. Molnar, A. Soppera, and D. Wagner. *Privacy for RFID Through Trusted Computing*, Proc. ACM workshop on Privacy in the Electronic Society, WPES'05, pp. 31-34, 2005.

[34] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon. *Entrapping Adversaries for Source Protection in Sensor Networks*, Proc. International Symposium on World of Wireless, Mobile and Multimedia Network, WoWMoM'06, pp. 23-34, 2006.

[35] L. B. Oliveira, H. C. Wong, and A. A. Loureiro. *LHA-SP: Secure Protocols for Hierarchical Wireless Sensor Networks*, Proc. IFIP/IEEE International Symposium on Integrated Network Management, pp. 31- 44, 2005.

[36] C. Ozturk, Y. Zhang, and W. Trappe. *Source-location Privacy in Energy-contained Sensor Network Routing*, Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN'04, pp. 88-93, 2004.

[37] A. Patwardhan, J. Parker, A. Joshi, M. Iorga, and T. Karygiannis. *Secure Routing and Intrusion Detection in Ad Hoc Networks*, Proc. 3rd International Conference on Pervasive Computing and Communications, IEEE, 2005.

[38] B. Przydatek, D. Song, and A. Perrig. *SIA: Secure Information Aggregation in Sensor Networks*, Proc. ACM SenSys, 2003.

[39] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. *SPINS: Security Protocols for Sensor Networks*, Wireless Networks Journal (WINE), September 2002.

[40] A. U. Schmidt, N. Kuntze, and M. Kasper. *On the deployment of Mobile Trusted Modules*, Proc. Wireless Communications and Networking Conference, WCNC'08, 2008.

[41] Trusted Computing Group. *www.trustedcomputinggroup.org*.

[42] TCG Mobile Phone Work Group. *https://www.trustedcomputinggroup.org/groups/mobile*.

[43] H. Wang, and Q. Li. *Distributed User Access Control in Sensor Networks*, Proc. Distributed Computing in Sensor Systems, LNCS 4026, pp. 305-320, 2006.

[44] M. Yarvis, et al. *Exploiting Heterogeneity in Sensor Networks*, Proc. IEEE INFOCOM'05.

[45] S. Zhu, S Setia, and S. Jajodia. *LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks*, Proc. ACM Conferenc on Computer and Communication Security, CCS'03, pp. 62-72, 2003.

[46] Y. Zhou, Y. Zhanga, and Y. Fang. *Access control in wireless sensor networks*, Ad Hoc Networks Vol. 5(1), pp. 3-13, Elsvier, 2007.