Anonymous Path Routing in Wireless Sensor Networks

Jang-Ping Sheu
Department of CS
National Tsing Hua University
Hsinchu, Taiwan

Jehn-Ruey Jiang
Department of CSIE
National Central University
Jhongli, Taiwan

Ching Tu
Department of CSIE
National Central University
Jhongli, Taiwan

Abstract—**How to secure data communication is an important problem in wireless sensor networks (WSNs). General solutions to the problem are to encrypt the packet payload with symmetric keys. But those solutions only prevent the packet content from being snooped or tampered. Adversaries still can learn network topology by the traffic analysis attack. In this paper, we propose an anonymous path routing (APR) protocol for WSNs. In APR, data are encrypted by pair-wise keys and transmitted with anonyms between neighboring sensor nodes and anonyms between the source and destination nodes of a multi-hop communication path. The encryption prevents adversaries from disclosing the data, and the anonymous communication prevents adversaries from observing the relation of the packets for further attacks. We implement APR on the MICAz platform to evaluate its overheads for demonstrating its applicability in practical WSNs.**

Keywords-Anonymous routing; pair-wise key; symmetric cryptography; wireless sensor networks

I. INTRODUCTION

A wireless sensor networks (WSN) consists of many spatially distributed, resource-constrained sensor nodes equipped with microcontrollers, short-range wireless radios, and analog/digital sensors. There are many applications of WSNs, like battle field surveillance, environmental monitoring, intrusion detection, and health care, etc. Sensor nodes sense environmental conditions, such as temperature, light, sound, or vibration, and transmit the sensed data to the sink node through multi-hop communication links. The sensed data are critical for some applications, especially for military ones. How to keep the sensor node communication secure is thus an important issue in WSNs. Since sensor nodes are resource-constrained, conventional security schemes using asymmetric cryptography cannot be directly applied to WSNs. New mechanisms are required to achieve secure communication for WSNs.

When sensor nodes communicate with each other via wireless transmission, the adversaries can easily eavesdrop on every packet. By analyzing the packets, adversaries can derive useful information and/or start attacks. The attacks are categorized into two types: active attacks and passive attacks. As implied by the name, active attacks are "invasive." Typical examples are *replaying attacks*, *denial-of-service attacks* and *forging attacks*, etc [15]. In contrast to active attacks, the passive attacks are "non-invasive" and difficult to detect. Typical examples are *traffic analysis* [10] and packet eavesdropping. *Traffic analysis* is usually the prelude of active attacks which can really damage the sensor networks. There are many schemes proposed for resisting attacks. SDAP [14] uses network topology changes to prevent adversaries from attacking the most effective data aggregation nodes in WSNs. However, SDAP has high control overhead due to the frequent changes of the network topology. Some protocols, such as ANODR [9], AnonDSR [12], SDAR [1], and MASK [15], use the concept of *anonymous communication* to hide identities of nodes participating in the communication to resist attacks in mobile ad hoc networks. Providing anonymity in WSNs is useful since hiding node identities in routing paths makes it more difficult for adversaries to identify the more active nodes to attack. However, the above-mentioned protocols for ad hoc networks are not suitable to WSNs due to high computation and communication overheads caused by asymmetric cryptography.

In this paper, we propose an Anonymous Path Routing (APR) protocol to achieve anonymous communication in WSNs. APR hides the identities of all nodes in the routing path and encrypt the data by a pair-wise key shared by the sender and the receiver. Only the sender and the receiver can decrypt the data and uncover the sender's and receiver's identities, while the others cannot. As a result, the adversaries cannot derive actual traffic patterns by snooping packets or even by compromising sensor nodes in the path. The WSNs can thus resist *traffic analysis* attacks and prohibit further attacks. Furthermore, APR is a two-way routing protocol; i.e., when an anonymous path from the source to the destination is established, a reverse anonymous path from the destination to the source is also obtained.

APR utilizes three basic schemes: (1) anonymous one-hop communication, (2) anonymous multi-hop path routing, and (3) anonymous data forwarding. By the first scheme, each node can create two unique pseudonyms for each link between itself and one of its neighbors with the help of a distributed pair-wise key establishment protocol. One pseudonym is for the in-bound direction of the link; the other, out-bound direction. Data are encrypted and sent without revealing the real identities of the sender and receiver. The encryption key changes for every data transmission, which makes it harder for adversaries to break the data encryption. By the second scheme, if a node needs to communicate with a node multiple hops away, it can find a two-way routing path in an anonymous way. Once an anonymous path is found, APR assigns a pseudo ID to the path for identification. An intermediate node in the path can identify

This work was supported by the National Science Council of R.O.C. under grants NSC 95-2221-E-008-019 and NSC 96-2221-E-008-007.

the path and know its pre-hop and/or next-hop nodes in the path, while the other nodes cannot. By the third scheme, packets can be forwarded in an anonymous way by using the pseudo ID. If adversaries eavesdrop on packets, they can only gather a large pool of pseudonyms but have no knowledge of the relationship between pseudonyms and real traffic links. This prevents adversaries from figuring out the network topology. For demonstrating the applicability and communication capability of APR, we implement it on the sensor platform MICAz with TinyOS operating system [6]. By the implementation, we observe that APR do not cost heavy computing overhead and long time delay on sensor nodes; APR is applicable to practical wireless sensor networks.

The remainder of this paper is organized as follows. Section II shows some related work. The design of APR is detailed in Section III. In Section IV, we analyze the security properties of APR. In Section V, we explain the implementation of APR on MICAz, and evaluate its computational overheads. Finally, we conclude the paper in Section VI.

II. RELATED WORK

In order to resist the malicious traffic analysis and/or other attacks in WSNs, some countermeasures have been developed. Since sensor nodes are resource-constrained, most countermeasures are based on mechanisms using symmetric keys, such as symmetric encryption/decryption and message authentication code (MAC). Those mechanisms require that symmetric key should be distributed to nodes beforehand. One common symmetric key distribution scheme for WSNs is the *pair-wise key* scheme [3], which requires two sensor nodes to communicate with each other to decide a shared key that only the two nodes know. This scheme can prevent a compromised node from disclosing the packet contents. Most distributed pair-wise key establishments are based on pre-distributed pair-wise keys [7].

Although symmetric cryptography and the pair-wise key scheme can protect packet contents from being uncovered by adversaries, the identities of communication parties can still be revealed and network traffic patterns can thus be derived. It is possible that this kind of information leakage could cause devastating security leak. The paper [15] proposes SDAP to reselect aggregation leaders for performing data aggregation every time when the sink node wants to collect the sensed data in the WSN. Since the topology for data aggregation changes frequently, SDAP prevents adversaries from locating the aggregation leader nodes. The cost of SDAP is high since a sensor node needs re-establish pair-wise keys for new leader nodes. Nevertheless, the adversaries can still learn the communication relationships between sensor nodes and involve some local attacks such as the replay attack and the denial-of-service attack.

There is another effective solution that uses the concept of anonymous communication to resist malicious attacks by hiding the sender's and/or receiver's identities. If adversaries cannot identify the packet sender and receiver, they have no way to learn the network topology and the relationship of communication parties. Several anonymous communication routing protocols for ad hoc networks are proposed in [1, 2, 8, 9, 13, 15]. Below, we describe some of them. In ANODR [9], routing request is encrypted with the "onion scheme" before it is forwarded. The onion scheme encodes routing information in a set of encrypted layers; each node appends the information about the forwarding node to the packet and then encrypts the appended packet by a random key each time the packet is retransmitted. In this manner, ANDOR provides route pseudonym. To improve ANODR protocol, AnonDSR [13] requires that the data should also be encrypted by onion scheme in order to prevent the global attacker from finding out the route by comparing the data part of the packets. MASK [15] is a pairing-based anonymous on-demand; it achieves anonymous neighborhood authentication with the assistance of certificate authority (CA) and key agreement from bilinear pairing [4]. The anonymous communication protocols for ad hoc networks are not suitable for WSNs due to their computation and memory overheads.

III. DESIGN OF ANONYMOUS PATH ROUTING (APR) PROTOCOL

A. Overview of APR

Anonymous Path Routing (APR) protocol proposed in this paper is designed for achieving anonymous communication in WSNs. In APR, nodes request routes and exchange data by recognizing hidden identities. APR is supposed to have the following properties:

1) *Transmission relationship anonymity:* The sender's and receiver's identities in the packet header are replaced by pseudonyms. Even if adversaries can eavesdrop on packets, they still cannot find where the packet comes from and where the packet goes to.
2) *Authentication through anonyms:* Any pair of nodes can authenticate mutually without revealing their identities.
3) *Unlocatability:* The adversaries cannot locate nodes by tracing the transmitted packets that they overhear back to the source or to the destination.
4) *Limited area of leakage:* If a node is compromised, the effect of the compromised behavior will be limited in a local area.
5) *Secure data transmission:* The intermediate nodes do not know the source and destination of the packets that they are forwarding. And only the source and the destination of the packet can decrypt the cipher of the packet payload.
6) *Resilience to packet loss*: Due to packet collision or other wireless medium interference, packet loss occurs frequently, and causes the sensor nodes confuse about the anonyms.
7) *Light-weight computation*: APR can achieve anonymous communication without heavy computation overhead. It is thus suitable for sensor nodes with limited resources.

B. Network Assumptions and Notations

We assume that the wireless links are symmetric and each node has limited transmission and reception capabilities. Notations used throughout this paper are defined as follows:

- ID_A: identity for node A
- rn: a random nonce number
- Seq_{AB} (= Seq_{BA}): sequence number of the link between node A and node B
- K_{AB} (= K_{BA}): the pair-wise key shared with node A and node B
- $K^i_{AB\text{-}enc}$ (=$K^i_{BA\text{-}enc}$): the i-th pair-wise key shared with node A and node B used to encrypting packet payload
- $K_{AB\text{-}enc}$ (=$K_{BA\text{-}enc}$): the set of $K^i_{AB\text{-}enc}$ for $i = 0\ldots n$
- $E\{K_{AB}, M\}$: a messge M encrypted with pair-wise key K_{AB}
- H: a one way hash function
- $K^i_{AB\text{-}mac}$ (=$K^i_{BA\text{-}mac}$): the i-th pair-wise key shared with node A and node B used to compute MAC (Message Authentication Code of the packet)
- $K_{AB\text{-}mac}$ (=$K_{BA\text{-}mac}$): the set of $K^i_{AB\text{-}mac}$ for $i = 0\ldots n$
- $HI^i_{A\to B}$: the i-th *hidden identity* presenting the link from node A to node B
- $HI_{A\to B}$: the set of $HI^i_{A\to B}$ for $i = 0\ldots n$
- HIP_{SD} (=HIP_{DS}): the *hidden identity* for a pair of source node S and destination node D

C. Three Schemes of APR

APR protocol consists of three schemes: anonymous one-hop communication, anonymous multi-hop path routing, and anonymous data forwarding. Below, we describe the three schemes one by one.

1) Anonymous One-hop Communication

This scheme is performed right after the sensors are deployed. By this scheme, every sensor node establishes a bidirectional anonymous communication link for each of its one-hop neighbors after sensor nodes are deployed in the sensing field. In order to achieve this goal, any two neighboring nodes must have a pair-wise key to create *hidden identities* (HIs) for each other. In practice, APR can rely on a pair-wise key establishment protocol like PIKE [12] to establish a one-hop pair-wise key K_{AB} and a random nonce number rn for a communication link between node A and A's neighboring node B. Afterwards, APR establishes two more keys, namely data encryption key $K_{AB\text{-}enc}$ and *MAC* (Message Authentication Code) encryption key $K_{AB\text{-}mac}$. APR also establishes two hidden identities, namely $HI_{A\to B}$ and $HI_{B\to A}$, for the anonymous links from A to B and from B to A, respectively. $K_{AB\text{-}enc}$, $K_{AB\text{-}mac}$, $HI_{A\to B}$ and $HI_{B\to A}$ are used only once; APR alters them when they are ever used. $K_{AB\text{-}enc}$ and $K_{AB\text{-}mac}$ are first calculated by hashing the values of ($K_{AB} \oplus C_1$) and ($K_{AB} \oplus C_2$) respectively, where $\oplus$ stands for EXCLUSIVE OR operation and C_1 and C_2 are pre-specified constants (refer to (1)). Afterwards, $K_{AB\text{-}enc}$ and $K_{AB\text{-}mac}$ are calculated by hashing their previous values (refer to (2)). And $HI_{A\to B}$ and $HI_{B\to A}$ are calculate by hashing the values of ($K_{AB} \oplus ID_B \oplus Seq_{AB} \times rn$) and ($K_{AB} \oplus ID_A \oplus Seq_{AB} \times rn$) respectively, where Seq_{AB}, which is initially 0, is the sequence number of the packet transmitted between A and B (refer to (3)).

$$\begin{cases} K^0_{AB-enc} = H(K_{AB} \oplus C_1) \\ K^0_{AB-mac} = H(K_{AB} \oplus C_2) \end{cases} \quad (1)$$

$$\begin{cases} K^{i+1}_{AB-enc} = H(K^i_{AB-enc}) \\ K^{i+1}_{AB-mac} = H(K^i_{AB-mac}) \end{cases} \quad (2)$$

$$\begin{cases} HI^{Seq_{AB}}_{A->B} = H(K_{AB} \oplus ID_B \oplus Seq_{AB} \times rn) \\ HI^{Seq_{AB}}_{B->A} = H(K_{AB} \oplus ID_A \oplus Seq_{AB} \times rn) \end{cases} \quad (3)$$

By (1), (2) and (3), a sensor node can create a link table which contains an entry for each link between itself and one of its one-hop neighbors. Each entry has the fields of the one-hop neighbor ID, the sequence number of the link, K_{enc}, K_{mac}, *HI-in*, and *HI-out*, where *HI-in* is the hidden identity for the in-bound direction, and *HI-out*, the out-bound direction. For example, assume node A has 3 one-hop neighbors, B, C, and D. After the pair-wise key establishment, A has the initial link table as shown in Table I.

TABLE I. THE INITIAL LINK TABLE OF NODE A

ID	*Seq*	*HI-in*	*HI-out*	K_{enc}	K_{mac}
ID_B	0	$HI^0_{B\to A}$	$HI^0_{A\to B}$	$K^0_{AB\text{-}enc}$	$K^0_{AB\text{-}mac}$
ID_C	0	$HI^0_{C\to A}$	$HI^0_{A\to C}$	$K^0_{AC\text{-}enc}$	$K^0_{AC\text{-}mac}$
ID_D	0	$HI^0_{D\to A}$	$HI^0_{A\to D}$	$K^0_{AD\text{-}enc}$	$K^0_{AD\text{-}mac}$

The link table changes for every communication. For example, the entry in the link table for the link from node A to node B becomes the six-tuple (ID_B, i, $HI^i_{A\to B}$, $HI^i_{B\to A}$, $K^i_{AB\text{-}enc}$, $K^i_{AB\text{-}mac}$) after the i-th . If a node A wants to send the (i+1)-th packet to B, it uses the corresponding out-bound hidden identity $HI^i_{A\to B}$ in the field *HI-out* to represent the link from A to B anonymously. Node A also encrypts the packet payload with the key $K^i_{AB\text{-}enc}$ in the field K_{enc}, and prepares a MAC for the packet payload by the key $K^i_{AB\text{-}mac}$ in the field K_{mac}. To be more precise, the (i+1)-th packet from A to B is of the form $<HI^i_{A\to B}, E\{K^i_{AB\text{-}enc}, DATA\|(i+1)\}, \text{MAC}(K^i_{AB\text{-}mac})>$. When node B receives the packet, it will check if the packet is sent to itself by comparing $HI^i_{A\to B}$ with all in-bound hidden identities of the field *HI-in* in its link table. If there is a match, node B receives this packet, decrypts the packet by $K^i_{AB\text{-}enc}$ and acknowledges the packet. Otherwise, it drops the packet.

One-hop acknowledgement

After the sender receives an acknowledgement (ACK) packet, form the receiver, the sender and receiver have both updated the values of the fields *Seq*, *HI-in*, *HI-out*, K_{enc}, and K_{mac} of corresponding link table entries by (2) and (3). Nodes can then communicate with the new setting properly. However, packet loss and transmission errors may occur to make the sender's and the receiver's link tables out of synchronization. This will hinder future communication of the link.

In order to solve this problem, APR requires one more field *Old-HI-in* in the link table to store the previous value of the *HI-in* field, and requires a receiver B to send an ACK packet, $<HI^i_{B\to A}, E\{K^i_{AB\text{-}enc}, ACK\|(i+1)\}, \text{MAC}(K^i_{AB\text{-}mac})>$, to the sender A when B receives A's (i+1)-th packet. The last ACK packet should also be kept for possible further use. On receiving a packet, B sends an ACK packet and updates the values of fields *Seq*, *HI-in*, *HI-out*, K_{enc}, K_{mac}, and *Old-HI-in* of the

corresponding link table entry. On receiving the ACK packet, *A* updates the values of fields *Seq*, *HI-in*, *HI-out*, K_{enc}, and K_{mac} of the corresponding link table entry. If *A* cannot receive the ACK packet after a timeout period due to the packet loss or transmission errors, *A* retransmits the packet. Node *B* can identify the retransmitted packet since its HI can match with a value of the field *Old-HI-in* of source link table entry. The coming of a retransmitted packet means that the ACK packet has been lost and *B* just resends the last ACK packet kept. After *A* receives the ACK packet properly, its link table is updated and in synchronization with *B*'s again.

2) Anonymous Multi-hop Path Routing

After one-hop anonymous link are established, a data source node may communicate with a destination node multiple hops away. APR establishes an anonymous two-way multi-hop path between the source and destination nodes. It contains two steps: *anonymous path routing request* and *anonymous path routing reply*. We assume the source node and the destination node share a pre-distributed pair-wise key. When the pair of nodes do not share a pre-distributed pair-wise key, they can first establish a pair-wise key by integrating the anonymous multi-hop path routing scheme with a pair-wise key establish protocol like PIKE. Afterwards, the routing can proceeds normally with the newly established pair-wise key. However, we omit the details in this paper.

Anonymous Path Routing Request

When a source node *S* wants to find a path to a destination node *D* multiple hops away, *S* broadcasts an anonymous path routing request (APR-REQ) locally. APR puts a hidden identity HIP instead of a destination node identity in the routing request. HIP stands the hidden identity for the pair of the source and the destination nodes. It is computed by hashing the value of $K_{SD} \oplus ID_S \oplus ID_D$, where K_{SD} is the pre-distributed pair-wise key and ID_D (resp., ID_S) is the identity of node *D* (resp., *S*).

$$HIP_{SD} = H(K_{SD} \oplus ID_S \oplus ID_D) \quad (4)$$

In APR, each sensor node maintains a HIP table right after it is deployed. The HIP table contains an entry for each possible source node that shares a pre-distributed pair-wise key with itself. Each entry has the fields of *ID*, *K* and HIP, which stand for the source node identity, the pair-wise key and in-bound HIP, respectively. The routing request is flooded to the entire network to find a path from source to destination. The nodes which receive routing request APR-REQ but are not the destination should keep the request for further use. Then they rebroadcast this routing request locally.

This APR-REQ packet has the form <*APR-REQ*, ID_S, HIP_{SD}, $RSeq_{S \to D}$>, where $RSeq_{S \to D}$ is set to be the last known routing request sequence number from *S* to *D*. The second field of the packet represents the identity of the node which initiates or rebroadcasts the packet. It is initially set to ID_S, the identity of *S*, and will be replaced by the identity of the node which rebroadcasts the packet. Every node which receives the APR-REQ packet stores it for further use. The node checks if there is any entry of HIP table matching HIP_{SD} of the APR-REQ packet. If so, the node is the destination node. If not, the node rebroadcasts the packet but replaces ID_S with its own identity. The steps to send anonymous path routing request packet from *S* to *D* through nodes *E* and *C* is depicted in Fig. 1 (steps 1 to 3).

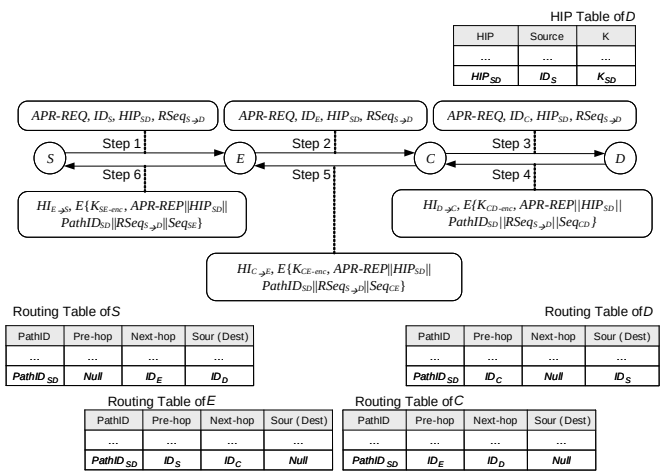


Figure 1. Anonymous multi-hop path establishment from the source *S* to the destination *D*

Anonymous Path Routing Reply

When the destination node *D* receives a routing request APR-REQ of node *S* from a neighboring node *C* (refer to Fig. 1 step 4), *D* specifies *C* as the forwarding node to send an anonymous path routing reply APR-REP to *S*. The APR-REP packet has the form <$HI_{D \to C}$, $E\{K_{CD\text{-}enc}, APR\text{-}REP||HIP_{SD}||PathID_{SD}||RSeq_{S \to D}||Seq_{CD}\}$, MAC($K_{CD\text{-}mac}$)>. Note that, we omit MAC field in the following context for simplicity. This routing reply contains a path identity $PathID_{SD}$ of the route path between *S* and *D*. Each route path is two-way, so $PathID_{SD}$ represents not only the path from *S* to *D*, but also the reverse path from *D* to *S*. $PathID_{SD}$ is generated by *D* randomly; however, every path identities for the path between *S* and *D* should be different.

Each node maintains a two-way routing table consisting of entries with the fields *PathID*, *Pre-hop*, *Next-hop*, and *Sour* (*Dest*). A node updates the table when receiving an APR-REP packet. The field *PathID* stores the identity of the route path, *Dest* stores the identity of the destination node, *Pre-hop* stores the identity of the node where the APR-REQ packet comes from; and *Next-hop* stores the identity of the node from which the corresponding APR-REP packet comes. After *D* sends out this APR-REP and receives ACK packet from *C*, *D* inserts an entry or updates the corresponding entry in its routing table as follows. It puts $PathID_{SD}$ into the *PathID* field, puts ID_C into the *Pre-hop* field, puts ID_S into the *Sour* (*Dest*) field, and sets the *Next-hop* field as *Null*.

When node *C* receives the APR-REP from *D*, *C* first figures out the pre-hop node, say node *E*, by matching HIP_{SD} of the APR-REP with that of previous APR-REQ packet stored in its memory. When a match occurs, the corresponding APR-REQ is purged from the memory. The APR-REP is then encrypted with the pair-wise key $K_{CE\text{-}enc}$ and sent to *E*. Node *C* inserts an entry in its two-way routing table as follows. It puts $PathID_{SD}$ in the *PathID* field; ID_D, *Next-hop* field; ID_E, *Pre-hop* field. However, the *Sour* (*Dest*) field is set as *Null* because *C* is just a forwarding node but not a destination or a source in this route path. Through hops of relaying, the source node *S*

receives an APR-REP with HIP_{SD} from a relaying node *E*. *S* recognizes that the APR-REP is a reply for the APR-REQ sent by *S* previously. *S* inserts an entry or updates the corresponding entry as follows. It sets the field *Pre-hop* as *Null* and puts $PathID_{SD}$ in the *PathID* field; ID_E, *Next-hop* field; ID_D, *Sour* (*Dest*) field. The steps to send anonymous path routing reply APR-REP through node *E* and node *C* is depicted in Fig. 1 (steps 4 to 6). By the procedures mentioned above, a path from *S* to *D* is established successfully and anonymously.

Because the identities of the source node and the destination node do not appear in the routing request and routing reply, the forwarding nodes only know where it should relay to but no other information. It is worthwhile to mention that the established path is a two-way path instead of a one-way path established by another protocol like MASK.

3) Anonymous Data Forwarding

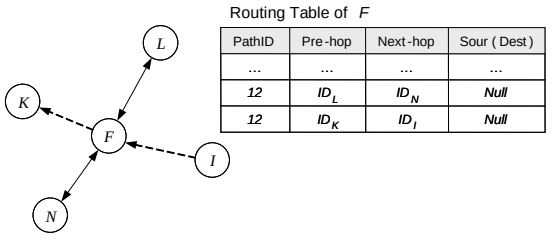
When the source node *S* wants to send a data packet to the destination node *D* multiple hops away after the anonymous path routing establishment (suppose the forwarding node is *E*), It first finds out $PathID_{SD}$ and the forwarding node identity ID_E for *D* by looking up the routing table. The identity of the forwarding node is stored in non-Null *Pre-hop* field or *Next-hop* field of the entry for destination *D*. Then *S* sends out the data packet to the forwarding node *E* with the form $<HI_{S\rightarrow E}, E\{K_{SE\text{-}enc}, MHOP\text{-}DT||PathID_{SD}||E\{K_{SD\text{-}enc}, DATA\}||Seq_{SE}\}>$, where MHOP-DT stands for the packet label for *multi-hop data transmission*. When a forwarding node *C* receives a MHOP-DT packet, it looks for the entry of $PathID_{SD}$ in its two-way routing table. If the identity stored in the *Pre-hop* (resp., *Next-hop*) field matches with ID_E (the identity of the node from which the packet comes), the next forwarding node is with the identity stored in the *Next-hop* (resp. *Pre-hop*) field. If the *Next-hop* (resp. *Pre-hop*) is *Null*, the destination node is reached. The destination node *D* knows the source node identity ID_S which is stored in the *Sour* (*Dest*) field in the routing table. Then *D* can decrypt the data cipher with the key $K_{SD\text{-}enc}$ shared by *S* and *D*.

D. PathID Collision Problem

Since PathID of a path is generated randomly by the destination node of the path, two different paths crossing at a same forwarding node may have the same PathID. We call this the PathID collision problem. There are two cases for such a problem: (case 1) the *Pre-hop* fields of the paths are different and (case 2) the *Pre-hop* fields of the two paths are identical.

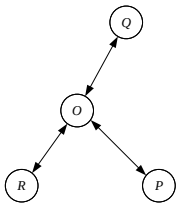
The scenario depicted in Fig. 2 is an example of case 1. The forwarding node, *F*, has two paths with PathID = 12. Since both the *Pre-hop* and *Next-hop* fields of the paths are different, the forwarding node can choose proper node to forward the packet. For example, packets with PathID = 12 from *I* should be forwarded to *K*, and packets with PathID = 12 from *L* should be forwarded to *N*.

The scenario depicted in Fig. 3 is an example of case 2. The forwarding node *O* receives an APR-REP packet with PathID = 13 from node *P*. According to the HIP in the APR-REP packet, this packet should be relayed to node *Q*. But there is already a path with PathID = 13 in the routing table, and the relay (pre-hop) node of the path is also, *Q*. To distinguish the two different paths, node *O* changes PathID from 13 to another number, say 14, in the APR-REP packet and sends it to the relay node *Q*. After that, if node *O* receives a packet from node *Q* with PathID = 14, it forwards this packet to node *P* by changing PathID to 13. Therefore, node *O* will keep PathIDs 13 and 14 in the routing table. To indicate that a PathID should be changed to a new one, node *O* stores the new PathID in *Change* field. If the *Change* field is *Null*, the PathID has no need to be changed. To make sure that the forwarding node *O* can choose a proper node for forwarding the packet sent from node *Q*, an *Original* field is added in the routing table. *Original* field of the first path with PathID = 13 is set to be "*true*", which means this entry should be checked first if node *O* receives a packet with PathID = 13. After checking the entry with *Original* is true, if the packet is sent from *Q* (*R*), node *O* forwards the packet to *R* (*Q*). Otherwise, node *O* will check the entry with *Original* being false and determines its next relay node accordingly.



PathID	Pre-hop	Next-hop	Sour (Dest)
...	...	...	...
12	ID_L	ID_N	Null
12	ID_K	ID_I	Null

Figure 2. Two different paths with the same PathID and different *Pre-hop* nodes



PathID	Pre-hop	Next-hop	Sour (Dest)	Original	Change
...	...	...	...	...	...
13	ID_Q	ID_R	Null	true	Null
13	ID_Q	ID_P	Null	false	14
14	ID_Q	ID_P	Null	false	13

Figure 3. Two different paths with the same PathID and *Pre-hop* node

IV. SECURE ANALYSIS OF ANONYMITY

In this Section, we analyze how well APR achieves anonymity for communication In APR, except the rebroadcast node ID in the APR-REQ packet, no real node identity appears in any packet after one-hop pair-wise keys are established. But the rebroadcast node ID is useless for adversaries because any local communication uses hidden identity. Furthermore, a multi-hop communication path is anonymous since it is identified by PathID and its source and the destination nodes are represented by HIP, which can only be identified by the corresponding source and destination nodes. Due to the anonymity, APR can resist the following attacks.

- *Traffic Analysis attacks*

In one-hop communication, two neighboring nodes use different parameters, such as hidden identity HI, data encryption key and MAC encryption key, for different packets.

Except the two communication nodes, no node can uncover the data packet or identify the sending and receiving nodes. Although the receiving node sends an ACK packet immediately to confirm that the packet is successfully received, the HI in the ACK packet is different from the HI of the received packet. So the other nodes cannot learn the relationship between the sending and receiving nodes to derive the network topology. That prevents the traffic analysis attack.

- *Forging attacks*

In APR, hidden identity should be used in common data transmission. If the adversary sends a malicious packet with a forged HI, the packet will be accepted with probability $1/2^h$, where h is the length of HI. However, even the adversary has a correct guess of HI, it should have a correct guesses of the packet MAC to make the forged packe be accepted. If the MAC has a length of m, the adversary has a $1/2^{h+m}$ chance in blindly forging a valid HI and MAC for a particular packet. A typical setting of h and m is h = 16 and m = 32. Under such a setting, if an adversary repeatedly attempts blind forging, s/he is guaranteed to succeed after about 2^{48} tries. Adversaries can try to broadcast with forgeries, but on a 125kb/s channel, sending 2^{48} packets would spend over 500 years.

- *Replay attacks*

Replay attacks use the legal packets sent before which have correct MAC and cipher. In APR, each HI is used once only, so sensor nodes will discard replayed packets. In order to maintain the synchronization of link table entries between neighboring nodes, APR keeps the last HI. If adversaries replay a data packet, this packet will only be accepted by the receiving node only once. This is because the receiving node will adjust its link table and update the last HI when accepting the replayed packet. Therefore, APR can resist replay attacks effectively.

- *Denial-of service (DoS) attacks*

DoS attacks are generally hard to resist. In APR, multi-hop communications are split as a chain of one-hop transmissions, which use different HIs. Without correct HI, DoS attack packets will be ignored directly. APR can limit the damage caused by this kind of attacks in a local area.

V. IMPLEMENTATION AND MEMORY OVERHEAD EVALUTION

In order to provide strong security protection and good scalability, APR combines symmetric key cryptographic algorithm, hash function and message authentication code together. We use Skipjack block cipher algorithm [13] as the symmetric key cryptosystem, SHA-1 [5] as the hash function, and CBC-MAC [11] as the MAC function.

We have implemented APR on the Berkeley sensor nodes to demonstrate its applicability and communication capability. In our implementation, we assume the pair-wise keys are pre-distributed. So the PIKE protocol is not included in our implementation. APR currently runs on the MICAz platform with TinyOS operating system. And it also can run on the MICA and MICA2 platform. The implementation of APR requires 9436 bytes of program space. The required SRAM size depends on the network size and node density.

In our implementation of Skipjack, encrypting 24 bytes data costs 1.51 milliseconds. And the operation of executing link table update after each one-hop communication costs 1.27 milliseconds. The time of computing MAC costs 0.81 milliseconds. Comparing to the average transmission time of the packet with 24 bytes data payload, which is 27.5 milliseconds, the three operations mentioned above do not cost heavy computing overhead and time delay on sensor nodes. The routing latency of APR is shown in Fig. 4. For a node needs to find a route path to another which is 7 hops away, the average time of path establishment is about 574 milliseconds.

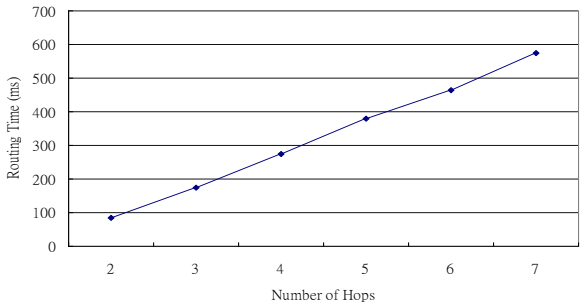


Figure 4. Average routing latency of APR

In APR, a node needs 50 bytes of SRAM for storing an entry of the link table, and needs 8 bytes of SRAM for storing information of a path in the routing table. We conducted analysis to evaluate the memory overhead of APR. Because HIP table is loaded in the program space of the flash memory before nodes are deployed. Thus, only the link table and routing table occupy SRAM. We analyze a WSN with sensor nodes randomly deployed in a $5R \times 5R$ field, where R is the communication range of sensor nodes. The varying parameters used for the analysis are the number of deployed nodes. In the $5R \times 5R$ field, the number of sensor nodes is 25, 50, … or 200, and each node must establish anonymous multi-hop communications with 5, 10, 15 or 20 randomly selected multi-hop neighboring nodes. The analysis results are shown in Figs. 5.2, 5.3 and 5.4.

Fig. 5 shows that the link table size grows with the node density. For the situation of 200 nodes, the link table size goes to 1.1 Kbytes. According to Fig. 6, the routing table size decreases when the number of nodes increases. When node density is higher, a node has more choices of farther one-hop neighbors to establish paths with fewer hops, which pass through fewer nodes. Thus, the average routing table size is smaller. As the number of nodes reaches 125, the routing table size hardly decreases from then on. This is because the number of hop count of paths hardly decreases when number of nodes is higher than 125.

By summing up the link table size and the routing table size per node, we can get the total memory overhead per node in APR, as shown in Fig. 7. The maximum memory overhead of APR shown in Fig. 7 is 1881 bytes when there are 200 nodes in the field, which is affordable for practical sensor nodes. To sum up, by the implementation and analysis results, APR is computationally feasible and only consumes little memory space. It is thus suitable for sensor nodes with limited resources.

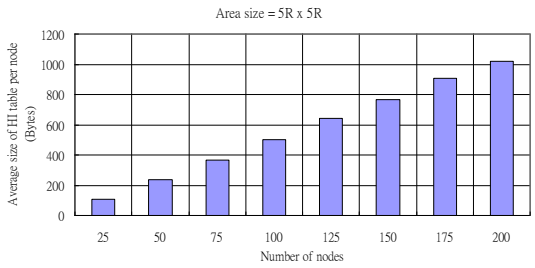


Figure 5. Average link table size of a node in a $5R \times 5R$ field

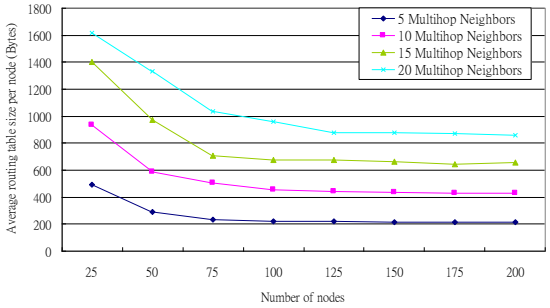


Figure 6. Average routing table size of a node in a $5R \times 5R$ field

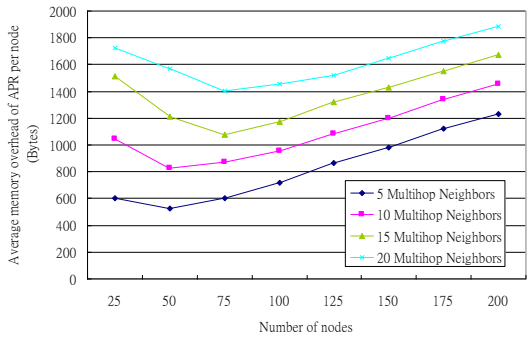


Figure 7. Average memory overhead of a node for varying number of nodes in a $5R \times 5R$ field

VI. CONCLUSION

In this paper, we propose an anonymous path routing protocol, called APR, for WSNs. APR has three basic schemes: (1) anonymous one-hop communication, (2) anonymous multi-hop path routing, and (3) anonymous data forwarding. By the first scheme, each node can create two hidden identities for each link of itself and one of its neighbors. One is for the in-bound direction of the link; the other, out-bound direction. Data are encrypted and sent without revealing the real identity of the sender. By the second scheme, APR can find a two-way multi-hop routing path for it in an anonymous way and assign a pseudonym, called PathID, to the path for identification. Only the pair of the source and destination nodes know each other's identity. The third scheme can forward packets in an anonymous way by using the PathIDs. If adversaries eavesdrop on packets, they can only gather a large pool of pseudonyms but have no knowledge of the relationship between pseudonyms and real traffic links. This prevents adversaries from figuring out the network topology. We implement APR on the sensor platform MICAz with TinyOS operating system and find that it is applicable to practical sensor nodes.

REFERENCES

[1] A. Boukerche, K. El-Khatib, and L. Kobra, "SDAR: A Secure Distributed Anonymous Routing Protocol for Wireless and Mobile Ad Hoc Networks," in *Proceedings of IEEE International Conference on Local Computer Networks* (LCN), pp. 618-624, Florida, USA, November 2004.

[2] B. Zhu, Z. wan, M. S. Kankanhalli, F. Bao, and R. H. Deng, "Anonymous Secure Routing in Mobile Ad Hoc Networks," in *Proceedings of IEEE International conference on Local Computer Networks* (LCN), pp. 102-108, Florida, USA, November 2004.

[3] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for WSNs," in *Proceedings of ACM International Conference on Embedded Networked Sensor Systems* (Sensys), pp. 162-175, Maryland, USA, November 2004.

[4] D. Boneh and M. Franklin, "Identify-based Encryption from Weil Pairing," *SIAM Journal on Computing*, Vol.32, No.3, pp. 586-651, March 2003.

[5] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," in *IETF Request for Comments 3174*, 2001.

[6] D. Gay, P. Levis, and D. Culler, "Software design patterns for TinyOS," in *Proceedings of ACM Conference on Languages, Compilers, and Tools for Embedded Systems* (LCTES), pp. 40-49, Illinois, USA, June 2005.

[7] H. Chan and A. Perrig, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks," in *Proceedings of IEEE International Conference on Computer and Communications Societies* (INFOCOM), pp. 524-535, Miami, USA, March 2005.

[8] J. C. Kao and R. Marculescu, "Real-Time Anonymous Routing for Mobile Ad Hoc Networks," in *Proceedings of IEEE Wireless Communications and Networking Conference* (WCNC), pp. 4139-4144, Hong Kong, China, March 2007.

[9] J. Kong and X. Hong, "ANODR: Anonymous on Demand Routing with Untraceable Routes for Mobile Ad-Hoc Networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing* (MobiHoc), pp. 291-302, Maryland, USA, June 2003.

[10] J. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," in *Proceeding of International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pp. 10-29, California, USA, January 2001.

[11] M. Bellare, J. Kilian, and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," in *Proceedings of International Cryptology Conference* (CRYPTO), pp. 341-358, California, USA, August 1994.

[12] R. Song, L. Korba, and G. Yee, "AnonDSR: Efficient Anonymous Dynamic Source Routing for Mobile Ad-Hoc Networks," in *Proceedings of ACM workshop on Security of ad hoc and sensor networks*, pp. 33-42, Alexandria, USA, November 2005.

[13] Y. W. Law, J. Doumen, and P. Hartel, "Survey and Benchmark of Block Cipher for WSNs," in *Proceedings of ACM Transaction on Sensor Networks* (TOSN), Vol. 2, No. 1, pp. 65-93, Feburary 2006.

[14] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing* (MobiHoc), pp. 356-369, Florence, Italy, May 2006.

[15] Y. Zhang, W. Liu, and W. Lou, "Anonymous Communication in Mobile Ad Hoc Networks," in *Proceedings of IEEE International Conference on Computer and Communications Societies* (INFOCOM), pp. 1940-1951, Miami, USA, March 2005.