An Anonymous Path Routing (APR) Protocol for Wireless Sensor Networks

JEHN-RUEY JIANG[1], JANG-PING SHEU[1,2], CHING TU[1], JIH-WEI WU[1,3]

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 of network topology by the traffic analysis attack for starting devastating attacks such as the denial-of-service attack and the like. 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, network security

1. INTRODUCTION

A wireless network is vulnerable to attacks since nodes in it communicate with each other via open-air radio channels. Secure communication is thus an important issue for wireless networks, such as mobile ad hoc networks (MANETs) [21], wireless mesh networks (WSNs) [1] and wireless sensor networks (WSNs) [12]. It is more challenging to keep WSNs secure for the following reasons: (a) Sensor nodes have limited capability on computing, data storage and communications. (b) Sensor nodes are usually unattended and easier to be captured and compromised.

A wireless sensor network (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. According to the sensed data, the sink node sends commands to specific actuators to activate responsive tasks. Although WSNs are usually provided with many-to-one and one-to-many routing primitives, point-to-point routing support is needed for several applications, such as PEG (a pursuer-evader game in which evader robots are tracked by pursuer robots) [32] and reactive tasking for controlling actuator based on sensed data [10], etc. In this paper, we focus on point-to-point routing for WSNs.

WSNs may be deployed in a harsh and hostile environment, like battle field, where adversaries try any ways possible to attack the network. By analyzing either the data packet or the control packet, adversaries can derive useful information and/or start attacks. The attacks are typically categorized into two types: active attacks and passive attacks. As the terms imply, active attacks are "invasive." Typical examples of active attacks are replaying attacks, denial-of-service (DoS) attacks and forging attacks, etc [34]. In contrast to active attacks, the passive attacks are "non-invasive" and difficult to detect. Typical

[1]Department of Computer Science and Information Engineering, National Central University, Taiwan, R.O.C.
[2]Department of Computer Science, National Tsing Hua University, Taiwan, R.O.C.
[3]Department of Informational Management, Lan-Yang Institute of Technology, Taiwan, R.O.C.

examples of passive attacks are traffic analysis [28] 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 the attacks in wireless networks. SDAP [33] uses the change of network topology to prevent adversaries from attacking the most effective data aggregation nodes in WSNs. However, SDAP has high control overhead due to the frequent change of the network topology. Some protocols, such as ANODR [22], AnonDSR [30], SDAR [7], and MASK [34], use the concept of anonymous communication to hide identities of nodes participating in the communication to resist attacks in MANETs. Anonymous communication is an effective way for secure communication of WSNs since hiding node identities in routing paths makes adversaries harder to couple network traffic with certain nodes for starting attacks. Furthermore, anonymity of sensor identities can protect the networks from being attacked even if some sensor nodes are compromised. However, the above-mentioned protocols for MANETs 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, which achieves anonymous communications in WSNs. APR hides the identities of all nodes in the routing path and encrypt the data with 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. In the first scheme, each node can create a pair of pseudonyms for each link between itself and one of its neighbors via the help of a distributed pair-wise key establishment protocol. One pseudonym is for the in-bound direction of the link and the other is for out-bound direction. Data are encrypted and sent without revealing the real identities of the sender and receiver. The encryption key changes for every packet transmission, which makes it harder for adversaries to break down the data encryption. In the second scheme, if a node needs to communicate with a node multiple hops away, it can establish a two-way routing path with anonymous communication capability. Once an anonymous path is found, APR assigns a pseudo ID to the path for identification. An intermediate node in the path can identify the path and can recognize its pre-hop and/or next-hop nodes in the path, while the other nodes cannot. In 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 [18]. By the evaluation, we observe that APR do not cost heavy computing overhead and do not make long time delay on sensor nodes; APR is thus applicable to practical wireless sensor networks.

The remainder of this paper is organized as follows. Section 2 shows some related work. The design of APR is detailed in Section 3. In Section 4, we analyze the anonymity and security properties achieved in APR. In Section 5, we explain the implementation of APR on MICAz, and evaluate its overheads. Finally, we draw the conclusion in Section 6.

2. 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 the 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, which requires two sensor nodes to communicate with each other via a shared key that only the two nodes know. This scheme can

prevent a compromised node from disclosing the packet contents. However, pair-wise key distribution should be performed beforehand to make the scheme possible. Some pair-wise key distribution schemes have been proposed [8][9][15][17]. Among them, PIKE (Peer Intermediaries for Key Establishment) [8] is a memory-efficient one. In PIKE, each node just pre-stores $O(\sqrt{n})$ keys and can on-the-fly establish a pair-wise key with every other node via an intermediate node, where n is the number of sensor nodes in the network. The evaluation of the PIKE scheme [8] shows a sub-linear (i.e., $O(\sqrt{n})$) trend in the communication and memory overheads per node as the network size increases. On the other hand, other schemes [9][15][17] show a linear (i.e., $O(n)$) tendency in either overhead as the network size rises. PIKE is thus scalable, while other schemes are not.

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 pattern can thus be derived. It is possible that this kind of information leakage could cause devastating security leak. The paper [33] proposes SDAP (Secure hop-by-hop Data Aggregation Protocol) for WSNs to resist passive traffic analysis. SDAP is a secure hop-by-hop tree-based data aggregation protocol based on the principles of divide-and-conquer and commit-and-attest. It partitions the aggregation tree into groups logically by reselecting group leaders every time when the sink node wants to collect the sensed data. Each group leader applies commitment-based aggregation to generate a group aggregate and sends it to the sink, which in turn identify the suspicious group according to received group aggregates. At last, each suspicious group participates in an attestation process to prove its group aggregate correctness. In ordinary aggregation process, the node nearest to the sink is the most effective aggregation node and is vulnerable to attacks. In SDAP, group leaders are more effective aggregation nodes. They are hard to locate because they change frequently. In this way, SDAP can provide certain assurance on the trustworthiness of the aggregation results.

There are other effective solutions that use the concept of anonymous communication to resist passive 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 of the network topology and the relationship of communication parties. Several anonymous communication routing protocols for MANETs are proposed in [7] [20][22][30][34][35]. Below, we describe some representatives of them, namely ANODR [22], SDAR [7], AnonDSR [30], and MASK [34].

ANODR [22] is an on-demand anonymous routing protocol based on the "broadcast with trapdoor information" and "boomerang onion" concepts. To find a route to a destination node, a source node floods a route request with a boomerang onion. The destination node ID in the route request is concealed in a trapdoor which only the destination node can open. On receiving a route request, a node tries to open the trapdoor. If the node can open the trapdoor, then it is the destination node. Otherwise, it forwards the route request after updating the onion by using a random key to encrypt the combination of the received onion and a random nonce. When the route request is received by the destination node, the trapdoor is opened and the onion is copied into the route reply to be sent reversely to the source node. The destination node also chooses a route pseudonym and embeds it in the route reply. On receiving the route reply, a node tries to decrypt the onion by the random key just used earlier in the onion encryption. If the decryption is successful and the corresponding random nonce have a match, the node knows that it is on the route and it forwards the reply with the onion peeled off one layer. The node also chooses its own route pseudonym and should keep in the routing table the association between the route pseudonyms of itself and the up-streamer's. It is noted that a node will use its own route pseudonym in forwarding the route reply. The association is used later in data forwarding as follows. When a node receives a data packet with route pseudonym X, it looks up in the routing table to see if there is an association of X and Y. If so, the node replaces X with Y and forwards the packet. Otherwise, it discards the packet. In this way, the data packet can be forwarded to the destination node. ANODR can hide IDs of the source, the destination and the intermediate nodes. It can also prevent adversaries from tracking a packet flow back to source or destination.

In SDAR [7], a node periodically sends out a HELLO message holding the certified public key of the node, and collects other nodes' public keys at the same time. SDAR also applies on-demand route discovery

with trapdoor information to establish routes. A SDAR source node floods a route request with a symmetric key kept in the trapdoor which is a public key encryption. In SDAR, the source node puts a one-time public key TPK in the flooding route request and puts the corresponding one-time private key TSK in the trapdoor. Each route request forwarder records TPK, chooses a random symmetric key K, uses TPK to encrypt K and its own ID, and appends the encrypted data to the route request. When the destination node receives the route request, it opens the trapdoor to find TSK for deriving the symmetric key chosen by every forwarder. It then appends all forwarder IDs and symmetric keys to the route reply message and encrypts this message repeatedly with the forwarders' symmetric keys. It then broadcasts locally a route reply along a reverse path of the route request. Every node along the reverse path removes a layer of the encryption and forwards the reply. When the source node receives the reply, it has the IDs and keys of all the forwarders on the route to the destination. The source node encrypts the data by all forwarders' keys and broadcasts it locally. Each node on the path decrypts a layer and forwards it. When the message reaches the destination node all the encryption layers have been removed and the data can be derived.

AnonDSR [30] employs SPE (Security Parameter Establishment) flooding before the anonymous routing to let the source node share a symmetric key with the destination node. The trapdoor is then set up by the symmetric key. AnonDSR is similar to SDAR. It applies on-demand route discovery with trapdoor information and lets the source node put a one-time public key TPK in the flooding route request and puts the corresponding one-time private key TSK in the trapdoor. AnonDSR uses onion in route request. A route request forwarder selects a random secret key K before forwarding the request. For the forwarder, the onion has two parts. The first part is the selected secret key encrypted by TPK, and the second part is the onion received from route request upstream node with a nonce encrypted all together by K. On receiving the route request, the destination node can open the trapdoor to obtain TSK to peel off all layers of the onion and get all forwarders' secret keys. The procedures of route request forwarding and data forwarding of AnonDSR are similar to those of SDAR; the details are omitted here.

Similar to SDAR, MASK [34] applies proactive neighbor detection. However, MASK allows neighboring nodes to authenticate each other without revealing their real identities. By bilinear pairing, neighboring nodes use dynamically changing pseudonyms (or LinkID) to authenticate each other. At the same time of authentication, the pairing nodes also calculate a common shared key, which is then used to generate a chain of shared keys and common link identifiers for further communication. MASK is also an on-demand routing protocol. Since it does not apply the trapdoor concept, it reveals the destination node ID in the route request. MASK thus lacks destination-anonymity, but has lower computation cost for the flooding route request. On receiving the route request, a non-destination node just forwards the request and keeps the LinkID of the upstream node. When the destination node receives the route request, it sends out a route reply by specifying the LinkID of its upstream node. Note that the route reply is confidential; its content is encrypted by the shared key associated with the LinkID. By changing the LinkID to be that of the upstream node hop by hop, the route reply can reach the source node and the routing tables of all intermediate nodes are set up properly for this route. By the routing table, the source node can send data to the destination node by a procedure similar to the route reply propagation.

3. DESIGN OF ANONYMOUS PATH ROUTING (APR) PROTOCOL

3.1 Design Goals 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. We define the expected goals or properties that APR will achieve as follows:

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 figure out where the packets come from and where the packets go to.
2) *Authentication through anonyms:* Any pair of nodes can authenticate mutually without revealing their identities.

3) *Location privacy:* 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 forwarding:* 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 packets can decrypt the cipher of the packet payload.
6) *Resilience to packet loss*: APR can recover the anonymous communication mechanism after packet loss occurs whether it is caused by packet collision or security attack.
7) *Light-weight computation*: APR can achieve anonymous communication without heavy computation overhead. It is thus suitable for sensor nodes with limited resources.

3.2 Network and Attack Models

We assume that each node has limited radio transmission and reception capabilities. Nodes within the transmission of each other are called neighboring nodes. Non-neighboring nodes communicate with each other via multi-hop wireless links. In addition, we assume that wireless links are symmetric; i.e., if a node can receive the signal from another node, then the latter can also receive the signal from the former.

Because the deployment of sensor nodes is wide-spreading and almost stationary, adversaries can not only eavesdrop on the packets but also compromise some sensor nodes to become internal adversaries. However, we assume that adversaries cannot compromise unlimited number of nodes and that they do not have unbounded computational capabilities to easily invert or read encrypted messages, or break all the pair-wise encryption.

We also assume that sensor nodes are equipped with countermeasures against the in-the-field tampering and/or secret leaking attacks. That is, in case sensor nodes are physically captured, adversaries cannot directly read/write the contents of data and programs stored in SRAM and FLASH memory of the microcontroller. This can be achieved by employing data security protection mechanisms to disable unauthorized access of sensor nodes [4]. For example, the TI MSP430 microcontroller has a security fuse which can be irreversibly blown to disable the reading and writing of arbitrary memory via external circuitry testing devices. After the fuse is blown, the only way to access MSP430 memory is via using a bootstrap loader with a 256-bit password. Any unauthorized operations without this password will cause "mass erase," i.e., erasure of all microcontroller memory. This can ensure that all data and program stored on the sensor nodes will not be exposed even though they are physically captured. However, we do suppose that adversaries can control the radio of a captured node to read, modify, delete and create any messages without accessing to the program or data of the sensor node. More in-depth "node capture" attacks are possible [24]; they are hard to conduct in the field, and are thus excluded from this paper.

3.3 Notation

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
- BID_A: the broadcast identity of node A
- $BSeq_A$: broadcast sequence number of node A
- BK_A: the broadcast key of node A
- EK^i_{AB} (=EK^i_{BA}): the i-th pair-wise key shared with node A and node B used to encrypt packet payload
- EK_{AB} (=EK_{BA}): the set of EK^i_{AB} for $i = 0 \ldots n$
- $E\{K, M\}$: a message M encrypted with a shared key K
- H: a one way hash function
- MK^i_{AB} (=MK^i_{BA}): the i-th pair-wise key shared with node A and node B used to compute MAC (Message

Authentication Code) of the packet

- MK_{AB} (=MK_{BA}): the set of MK^i_{AB} 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
- PID_{SD}: path identity of the routing path between source node S and destination node D

3.4 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 [8] to establish a one-hop pair-wise key K_{AB} and a random nonce rn for a communication link between node A and A's neighboring node B. Besides the pair-wise key, a node A in APR also needs to decide a random broadcast identity BID_A and a random broadcast key BK_A. For each neighboring node B, node A encrypts BK_A with the one-hop pair-wise key K_{AB} and sent it to B right after the pair-wise key establishment. Afterwards, APR establishes two more keys, namely data encryption key EK_{AB} and MAC encryption key MK_{AB}. 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. EK_{AB}, MK_{AB}, $HI_{A\to B}$ and $HI_{B\to A}$ are used only once; they are altered when they are ever used. EK_{AB} and MK_{AB} 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 Eq. (1)). Afterwards, EK_{AB} and MK_{AB} are calculated by hashing their previous values (refer to Eq. (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 Eq. (3)).

$$\begin{cases} EK^0_{AB} = H(K_{AB} \oplus C_1) \\ MK^0_{AB} = H(K_{AB} \oplus C_2) \end{cases} \tag{1}$$

$$\begin{cases} EK^{i+1}_{AB} = H(EK^i_{AB}) \\ MK^{i+1}_{AB} = H(MK^i_{AB}) \end{cases} \tag{2}$$

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

By Eqs. (1), (2) and (3) and the received broadcast key, 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 Seq of the link, the data and MAC encryption keys EK and MK, hidden identities *HI-in* and *HI-out* for the in-bound and the out-bound directions, and the broadcast sequence number $BSeq$, the broadcast identity BID and the broadcast key BK. For example, assume node A has three one-hop neighbors, B, C, and D. After the pair-wise key establishment, A has the initial link table as shown in Table 3.1.

Table 3.1: The initial link table of node A

ID	Seq	$HI\text{-}in$	$HI\text{-}out$	EK	MK	BID	$BSeq$	BK
ID_B	0	$HI^0_{B\to A}$	$HI^0_{A\to B}$	EK^0_{AB}	MK^0_{AB}	BID_B	0	BK_B
ID_C	0	$HI^0_{C\to A}$	$HI^0_{A\to C}$	EK^0_{AC}	MK^0_{AC}	BID_C	0	BK_C
ID_D	0	$HI^0_{D\to A}$	$HI^0_{A\to D}$	EK^0_{AD}	MK^0_{AD}	BID_D	0	BK_D

The link table changes for every communication. For example, the entry in the link table of node A for the link form node A to node B becomes the nine-tuple (ID_B, i, $HI^i_{A\to B}$, $HI^i_{B\to A}$, EK^i_{AB}, MK^i_{AB}, BID_B, j, BK_B) after the i-th packet transmission and j-th B's broadcasting. (Note that the broadcast identity BID_B and the broadcast key BK_B do not change, though.) 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 EK^i_{AB} in the field EK, and prepares a MAC for the packet payload by the key MK^i_{AB} in the field MK. To be more precise, the (i+1)-th packet from A to B is of the form $<HI^i_{A\to B}, E\{EK^i_{AB}, DATA||(i+1)\}, \text{MAC}(MK^i_{AB})>$. 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 EK^i_{AB} and acknowledges the packet by sending an acknowledgement (ACK) packet. Otherwise, it drops the packet. Each sender uses the MAC mechanism to enable the receiver to detect tampering in the message.

Each node A also keeps a broadcast sequence number $BSeq_A$. When node A wants to broadcast a packet locally, it just embeds the broadcast identity BID_A in the packet, increases $BSeq_A$ by one, and then encrypts $BSeq_A$ and the data payload by the broadcast key BK_A. To be more precise, the broadcast packet is of the form $< BID_A, E\{BK_A, BSeq_A, DATA\}>$. On receiving the broadcast packet, a node will decrypt the packet to obtain broadcast sequence number $BSeq_A$ and data payload plaintext by the broadcast key BK_A associated with the broadcast identity BID_A. If $BSeq_A$ is not larger than that stored in the link table, then the packet is discarded. Otherwise, $BSeq_A$ is stored and the data payload plaintext is further processed. Note that there is no acknowledgement for a broadcast packet. Also note that each packet, for either broadcast or unicast that contains either data or control commands, is attached with some padding data of a random size so that adversaries cannot relate two packets that are of roughly the same size and are sent within nearby areas at close time instances.

When a node A is rebooted and loses all the information about previous communications, it can use broadcast key BK_A to broadcast a secure reset packet asking all one-hop neighboring nodes to reset the corresponding link table entries to have initial sequence numbers and keys. To prevent attacks, a challenge-response authentication mechanism based on the broadcast key BK_A should be applied to secure the reset procedure. For example, a neighboring node B of A sends a random number r to node A, and node A uses BK_A to encrypt r and sends the encrypted r to B for B to authenticate A. In this way, the rebooted node can join the network properly.

One-hop acknowledgement

After receiving and authenticating a packet properly, the receiver will send an ACK packet to the sender to acknowledge the packet, and will update the values of the *Seq*, *HI-in*, *HI-out*, *EK*, and *MK* fields of the link table entry associated with the sender by Eqs. (2) and (3). After the sender receives the ACK packet, it will also update the appropriate link table entry by Eqs. (2) and (3). Without packet loss, nodes can then communicate by the new link table setting properly. However, packet loss or 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 two more fields *Old-HI-in* and *Old-MK* in the link table to

store the previous values of the *HI-in* and the *MK* fields, and requires a receiver *B* to send an ACK packet, $<HI^{i}_{B\rightarrow A}, E\{EK^{i}_{AB}, ACK\|(i+1)\}, \text{MAC}(MK^{i}_{AB})>$, to the sender *A* when *B* receives *A*'s (*i*+1)-th packet. On receiving a packet, *B* sends an ACK packet and updates the values of fields *Seq*, *HI-in*, *HI-out*, K_{enc}, K_{mac}, *Old-HI-in* and *Old-MK* of the corresponding link table entry. It is noted that node *B* should keep the last ACK packet sent to each neighbor for possible further use. When node *A* receives the ACK packet, it updates the values of fields *Seq*, *HI-in*, *HI-out*, *EK*, and *MK* of the corresponding link table entry. If *A* cannot receive the ACK packet after a timeout period due to 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. Furthermore, node *B* can authenticate the packet and check the integrity of the packet by the key stored in the *Old-MK* field. Note that node B does not need to decrypt the packet since it has received and decrypted the packet earlier. The receipt 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. Also note that the link table of node *B* is again updated only after node *B* receives node *A*'s packet whose *HI* matches with the value stored in the *HI-in* field. Therefore, in the case of multiple retransmissions due to repeated failures, nodes *A* and *B* still keep the synchronization of their link tables.

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* in APR wants to find a path to a destination node *D* multiple hops away, *S* broadcasts an APR-REQ (anonymous path routing request) packet locally. APR puts a hidden identity HIP instead of a destination node identity in the routing request. HIP represents the hidden identity for the pair of the source and the destination nodes. It is computed by hashing the value of $K_{SD} \oplus ID_S \oplus ID_D$ (see Eq. (4)), where K_{SD} is the pre-distributed pair-wise key and ID_D (resp., ID_S) is the identity of node *D* (resp., *S*).

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

Each sensor node *X* in APR maintains a HIP table right after it is deployed. The node *X*'s HIP table contains an entry for every possible source node that shares a pre-distributed pair-wise key with *X*. And each HIP table 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 APR-REQ is broadcast locally to be flooded throughout the entire network for finding a path from the source node *S* to the destination node *D*. It has the form $<BID_S, E\{BK_S, BSeq_S, REQ, ID_S, HIP_{SD}, RSeq_{S\rightarrow D}\}>$, where $RSeq_{S\rightarrow D}$ is set by *S* to be one more than the last known routing request sequence number from *S* to *D*. There is a field to represent the identity of the node which initiates or rebroadcasts the packet. It is initially set to ID_S, the identity of the source node *S*, and will be replaced by the identity of the node which rebroadcasts the packet.

Every node which receives the APR-REQ packet stores the packet information for checking incoming APR-REQ or APR-REP (anonymous path routing reply) packets. For example, if $RSeq_{S\rightarrow D}$ of an incoming APR-REQ packet is smaller than or equal to that of the stored packet information of the same HIP_{SD}, it means that the incoming APR-REQ is obsolete and should be dropped to avoid forming request cycle. It is noted that due to the limitation of memory capacity in sensor nodes, we have to allocate a fixed-size buffer to temporarily keep request packet information. Once the buffer is full, a simple data replacement strategy, such

as FIFO (first in first out), can be employed to make room for a new request packet. The proper buffer size depends on the information size per request packet, the route request frequency, the network diameter (the maximum number of hops between two sensor nodes), the one-hop communication turn-around time, and the one-hop packet loss rate.

On receiving a valid APR-REQ, a 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 while incoming *APR-REQ* is fresh, or drops the packet while incoming *APR-REQ* is obsolete. Note that all nodes except for the destination node will rebroadcast the APR-REQ packet, causing the broadcast storm problem [25]. We can apply mechanisms proposed in [25], such as the counter-based scheme and the distance-based scheme, to solve such a problem. The readers are referred to the paper [25] for more details. Fig. 3.1 demonstrates an example of the steps (steps 1 to 3) to send APR-REQ from *S* to *D* through nodes *E* and *C*. Note that we use $<REQ, ID_S, HIP_{SD}, RSeq_{S\rightarrow D}>$ in Fig. 3.1 to represent $<BID_S, E\{BK_S, BSeq_S, REQ, ID_S, HIP_{SD}, RSeq_{S\rightarrow D}\}>$ for short. Also note that we don't need to embed ID_S in the APR-REQ packet; it can be inferred from the BID_S entry in the link table. However, we leave it in the packet for the purpose of explaining the protocol easily.

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. 3.1 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 $<HI_{D\rightarrow C}, E\{EK_{CD}, REP||HIP_{SD}||PID_{SD}||RSeq_{S\rightarrow D}||Seq_{CD}\}$, MAC($MK_{CD}$)>. Note that, we omit MAC field in the following context for simplicity. This routing reply contains a path identity PID_{SD} of the routing path between *S* and *D*. Each route path is two-way, so PID_{SD} represents not only the path from *S* to *D*, but also the reverse path from *D* to *S*. PID_{SD} is generated by *D* randomly. However, if there is another path that was routed from *C* to *D* before, PID_{SD} must be different from that of the path.

Each node maintains a two-way routing table consisting of entries with the fields *PID*, *Pre-hop*, *Next-hop*, and *Sour* (*Dest*). A node updates the table when receiving an APR-REP packet. The field *PID* 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 of the same HIP_{SD} and $RSeq_{S\rightarrow D}$ comes from, and *Next-hop* stores the identity of the node from which the 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 PID_{SD} into the *PID* 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*, node *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 EK_{CE} and sent to *E*. Node *C* inserts an entry in its two-way routing table as follows. It puts PID_{SD} in the *PID* 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 PID_{SD} in the *PID* 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. 3.1 (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.

For each multi-hop end-to-end between the source *S* and the destination *D* represented by HIP_{SD}, the sequence numbers $RSeq_{S\rightarrow D}$ and $RSeq_{D\rightarrow S}$ are distinguishable and are given by the source node and the

destination node, respectively. They are used to avoid forming route request cycles. When a node is rebooted and loses all the information about previous communications, it can flood a route request with the largest sequence number to find a route to the desire destination. Such a route request will neither be discarded nor form request cycles. It will also reset sequence numbers of all nodes when flooded throughout the network. On receiving such a route request, the destination node can securely send a reply packet with a reset sequence number to the source node. In this way, a rebooted node can perform route request properly.

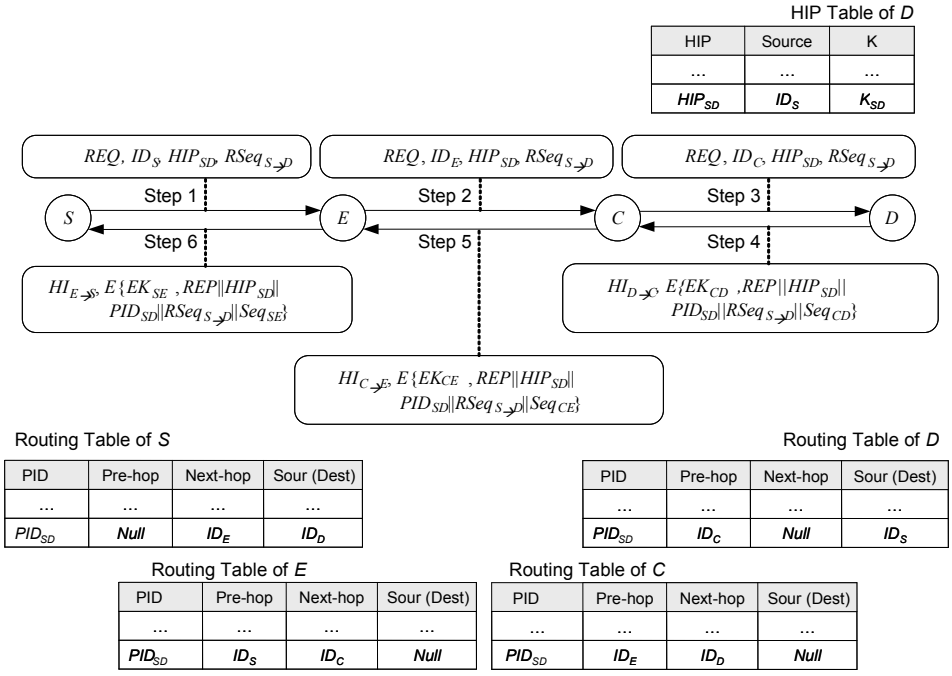


Fig. 3.1. Anonymous multi-hop path establishment from the source *S* to the destination *D*

3) Anonymous Data Forwarding

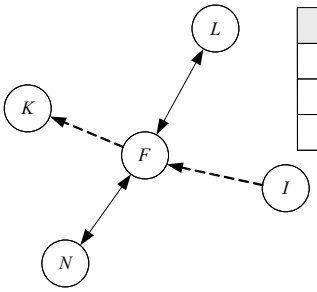
When the source node *S* wants to send a data packet to the destination node *D* multiple hops away after the anonymous path routing establishment (suppose the forwarding node is *E*), it first finds out PID_{SD} and the forwarding node identity ID_E for *D* by looking up the routing table. The identity 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 $<HI_{S\to E}, E\{EK_{SE}, MDT||PID_{SD}||E\{EK_{SD}, DATA\}||Seq_{SE}\}>$, where MDT stands for the packet label for *multi-hop data transmission*. When the forwarding node *E* receives a MDT packet, it looks for the entry of PID_{SD} in its two-way routing table. If the identity stored in the *Pre-hop* (resp., *Next-hop*) field matches with ID_S (the identity of the node from which the packet comes), the next forwarding node *C* 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 EK_{SD} shared by *S* and *D*.

3.5 PID Collision Problem

Since PID 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 PID. We call this the PID 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. 3.2 is an example of case 1. The forwarding node, *F*, has two paths with PID

= 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 PID = 12 from *I* should be forwarded to *K*, and packets with PID = 12 from *L* should be forwarded to *N*.

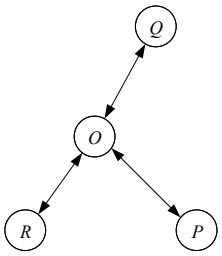


Routing Table of *F*

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

Fig. 3.2. Two different paths with the same PID and different *Pre-hop* nodes

The scenario depicted in Fig. 3.3 is an example of case 2. The forwarding node *O* receives an APR-REP packet with PID = 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 PID = 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 PID 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 PID = 14, it forwards this packet to node *P* by changing PID to 13. Therefore, node *O* will keep PIDs 13 and 14 in the routing table. To indicate that a PID should be changed to a new one, node *O* stores the new PID in *Change* field. If the *Change* field is *Null*, the PID 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 PID = 13 is set to be "*true*", which means this entry should be checked first if node *O* receives a packet with PID = 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.



Routing Table of *O*

PID	Pre-hop	Nex -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*

Fig. 3.3. Two different paths with the same PID and *Pre-hop* node

4. ANONYMITY AND SECURITY ANALYSIS

In this Section, we analyze how well APR meets the design goals. First, we examine the anonymity of APR in single-hop communication and data forwarding. Subsequently, we discuss the attacks that APR is able to defend against.

4.1 Anonymity analysis

Here we discuss two adversary conditions: (1) external eavesdroppers – passive observers who just overhear the passing messages, and (2) internal adversaries – adversaries who have compromised some sensor nodes.

Communicating with the pair-wise hidden identities in one-hop communication guarantees that any two neighboring nodes can establish an anonymous yet secure link without revealing their identifiers. Both routing packets and data packets are locally exchanged between two neighboring nodes with the established HIs (hidden identities) rather than their real identifiers. The HIs do not provide any information to external eavesdroppers, except the two nodes which constitute the link. In APR, except the rebroadcast node ID in the APR-REQ packet, no real node identities appear in any packet after one-hop pair-wise keys are established. The rebroadcast node ID is useless for adversaries because all of local communications use hidden identity. Furthermore, a multi-hop communication path is anonymous since it is identified by PID and its source and the destination nodes are represented by HIP, which can only be identified by the associated source and destination nodes.

If a node is compromised, the internal adversary can capture locally exchanged packets with the hidden identities. Nonetheless, the anonymity of the network except this compromised node never loses, although this compromised node could be used as a gateway to interfere the network's operation by attackers. As our assumptions in section III, the adversaries have unbounded eavesdropping capability but bounded computing and node intrusion capabilities, so compromised sensor nodes could just degrade the anonymity in a proportion of the whole network. As the numbers of compromised nodes increase, the anonymity of network degrades. If we desire to maintain the anonymity to a certain degree, how many compromised sensor nodes can be tolerated in our system? We analyze the relationship between the number of compromised sensor nodes and the degree of anonymity with the entropy-based measurements suggested in [13].

The concept of entropy is a useful tool to measure anonymity of a system under certain attacks [11]. After attacks, some nodes are compromised, while the others still adhere to the original program and behave honestly. The honest nodes constitute the *anonymity set S*. Let X be a discrete random variable with probability mass function $p_i = Pr(X = i)$, where i represents each possible element of anonymity set S. The entropy $H(X)$ of the system with anonymity set S can be calculated as:

$$H(X) = -\sum_{i \in S} p_i \log p_i \tag{5}$$

Let n be the size of the set N of sensor nodes. The system has maximum entropy when there is no attack and all nodes are honest. The maximum entropy for a system of n nodes is

$$H_M = \log n \tag{6}$$

Let s be the size of the anonymity set S (or the number of uncompromised nodes). We assume that the attacker cannot assign different probabilities for the nodes in S. We have

$$\begin{cases} p_i = 0, \ \ for\ i \in (N - S) \\ p_i = \dfrac{1}{s}, \ for\ i \in S \end{cases} \tag{7}$$

According to [13], the degree d of anonymity of the system is defined to be

$$d = 1 - \frac{H_M - H(X)}{H_M} = \frac{H(X)}{H_M} = \frac{\log s}{\log n} \tag{8}$$

Fig 4.1 shows the degree of anonymity for different *s* values with *n*=300. Obviously, degree of anonymity decreases when *s* increases. By the figure, we can find out the required number *s* of uncompromised nodes under a given degree of anonymity. For example, if we want to obtain a degree of anonymity larger than 0.8, we need to keep $s > 100$.

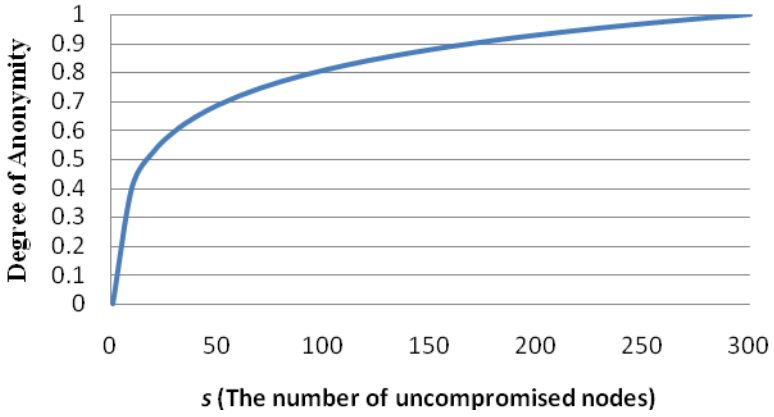


Fig 4.1 Degree of anonymity for APR

4.2 Security analysis

Due to the anonymity, APR can resist the following attacks:

- *Tampering attacks*

This might be the result of physical capture and result compromise. A. Becher et al. [4] pointed out that a sensor node has no user interfaces, which is different from other form of mobile wireless communication devices. Invasive attacks require access to a chip's internals, which can be designed with hardware/software protection mechanisms. As sensor nodes operate unattended and cannot be made tamper proof because they should be as cheap as possible. The in-the-field node capture stack is not so easy as usually assumed in the literature. They require expert knowledge, costly equipment and other resources, and, most important, removal of nodes from the network for a non-trivial amount of time [4]. At the moment, there have been some hardware and software solutions for developments of sensor nodes to reduce the impact the captured/compromised nodes arise [4][24]. Depending on the efforts attackers make, node capture attacks can have significant impact. The more in-depth "node capture" attacks is possible but we intend to exclude them. However, the countermeasures against physically capture of mobile communication device is a profound and extensive research field.

- *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. The network topology is thus concealed. In multi-hop communication, the source node utilizes the broadcast key to encrypt route request information. Furthermore, APR utilizes HIP to specify the destination node. Since HIP can be recognized only by the source node and the destination node, the source and the destination node identities are concealed. When establishing the routing table and forwarding data, PID is used instead of HIP. All intermediate nodes in a routing path forward data according PID, and they only know next-hop and pre-hop nodes' hidden identities associated with the PID. As we have mentioned, each packet is attached with some padding data of a random size so that adversaries cannot relate two packets that are of roughly the same size and are sent within nearby areas at close time instances. Therefore no nodes, except the source and the destination nodes, can figure out the

end-to-end relationship of any multi-hop communication. And the adversary cannot track any packet to the source or to the destination node. 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 packet 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 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 only once for unicast transmission; it is updated every time when a packet is proper received. In order to maintain the synchronization of link table entries between neighboring nodes, APR keeps the last *HI*. Therefore, if the sender re-sends the packet when not receiving the corresponding ACK packet, the receiver can recognize the re-sent packet and issue the ACK packet properly. If adversaries replay a data packet, this packet will be regarded as a resent packet and cause the receiver to send the ACK packet to the sender. Fortunately, the replayed packet will not make the corresponding link table entries of the sender and the receiver out of synchronization. Furthermore, after the sender starts to send a new packet, the replayed packet will be discarded. This is because the receiver will adjust its link table and update the last *HI* when receiving the new packet from the sender. For broadcast transmission, a node decrypts the packet to obtain and record *BSeq* for the sender when receiving a broadcast packet. The packet will be discarded when its *BSeq* is not larger than that stored in the link table. A replayed broadcast packet will obviously be discarded. 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 and different encrypted BSeqs. Without correct HIs or BSeqs, DoS attack packets will be ignored directly. APR can thus limit the damage caused by this kind of attacks to the one-hop communication area.

5. IMPLEMENTATION AND OVERHEAD EVALUATION

In order to provide strong security protection and good scalability, APR combines symmetric key cryptographic algorithm, hash function and message authentication code (MAC) together. We use Skipjack block cipher algorithm [23] as the symmetric key cryptosystem, SHA-1 [16] as the hash function, and CBC-MAC [5] as the MAC function.

We have implemented APR 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. 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. 5.1 for a WSN with 200 sensor nodes randomly deployed in a $5R \times 5R$ field, where R is the communication range of sensor nodes. For a node needing to find a route path to another which is 7 hops away, the average time of path establishment is about 574 milliseconds.

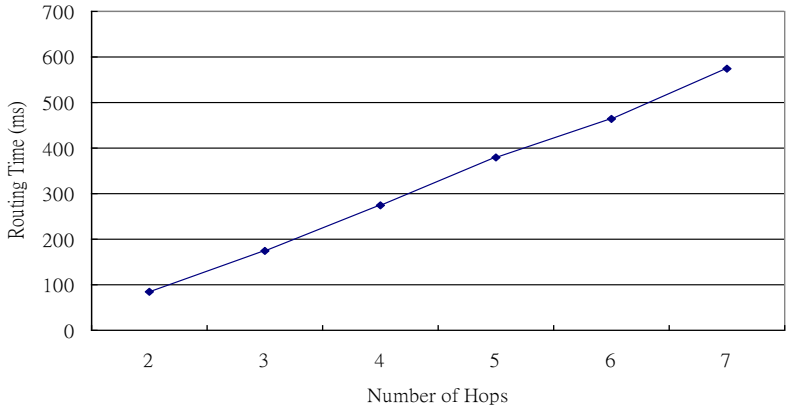


Fig. 5.1. Average routing latency of APR

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. Since HIP table is loaded in the program space of the flash memory before nodes are deployed, 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.

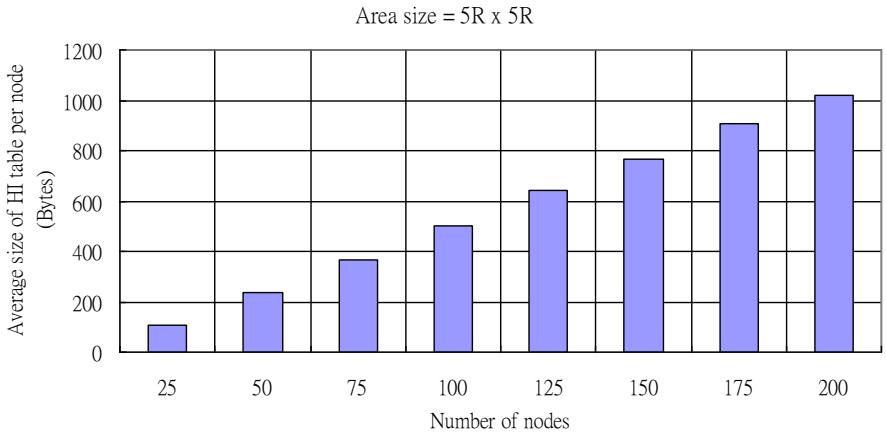


Fig. 5.2. Average link table size of a node in a $5R \times 5R$ field

Figure 5.2 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. 5.3, 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.

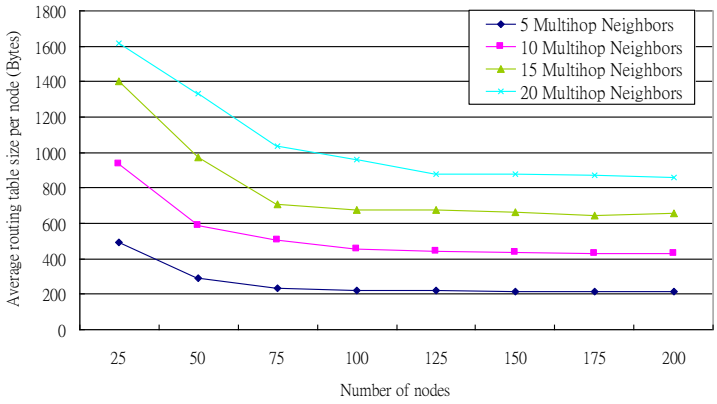


Fig. 5.3. Average routing table size of a node in a $5R \times 5R$ field

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. 5.4. The maximum memory overhead of APR shown in Fig. 5.4 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.

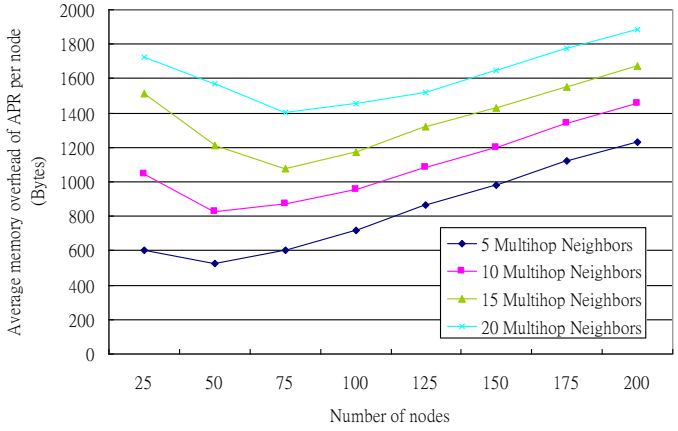


Fig. 5.4. Average memory overhead of a node for varying number of nodes in a $5R \times 5R$ field

Below, we evaluate extra overheads of employing APR in terms of communication and computation overheads, which are relevant to energy consumption of the protocol and the network lifetime. Since APR is similar to AODV (Ad-Hoc On-Demand Distance Vector Routing) protocol [26], we perform the evaluation by comparing APR to AODV. Comparing with ADOV, the extra communication and computing overheads in APR are incurred mainly in the following three aspects:

(1) Anonymous one-hop communication: In the bootstrap phase of APR, a sensor node establishes shared keys K^0_{enc} and K^0_{mac} for each neighbor via 2 hashing and 2 XOR operations. Subsequently, in ordinary communication phase, the node updates the keys K^i_{enc} and K^i_{mac} and the pair of hidden identities ($HI^i_{in\text{-}bound}$ and $HI^i_{out\text{-}bound}$) by 4 hashing, 4 XOR and 2 multiplication operations. In APR, two pairs of shared key encryption and decryption operations (one pair for processing data and the other for MAC code) are required for one-hop communication. APR employs ACK packets to synchronize the sender's and receiver's sequence number in each communication session, which leads to an additional one-hop

communication overhead of transmitting a packet and the associated computation overhead.

(2) Anonymous multi-hop routing path establishment: In this aspect, APR needs an extra shared key encryption for broadcasting route request locally, and an extra shared key decryption for each neighbor to receive the request. On receiving a route request, each node needs to search its HIP table to see if test if itself is the destination node. In APR, delivering a route reply between a pair of consecutive intermediate nodes on the route path incurs extra computation overheads of encrypting and decrypting the route reply packet, and incurs extra communication overheads and the associated computation overheads of transmitting the corresponding ACK packet.

(3) Anonymous data forwarding: Forwarding data in APR between a pair of consecutive intermediate nodes on a route path incurs extra computation overheads of encrypting and decrypting the data packet, and incurs extra communication overheads and the associated computation overheads of transmitting the corresponding ACK packet. When two different route paths have the same path ID coincidentally, extra computation overheads are incurred to solve the problem caused by the coincidence.

Like other anonymous routing protocols, namely ANODR [22], SDAR [7], AnonDSR [30] and MASK [34], APR applies extra cryptographic operations and extra packets to achieve anonymity. Extra computation and communication overheads are thus incurred. Fortunately, most of the extra overheads incurred in APR are computation overhead. As shown in [29], the cost of transmitting one bit is on the order of thousand times the cost of processing one bit. Therefore, we can conclude that APR does not trade unaffordable extra overheads or energy consumption for communication anonymity.

6. CONCLUSION

In this paper, we propose an anonymous path routing (APR) protocol for secure data communication of WSNs. APR uses 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 between itself and one of its neighbors. One identity is for the in-bound direction of the link; the other, the 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 in an anonymous way and assign a pseudonym, called PID, to the path for identification. Only the source and destination nodes know each other's identity. The third scheme can forward packets in an anonymous way by using the PIDs.

In APR, data are encrypted by pair-wise keys and transmitted with anonyms (HIs) between neighboring sensor nodes and anonyms (HIPs) 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. As we have shown, APR can resist several types of attacks, such as the traffic analysis attack, forging attack, replay attack, and DoS attack. We have implemented APR on MICAz sensor device with TinyOS operating system for overhead evaluation to demonstrate that APR is applicable to practical WSNs.

REFERENCES

1. Ian. F. Akyildiz and Xudong Wang, "A Survey on Wireless Mesh Networks", *IEEE Communications Magazine*, vol. 43, no. 9, pp. 23-30, Sept. 2005.
2. F. Anjum and S. Sarkar, "Security in Sensor Networks", *Mobile, Wireless, and Sensor Networks: Technology, Applications, and Future Directions*, edited by R. Shorey et al., pp.283-307, John Wiley & Sons, 2006.
3. S. Basagni, K. Herrin, E. Rosti and D. Brushi, "Secure pebblenets", in *Proceedings of MobiHoc*, 2001.

4. A. Becher, Z. Benenson, and M. Dornseif, "Tampering with motes: Real-world physical attacks on wireless sensor networks", in *Proceedings of the 3rd International Conference on Security in Pervasive Computing (SPC)*, April 2006.
5. 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.
6. D. Boneh and M. Franklin, "Identify-based Encryption from Weil Pairing", *SIAM Journal on Computing*, Vol.32, No.3, pp. 586-651, March 2003.
7. 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.
8. 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.
9. W. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for Sensor networks", in *Proceedings of IEEE Symposium on Security and Privacy*, May 2003.
10. Jiming Chen, Yanping Zhang, Xianghui Cao, and Youxian Sun, "A Communication Paradigm for Wireless Sensor/Actuator Networks", in *Proceedings of 2007 International Conference on Sensor Technologies and Applications*, 2007.
11. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley and Sons, Inc., 1991.
12. David Culler, Deborah Estrin, Mani Srivastava, "Overview of wireless sensor networks", *IEEE Computer, Special Issue in Sensor Networks*, August 2004.
13. C. Diaz, S. Seys, J. Claessens and B. Preneel, "Towards measuring anonymity", in *Proceedings of PET*, 2002.
14. F. Dressler, Y. Guan and Z. Jiang, "Wireless and Sensor Networks: A Retrospection", in *Proceedings of IEEE International Conference of Mobile Adhoc and Sensor Systems*, pp.1-6, 2007.
15. W. Du, J. Deng, Y. Han, and P. Varshoey, "A pairwise key pre-distribution scheme for wireless sensor networks", in *Proceedings of the Tenth ACM Conference on Computer and Communications Security (CCS 2W3)*, pp. 42-51, October 2003.
16. D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", in *IETF Request for Comments 3174*, 2001.
17. L. Eschenauer and V. Glisor, "A key management scheme for distributed sensor networks", in *Proceedings of the 9th ACM Conference on Computer and Communication Security*, pp. 41-47, 2002.
18. 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.
19. 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.
20. 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.
21. D. K. Kim, "A New Mobile Environment: Mobile Ad Hoc Networks (MANET)", *IEEE Vehic. Tech. Soc. News*, pp. 29-35, August 2003.
22. 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.
23. Y. W. Law, J. Doumen, and P. Hartel, "Survey and Benchmark of Block Cipher for WSNs", *ACM Transaction on Sensor Networks (TOSN)*, Vol. 2, No. 1, pp. 65-93, Feburary 2006.
24. K. Makki et al., *Mobile and Wireless Network Security and Privacy*, Springer, 2007.

25. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999.
26. C. E. Perkins and E. M. Royer, "Ad-Hoc On-Demand Distance Vector Routing", in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, 1999.
27. A. Perrig, R. Szewcczyk, V. Wen, D. Culler and J.D. Tygar, "SPINS: Security protocols for sensor networks", in *Proceedings of Mobile Computing and Networking Conference*, Rome, Italy, 2001.
28. J. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems", in *Proceedings of International Workshop on Designing Privacy Enhancing Technologies*, pp. 10-29, California, USA, January 2001.
29. Brian Schucker et al., "Embedded Operating Systems for Wireless Microsensor Nodes", *Handbook of Sensor Networks: Algorithms and Architectures*, edited by Ivan Stojmenovic, pp.417-456, John Wiley & Sons, 2005.
30. 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.
31. Ivan Stojmenovic and Stephan Olariu, "Data-centric Protocols for Wireless Sensor Networks", *Handbook of Sensor Networks: Algorithms and Architectures*, edited by Ivan Stojmenovic, pp.417-456, John Wiley & Sons, 2005.
32. Mo Wei, Genshe Chen, Cruz, J.B., Jr., L. Hayes, and Mou-Hsiung Chang, "A decentralized approach to pursuer-evader games with multiple superior evaders", in *Proceedings of 2006 IEEE Intelligent Transportation Systems Conference (ITSC'06)*, pp. 1586 – 1591, 2006.
33. 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.
34. 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.
35. 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.

Jehn-Ruey Jiang (江振瑞) received his Ph. D. degree in Computer Science in 1995 from National Tsing-Hua University, Taiwan, R.O.C. He joined Chung-Yuan Christian University as an Associate Professor in 1995. He joined Hsuan-Chuang University in 1998 and became a full Professor in 2004. He is currently with the Department of Computer Science and Information Engineering, National Central University, Taiwan. He was a recipient of Best Paper Award of the 32nd International Conference on Parallel Processing, 2003, and editors of Journal of Information Science and Engineering and International Journal of Ad Hoc and Ubiquitous Computing in 2004 and 2005, respectively. He has organized the 1st, 2nd and 3rd International Conference on Peer-to-Peer Networked Virtual Environments in 2007, 2008 and 2009, respectively. His research interests include distributed algorithms, algorithms for peer-to-peer networks, algorithms for mobile ad hoc networks and algorithms for wireless sensor networks.

Jang-Ping Sheu (許健平) received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively.

He is currently a Chair Professor of the Department of Computer Science, National Tsing Hua University. He was a Chair of Department of Computer Science and Information Engineering, National Central University from 1997 to 1999. He was a Director of Computer Center, National Central University from 2003 to 2006. His current research interests include wireless communications and mobile computing. He was an associate editor of the IEEE Transactions on Parallel and Distributed Systems, Journal of the Chinese Institute of Electrical Engineering, Journal of Information Science and Engineering, Journal of the Chinese Institute of Engineers, and Journal of Internet Technology. He is an associate editor of the International Journal of Ad Hoc and Ubiquitous Computing and International Journal of Sensor Networks.

He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993-1994, 1995-1996, and 1997-1998. He received the Distinguished Engineering Professor Award of the Chinese Institute of Engineers in 2003. He received the certificate of Distinguished Professorship, National Central University in 2005. He received the K. -T. Li Research Breakthrough Award of the Institute of Information and Computing Machinery in 2007. He received the Y. Z. Hsu Scientific Chair Professor Award in 2009. Dr. Sheu is an IEEE Fellow, a member of the ACM, and Phi Tau Phi Society.

Ching Tu (涂靖) received the Master degree in Computer Sciences and Information Engineering from the National Central University, Taiwan, R.O.C., in 2007. His research interests include wireless sensor networks, security, and routing.

Jih-Wei Wu (吳季偉) received the B.S. degree in management science from National Chiao Tung University, Taiwan, R.O.C., in 1994, and the M.S. degrees in information management from National Cheng Kung University, Taiwan, R.O.C., in 1996. He is currently working toward the Ph.D. degree in the Department of Computer Science and Information Engineering, National Central University, Taiwan, R.O.C. His research interests include wireless sensor networks, security, and distributed networks.