

A New Approach to Anonymous Multicast Routing in Ad Hoc Networks

Lichun Bao, University of California, Irvine, U.S.A.

Abstract— **Anonymity in ad hoc network routing came as a means to hide the identification information of nodes, traffic, paths and network topology, which is an effective counter-attack measure to a number of attacks such as traffic analysis, spoofing and denial of services. In this paper, we propose AMUR, an Anonymous MULTicast Routing protocol, for ad hoc networks. AMUR uses Bloom filters to encode source multicast tree in each multicast packet to provide anonymity of nodes, links, routing tables, and source routing trees for multicast purposes. We specify the AMUR protocols, and investigate its robustness against network mobility and various attacks.**

I. INTRODUCTION

Unicast and multicast routing in ad hoc networks presents a number of challenging problems that are different from wired networks [1]. First, ad hoc networks are deployed on demand and potentially in large numbers of nodes. Secondly, wireless transmissions are based on shared media, thus vulnerable to eavesdropping, intrusion and denial of service attacks. Anonymous routing in network communication prevents address spoofing, route forgery, Sybil attack, birthday attack and certain denial of service (DoS) attacks.

Many solutions exist for the Internet and ad hoc networks, however, mostly focusing on unicast routing anonymity. In the Internet, several network-based anonymity approaches provide anonymous communication between end-nodes, including DC-nets [9], Crowds [27], MIX networks [8], and Onion Routing [16]. Crowds network is a randomized packet forwarding network that obscures the packet senders by randomly throwing each packet between Crowd network nodes for a certain number of times before delivering to the final destination. The drawbacks of Crowds are that it provides no recipient anonymity, and no anonymity against a global attacker or a local eavesdropper [27].

MIX networks and *Onion Routing* use cryptographic algorithms to establish anonymous paths as well as encrypting the data packets. In Onion routing, to construct an anonymous path, a packet origin must store and maintain information about the topology of the Onion router network, out of which a number of Onion routers are selected by the packet origin to construct the path. In addition, each node on the path is supplied with a symmetric key for decrypting data packets and getting instructions for the next hop router in the path. When a data packet is ready to send, the source recursively encrypts the packet and the next hop information with the symmetric keys along the path. This creates a layered structure, in which it is necessary to decrypt all outer layers of the onion in order to reach an inner layer.

In ad hoc networks, efficient anonymity is still an elusive topic. Similar approaches to Onion routing are used in wireless ad hoc networks, such as ANODR [19]. ASRP (Anonymous Secure Routing Protocol) [10] and MASK [36] provided anonymous and authenticated on-demand routing by setting up virtual circuits between neighbors in ad hoc networks.

However, keeping up-to-date information about the topology of the network is complex in the presence of dynamic topology.

Therefore, a *trapdoor* mechanism is used to collect the source route between communicating pairs. A trapdoor is a common concept in cryptographic functions that defines a one-way function between two sets [32]. A *global trapdoor* is an information collection mechanism in which intermediate nodes may add information elements, such as node IDs, into the trapdoor. Only certain nodes, such as the source and destination nodes, can unlock and retrieve the elements using pre-established secret keys. The design of a global trapdoor requires end-to-end key agreement between the source and destination nodes. Global trapdoors are used in several proposals, such as SDAR (Secure Distributed Anonymous Routing) [5], AnonDSR (Anonymous DSR) [31] and SDDR (Secure Dynamic Distributed Routing) [14].

Unfortunately, anonymous multicast routing has not been fully addressed. Existing solutions have adapted Onion routing for anonymous multicast purposes [11], [35].

We extend our previous results on anonymous unicast routing approach [33], and propose AMUR (Anonymous MULTicast Routing) that utilize Bloom filters for multicast purposes in ad hoc networks. Using Bloom filters and Diffie-Hellman key exchange protocols, AMUR provides a multicast routing model that maintains anonymous routing information regarding the network node, link and multicast tree information. AMUR is scalable and storage-, processing- and communication-efficient, making it suitable in the ad hoc network environment.

It is worth noticing that Castelluccia *et al.* used Bloom filters to compress source route information after the source route is discovered using DSR [7], [17]. Xi *et al.* also leveraged Bloom filters for random walk routing [34], but not for source routing purposes as AMUR does. In both works [7], [34], anonymity is neither claimed or achieved.

The rest of the paper is organized as follows. Section II describes AMUR data-plane functionalities, such as the node naming, addressing, multicast forwarding, and how these functionalities are combined in AMUR. Section III details the network control-plane operations for multicast tree maintenance. Section IV evaluates AMUR. Section V concludes the paper.

II. DATA PACKET FORWARDING IN AMUR

A. Network Assumptions

We assume that wireless ad hoc networks consist of resource-constrained nodes that are deployed wide range of terrains and communicate over multi-hop distances.

In this paper, we refer *anonymity* by the following connotations:

- *Identity anonymity* is to establish and maintain routes under the condition that only connected nodes know one another's identity.
- *Location/topology anonymity* is that a node's approximate location be concealed.
- *Routing anonymity* is that any forwarding node cannot determine the identity of any other node on the route; including the sender and intended recipient.

Most of the anonymity in our protocols is provided by Bloom filters [4]. A Bloom filter is a space-efficient probabilistic data

This work was supported in part by NSF under grant CT-0524050 and UCI Ted and Janice Smith Funds.

structure that tests whether or not an element is a member of a set. The crucial parameters of a Bloom filter include its table size, *i.e.* the number of bits N in the filter, and the number of indexing keys k to search the filter. Traditional hash table is a special case of Bloom filter where $k = 1$. Bloom filter performance is evaluated by the *false positive rate* and the *false negative rate* as elements are added or removed from the filter. Many variants of Bloom filters have been proposed. A complete survey about Bloom filters is in [6].

A.1 Packet Format

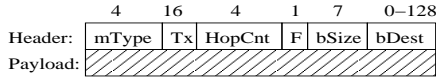


Fig. 1. AMUR Generic Packet Format.

Fig. 1 illustrates the generic packet format used in AMUR, in which **mType** is the message type, **Tx** is the transmitter, **HopCnt** is the hop count, which is a distance limiting factor similar to the TTL field in IP packets, **F** is the forwarding flag, **bDest** indicates the destination(s), **bSize** is the size of **bDest** in bits, and **payload** is the payload field. The sample field sizes in the AMUR packet header are denoted atop each field. Field **mType** takes value from set

{DATA, TOPO, JOIN, LEAVE},

of which type **DATA** means the packet carries multicast data, type **TOPO** is used to exchange topology information, and **JOIN** **LEAVE** types are used to find out multicast tree of the network.

Especially in Fig. 1, the destination **bDest** field is a Bloom filter, in which the complete source multicast tree is encoded by storing the links of the multicast tree into the Bloom filter.

It is well-known that a Bloom filter of size N with k indexing keys can hold up to $n = N/(-\ln P)$ elements, when the false positive rate is P , and the parameter k is set to around $(-\log_2 P)$ [4]. Therefore, considering a **bDest** field of 96-bit vector, combining two Ethernet MAC addresses, we can store a multicast tree with $n = 96/(-\ln 0.01) = 21$ links in the **bDest** field of an AMUR packet with false positive rate of $P = 0.01$ and $k = -\log_2 0.01 = 7$ indexing keys.

Furthermore, the **F** field is a 1-bit flag indicating the traversing direction over the multicast tree. When **F=1**, the packet travels away from the multicast source. When **F=0**, the packet goes towards the multicast source.

In addition, the **Tx** field of Fig. 1 is also a Bloom filter that contains the links from the transmitter to the intended receivers. We will discuss how **bDest**, **F** and **Tx** fields are used in packet forwarding specifications, shortly.

B. Anonymity of the Source Multicast Tree in AMUR

The multicast links in AMUR packets are encrypted using shared keys derived from the Diffie-Hellman algorithm [13], [24]. According to the Diffie-Hellman algorithm, each node generates its own private random number X , and the public Y values are broadcast to its neighbors. After a node gets all its neighbors' Y values and the corresponding shared keys are derived from Diffie-Hellman algorithm, its links can now be encrypted using the shared keys between itself and the other end-points. For AMUR, such key agreement is done infrequently.

Using Diffie-Hellman algorithm and Bloom filter, AMUR provides two integral levels of link anonymity:

1. Link encryption using the shared keys;

2. When a multicast link is stored in the **bDest** fields, the link is stored a Bloom filter. Thus, the link even does not appear as cipher text, thwarting any attempt of decrypting the link.

For clarity in our presentations, we refer to links in their plain-text forms in our discussions.

C. Packet Forwarding Algorithm

As we said, the **bDest** field of an AMUR data packet contains all the links of the multicast tree. Therefore, when an AMUR data packet arrives at node i , *i.e.* **mType**=DATA, the packet is examined field-by-field according to the following algorithm:

1. If **Tx** field contains a link with node i as the tail of the link, forward the packet to upper layer, then continue to the next step; otherwise, stop.
2. If **F=1**, examine whether any link with node i as the head and one of node i 's neighbors as the tail is contained in **bDest** field. If so, decrement **HopCnt**, and check whether **HopCnt**=0. If yes, the packet is dropped, if else, forwarded the packet to all neighbors. Similarly, if **F=0**, links with node i as the tail are examined.

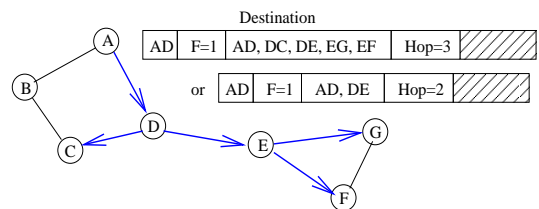


Fig. 2. Multicast Routing Model.

Fig. 2 illustrates an example for multicast routing operations, in which the complete path information is encoded in the **bDest** address field of the multicast packet using Bloom filters, *i.e.*, the packet's **bDest** field contains the complete multicast tree information in the form of links (A, D) , (D, C) , (D, E) , (E, G) and (E, F) . The forwarding flag is set to **F=1**, indicating that the packet is running from A to G , and that the outgoing links at each hop should be examined when each node makes the forwarding decision. When the forwarding flag **F=0**, the packet travels in the reverse direction from leaves to the root of the multicast tree, and the incoming links are examined for "hits" in the **bDest** field, instead. The **HopCnt** is useful for distance-limiting purposes.

In addition, there is another way to encode the multicast tree as shown in the second packet in Fig. 2, where the leaf nodes are considered subscribers of the multicast, and managed by multicast group leaders in the tree.

III. MULTICAST ROUTING CONTROL IN AMUR

In order to facilitate the multicast routing tree establishment, we proactively maintain network topology information necessary for routing purposes. Therefore, routing control protocols in AMUR consist of routing information maintenance and multicast subscription services (**JOIN** and **LEAVE** operations).

A. Routing Information Maintenance

There are two approaches for routing information maintenance — one is the *reactive routing approach*, such as multi-threaded route search DSR [17] and AODV [29], or single-threaded path discovery using gossiping [18], [23], parametric probabilistic routing [3] or random wandering [30]. The other is to use *proactive routing approach*, such as WRP [25], and TBRPF [26]. There are pros and cons in either approach, and we explore the proactive approach in AMUR in this paper.

A.1 Asymmetric Bloom Filters for Storing Routing Information

Our proactive routing information maintenance is based on the *attenuated Bloom filter* (ABF) [28]. ABFs were used by the OceanStore peer-to-peer system for finding files in the Global-Scale Persistent Storage [20]. In OceanStore, a node establishes an ABF for each neighboring peer. An ABF of depth D is an array of D normal Bloom filters for recording objects at different hop distances. The first Bloom filter records objects contained locally. The i -th Bloom filter is the merger of all Bloom filters for all of the nodes at distance i through any path starting with that neighbor link. A query for an object is routed along the edge whose filter indicates the presence of the object at the smallest distance.

AMUR uses similar Bloom filters, called **TopoBFs** (Topology Bloom Filters) and maintained by the *neighbor protocol*, to store next hop nodes and distances to network nodes. However, the application of attenuated Bloom filter in wireless ad hoc networks is different from OceanStore in that the ad hoc network topology constantly change, so does the distance. Therefore, we propose a new algorithm that allows the element removal in TopoBFs, which is not done previously.

TopoBFs are *asymmetric* in that the numbers of indexing keys used in read and write operations are different, which we explain in the following. In contrast, the normal Bloom filters are referred to *symmetric Bloom filters*.

In AMUR, the neighbor protocol of a node collects and saves the **TopoBF** of *each* of its neighbors, and generates its own **TopoBF** by aggregating the **TopoBFs** of its neighbors. The neighbor protocol periodically propagates its **TopoBF** to neighbors via the *HELLO* message of type $mType=TOPO$.

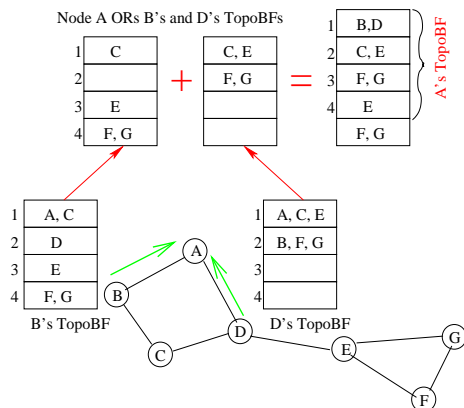


Fig. 3. **TopoBF** Maintenance.

Fig. 3 illustrates the generation of **TopoBF** at node A. After node A receives the **TopoBF** updates from nodes B and D as indicated by the green arrows, node A removes redundant entries in the tables, respectively. For instance, routing information to node A in node B's **TopoBF** is useless at node A. Therefore, node A looks through node B's **TopoBF** from level 1 to D, and removes the entry for routing to node A by setting the bits to 0 corresponding to A's index keys. After these cleanups, a simple "OR"-operation on the modified **TopoBFs** of nodes B and D produces node A's **TopoBF** table, plus the first Bloom filter containing nodes B and D.

Obviously, such cleanup operation introduces *false negatives* to certain destinations. To reduce the false negative rates, we propose to carry out *asymmetric* write and read operations on the **TopoBF**. That is, if we used k index keys to store an element in the Bloom filter, but we use any $k - t$ bits of the aforementioned k keys to infer membership of an element in the

Bloom filter. In addition, whenever an element is a "hit" in the Bloom filter, the element is re-inserted into the Bloom filter to *repair* the Bloom filter.

It is easy to see that each node only knows one-hop neighbor information in **TopoBF**, but does not know who are at two hops and beyond. Therefore, the routing table maintains the anonymity and privacy goals in wireless ad hoc networks.

A.2 Asymmetric Bloom Filter Evaluation

To evaluate the asymmetric behaviors of **TopoBF**, we carried out four experiments to compare a few design choices of the asymmetric Bloom filter algorithm. The experiments are based on Bloom filters of size $N = 4096$ with false positive rate $P = 0.01$. Therefore, the index key size is $-\log_2(P) = 7$, and the maximum number of elements in the Bloom filters should be about $N/(-\ln P) = 889$ [4]. In the experiments, element identifiers 1...3000 are continuously inserted, incrementally filling the Bloom filter until its saturation rate increases just above 50%, the "Golden" threshold [21]. Then, old elements with the lowest IDs are removed, and new higher IDs are again inserted so that the saturation rate crosses the 50% threshold, twice. At every upward threshold-crossing point, statistics are collected regarding the number of current elements in the Bloom filter, false positive and false negative rates. The "repair" operations rewrite one third of the elements in two of the experiments at each statistics collection point.

Fig. 4 presents the three metrics, respectively. The numbers in the legends indicate the numbers of write and read index keys, e.g., "7-7" means seven write and seven read index keys are used in the traditional Bloom filters, while "7-6" means seven write and six read index keys are used in the Bloom filters, which is an asymmetric operation. "Repair" and "NonRepair" mean whether the repair operations are carried out.

As we can see from Fig. 4, "Repair" operation allows both symmetric and asymmetric Bloom filters to maintain the same constant numbers of elements (about 400), and almost the same lowest false rates (about 0%). Under the "NonRepair" option, both "7-6-NonRepair" and "7-7-NonRepair" Bloom filters contain less elements and high false negative rates in the long run. "Repair" or "NonRepair" makes no big different to false positive rates in either symmetric or asymmetric Bloom filters. It is worth noting that "NonRepair" option is closer to real-life asymmetric Bloom filter operation in ad hoc networks.

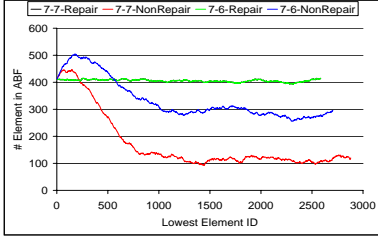
B. Multicast JOIN Protocol

Many multicast routing protocols have been proposed for mobile ad hoc networks under two basic approaches: source-based and core-based multicast protocols. Source-based multicast protocols, such as distance vector based multicast routing protocol DVMRP [22], [29] and PIM-DM (Protocol Independent Multicast - Dense Mode) use the flooding and pruning mechanisms to build multicast trees. CBT (core-based trees) [2], [12], CAMP (Core-Assisted Mesh Protocol) [15] and PIM-SM (Protocol Independent Multicast - Sparse Mode) address the multicast scalability problem by maintaining the core or rendezvous points for senders and receivers.

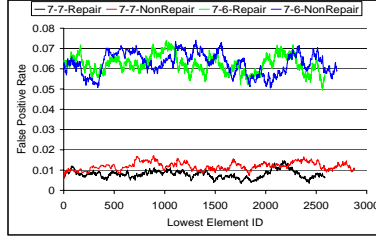
In AMUR, we assume that the multicast source is known *a priori*, and describe how a node discovers the path to the source of the multicast tree using JOIN messages.

Fig. 5 shows how the JOIN request of node E is fulfilled in five steps, supposing that the multicast source is node A, and that a partial multicast tree already exists, consisting of nodes A, C, F and G:

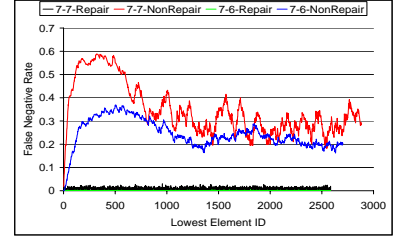
(1) Node E sends the JOIN request message with forwarding flag



(a) The Number of Elements



(b) False Positive Rate



(c) False Negative Rate

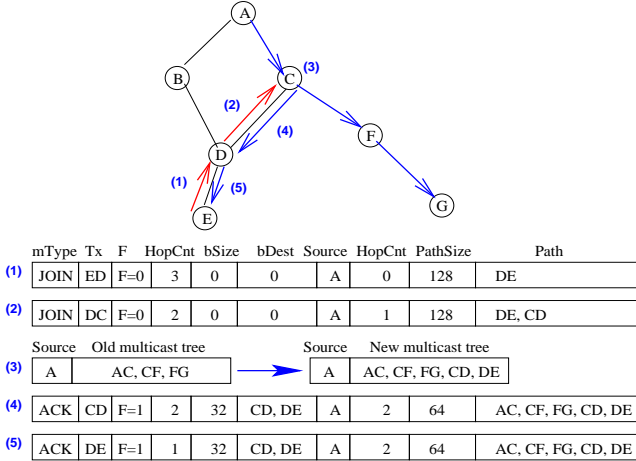
Fig. 4. The $N = 4096$, $k = 7$ Bloom Filter Behavior with Asymmetric Number of Write/Read and Repairing Options.

Fig. 5. Multicast Tree JOIN Message Processing.

$F=0$ to node D , because node D is closer to the multicast source A according to node E 's topology Bloom filter **TopoBF**. In addition, the **payload** field of the JOIN request message carries the traversed path of the message in the form of the same Bloom filter as the **bDest** field. In this case, the path is represented by a bit-vector **Path** of size **PathSize**.

(2) Node D looks up its own topology information table **TopoBF**, inserts another link (C, D) to the **payload** field of the JOIN request message, and keeps forwarding the request to node C .

(3) After node C receives the JOIN request, the original multicast tree is augmented with the multicast subtree by XORing the Bloom filter storing the original multicast tree with the Bloom filter stored in the **payload** field of the JOIN request message.

(4) The new multicast tree is stored in the **payload** field of the JOIN acknowledgment message, where $F=1$ and **bDest** = $\{(C, D), (D, E)\}$. The acknowledgment message is sent back toward node E ;

(5) Node D keeps forwarding the acknowledgment, which gets to node E .

Note that the field **PathSize** have changed from step (2) to step (4). This is because Bloom filter can be folded without causing noticeable false rates when the Bloom filter has few elements [6]. Besides, field **bDest** could contain the whole multicast tree, but the partial tree also works in this case. Furthermore, note that the multicast routing does not have to be a “tree” — it can be also be a directed acyclic graph (DAG) for multi-path routing purposes. In AMUR, a multicast routing based on DAG is also allowed.

C. Multicast LEAVE Protocol

In wireless ad hoc networks, multicast groups are in a flux state with node joining and leaving the multicast tree frequently. Because the Bloom filter that maintains the multicast tree cannot selectively prune the tree due to the implicit node/link anonymity, it is hard to keep concise and accurate multicast tree in AMUR. There are two ways to provide multicast LEAVE mechanisms — partial adjustment and complete re-establishment of the multicast trees.

1. Selective link removal at multicast tree branches can stop the multicast traffic from flowing down certain paths. This requires that we implement the asymmetric Bloom filter in the **bDest** field of multicast packets so that certain links can be removed, similar to the **TopoBF** maintenance in Section III-A. Under such selective link removal scheme, there are two mechanisms to maintain multicast group membership:

- **Implicit timeout mechanism:** In order to continuously receive multicast packets, multicast members have to periodically report their intentions by sending JOIN messages toward upstream branching nodes that are closer to the multicast source. Multicast branching nodes on the tree keep timing information of the JOIN messages. If the JOIN messages stop for an extended period of time, the one-hop downstream link is removed from the multicast tree in the **bDest** field of packets for the multicast group.

- **Explicit removal mechanism:** The multicast receivers on the multicast tree can explicitly send LEAVE messages to its upstream node to stop receiving multicast packets. The upstream node can either remove the link from the multicast tree, or stop forwarding multicast packets along that path.

2. Complete multicast tree reconstruction can rebuild the complete or portions of the multicast tree periodically, in which case the multicast tree is marked with sequence numbers. Interested multicast receivers must periodically subscribe to the multicast tree according to the current sequence number.

IV. ROBUSTNESS EVALUATIONS

A. Mobility Handling

The multicast source routing operation in AMUR can avoid routing disruption in certain mobility scenarios. If the network mobility only folds portions of the source paths or branches, and the remaining source route links still connects the source and destination nodes in the original link order, then AMUR source routing remains functional. In addition, AMUR supports multipath routing in its multicast source route Bloom filters, therefore can provide multiple paths in each AMUR data packet for reliability purposes in the presence of network mobility.

However, if mobility causes node order changes in the source route, or the source paths breaks, AMUR has to restart the

multicast tree construction operations using JOIN messages.

The other design paradigm is to store node pseudonyms instead of encrypted links into the source routes, which can avoid the source route breakage problems due to network link order and direction changes in the source route, therefore is more robust to network mobilities than link-based source routing. However, such a choice requires us to rethink about various aspects of AMUR, and is not covered in this paper.

B. Attacks Handling

AMUR implements a novel use of Bloom filters for efficient and anonymous multicast routing model in ad hoc networks. Therefore, there is no need for specially allocated multicast addresses, nor for soft-state maintenance for a moderate multicast tree when the `bDest` field can contain the whole multicast tree.

AMUR can provide strong anonymity to data forwarding and routing control mechanisms. First, link identities are doubly hidden in the `bDest` field of data packets by encrypting the link based on the Diffie-Hellman shared key, and by storing the encrypted link into the Bloom filter of data packets. Secondly, the source multicast tree and transmitter information of each data packet is completely hidden. Third, the routing information maintenance never reveals the identities of nodes two hops away from a node. Due to the use of Bloom filters, unless the attackers are omnipresent, an attack can only target at small parts of the ad hoc networks.

AMUR thwarts address spoofing attacks. The success rate of an address-spoofed packet getting through the network is exponentially reduced. In addition, AMUR provides the complete source multicast tree in each packet, and packet traceback is no longer necessary for detecting spoofing attackers. Although false positives in the Bloom filters of data packets may cause needless traffic in data forwarding, such cases are rare. In addition, even when false positive happens, the falsely forwarded data packets are highly unlikely forwarded farther.

However, AMUR cannot prevent nodes on the source path from injecting invalid packets and staging denial of service attacks in the ad hoc network.

V. CONCLUSION

We have presented AMUR, a new approach to anonymous multicast routing in ad hoc networks based on source multicast routing extensively using Bloom filters. We have also described the mechanisms to efficiently store multicast source routes anonymously, to manage topology information anonymously, and to forward data packets anonymously. Bloom filter is greatly adapted in AMUR for these purposes. The multicast source route anonymity is further enhanced using link encryption through Diffie-Hellman algorithm in AMUR.

REFERENCES

- [1] J. N. Al-Karaki and A. E. Kamal. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*, 11(6):6–28, Dec. 2004.
- [2] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees. *ACM SIGCOMM Computer Communication Review*, 23(4):85–95, August 1993.
- [3] C. L. Barrett, S. J. Eidenbenz, L. Kroc, M. Marathe, and J. P. Smith. Parametric probabilistic sensor network routing. In *Proc. of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA)*, pages 122–131. ACM Press, 2003.
- [4] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, Jul. 1970.
- [5] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. A novel solution for achieving anonymity in wireless ad hoc networks. In *Proc. of the 1st ACM PE-WASUN*, 2004.
- [6] A. Broder and M. Mitzenmacher. Network applications of Bloom filters: a survey. In *Proc. of the 40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.
- [7] C. Castelluccia and P. Mutaf. Hash-Based Dynamic Source Routing. In *IFIP Networking, LNCS 3042*, pages 1012–23, 2004.
- [8] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 4(2), Feb. 1982.
- [9] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptography*, 1(1):65–75, 1988.
- [10] Y. Cheng and D.P. Agrawal. Distributed Anonymous Secure Routing Protocol in Wireless Mobile Ad Hoc Networks. In *OPNETWORK 2005*, Aug. 2005.
- [11] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, 2000.
- [12] S.K. Das, B.S. Manoj, and C.S.R. Murthy. A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks. In *Proceedings of ACM/MOBIHOC*, 2002.
- [13] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transaction on Information Theory*, 22:644–654, Nov. 1976.
- [14] K. El-Khatib, L. Korba, R. Song, and G. Yee. Secure dynamic distributed routing algorithm for ad hoc wireless networks. In *International Conference on Parallel Processing Workshops (ICPPW)*, 2003.
- [15] J. J. Garcia-Luna-Aceves and E. L. Madruga. The Core-Assisted Mesh Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1380–94, 1999.
- [16] D. Goldschlag, M. Reed, and P. Syverson. Onion Routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):39C4, 1999.
- [17] D.B. Johnson and D.A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer Academic Publishers, 1996.
- [18] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In *Proc. of the thirty-third annual ACM symposium on Theory of computing (STOC)*, pages 163–172. ACM Press, 2001.
- [19] J. Kong and X. Hong. ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. In *MOBIHOC*, 2003.
- [20] J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proc. of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Nov. 2000.
- [21] A. Kumar, J. Xu, E.L. Li, and J. Wang. Space-code bloom filter for efficient traffic flow measurement. In *Proc. of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 167–172, Miami Beach, FL, 2003.
- [22] S.J. Lee, M. Gerla, and C.-C. Chiang. On Demand Multicast Routing Protocol. In *Proc. of IEEE WCNC*, pages 1298–1302, Sep. 1999.
- [23] J. Luo, P. Th. Eugster, and J.P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *Proc. of INFOCOM*, 2003.
- [24] R. C. Merkle. Secure Communication over an Insecure Channel. *Communication of ACM*, 21:294C99, Apr. 1978.
- [25] S. Murthy and J.J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal, Special issue on Routing in Mobile Communication Networks*, 1996.
- [26] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). Technical report, Network Working Group, Feb. 2004.
- [27] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, Nov. 1998.
- [28] S. C. Rhea and J. Kubiawicz. Probabilistic Location and Routing. *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, Jun. 2002.
- [29] E. M. Royer and C. E. Perkins. Multicast Operation of the Ad hoc On-Demand Distance Vector Routing Protocol. In *Proc. of MOBICOM*, pages 207–218, Seattle, WA, Aug. 1999.
- [30] S. D. Servetto and G. Barrenechea. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Proc. of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 12–21. ACM Press, 2002.
- [31] R. Song, L. Korba, and G. Yee. AnonDSR: Efficient Anonymous Dynamic Source Routing for Mobile Ad-Hoc Networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005.
- [32] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, Jul. 15 1998.
- [33] D. Sy, R. Chen, and L. Bao. ODAR: On-Demand Anonymous Routing in Ad Hoc Networks. In *Proc. The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.
- [34] Y. Xi, L. Schwiebert, and W. Shi. Preserving Privacy in Monitoring-based Wireless Sensor Networks. In *Proc. of the 2nd International Workshop on Security in Systems and Networks (SSN)*, April 2006.
- [35] L. Xiao, Y. Liu, W. Gu, D. Xuan, and X. Liu. Mutual anonymous overlay multicast. *J. Parallel Distrib. Comput.*, 66(9):1205–16, 2006.
- [36] Y. Zhang, W. Liu, and W. Lou. Anonymous communications in mobile ad hoc networks. In *Proc. of the 24th International Conference of the IEEE Communications Society (INFOCOM)*, 2005.