# Secure Cooperative Cache Based Data Access in Ad Hoc Networks

Wensheng Zhang, Liangzhong Yin, and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {wezhang,yin,gcao}@cse.psu.edu

***Abstract*— Cooperative caching, which allows the sharing and coordination of cached data among multiple nodes, can be used to improve the performance of data access in ad hoc networks. When caching is used, data from the server is replicated on the caching nodes. Since a mobile node may return the cached data, or modify the route and forward a request to a caching node, it is very important that the mobile nodes do not maliciously modify data, drop or forward the request to the wrong destination. In this paper, we identify possible security attacks on cache consistency and propose a randomized grouping based schemes for intrusion detection, damage recovery and intruder identification. We also address other security issues in cooperative cache based data access.**

**Index Terms:** Cooperative cache, security, invalidation report, ad hoc networks, randomized group.

## I. Introduction

Mobile ad hoc networks have been the focus of recent research due to their potential applications in civilian and military environments such as battlefield, disaster recovery, group conference, and wireless office. In ad hoc networks, mobile nodes communicate with each other using multi-hop wireless links. Due to lack of infrastructure support, each node acts as a router, forwarding data packets for other nodes. Most of the previous research in ad hoc networks focuses on the development of dynamic routing protocols [8], [15], [16], [18] that can efficiently find routes between two communicating nodes. Although routing is an important issue in ad hoc networks, other issue such as information (data) access is also very important since the ultimate goal of using ad hoc networks is to provide information access to mobile nodes.

Caching frequently accessed data items at the client side is an effective technique to improve performance in mobile environments. With caching, the data access delay is reduced since some data access requests can be served from the local cache, thereby obviating the need for data transmission over the scarce wireless links. However, caching techniques used in single-hop mobile environment (i.e., cellular networks) may not be applicable to multi-hop mobile environments since the data or request may need to go through multiple hops. As mobile nodes in ad hoc networks may have similar tasks and share common interests, cooperative caching [25], which allows the sharing and coordination of cached data among multiple nodes, can be used to reduce the bandwidth and power consumption. For example, in a battlefield, an ad hoc network may consist of several commanders and groups of soldiers. The commander has the data center, and the solders need to access the data center to get information about the enemy, the battlefield, and the attack plans. After a soldier obtained the attack information from the data center, it is very likely that nearby soldiers also need the same information from the data center. Bandwidth and power can be saved if these data accesses are served by the soldier with the cached data instead of the data center, which may be far away. Certainly, this cache sharing requires mobile nodes to coordinate with each other regarding their cached data.

When caching is used, data from the server is replicated on the caching nodes. Since a mobile node may return the cached data, or modify the route and forward a request to a caching node, it is very important that mobile nodes do not maliciously modify data, drop or forward the request to the wrong destination. The proposed research will study methods to avoid or detect such malicious nodes via authentication mechanisms. One common approach for data authentication is based on digital signature. With this approach, the data source can sign the data with its private key, so that intermediate routers cannot modify the data. However, the digital signature approach has high overhead, both in terms of time to sign and verify, and in terms of bandwidth. In this research, we design and evaluate techniques to reduce such overhead and balance system performance and security strength. Further, we identify possible security attacks on cache consistency and propose viable mechanisms to defend against such attacks.

The paper is organized as follows. In the next section, we present our cooperative cache based data access schemes. In Section 3, we propose techniques to deal with cache consistency attacks. In Section 4, we address other security issues in cooperative cache based data access. Section 5 presents related work, and Section 6 concludes the paper.

## II. Cooperative Cache Based Data Access

The idea of cooperative caching [6] can be explained by Figure 1, which shows part of an ad hoc network. Some nodes in the ad hoc network may have wireless interfaces to connect to the wireless infrastructure such as wireless LAN or cellular

This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

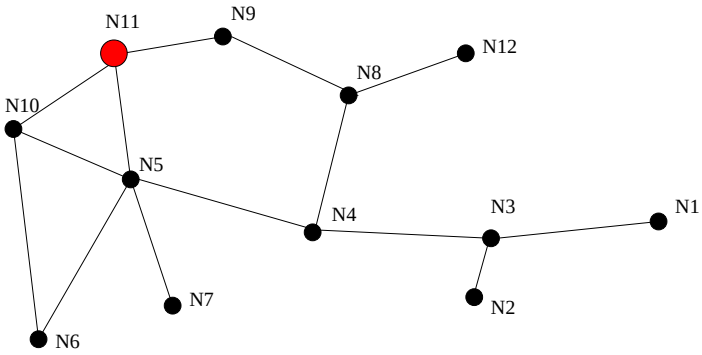


Fig. 1. An ad hoc network

networks. Suppose node $N_{11}$ is a data source (center), which contains a database of $n$ items $d_1, d_2, ..., d_n$. Note that $N_{11}$ may be a connecting node to the wired network which has the database. In ad hoc networks, a data request is forwarded hop-by-hop until it reaches the data center and then the data center sends the requested data back. Various routing algorithms have been designed to route messages in ad hoc networks. To reduce the bandwidth consumption and the query delay, the number of hops between the data center and the requester should be as small as possible. Although routing protocols can be used to achieve this goal, there is a limitation on how much they can achieve. In the following, we propose three cooperative caching schemes: *CachePath*, *CacheData* and *HybridCache*.

### A. Cache the Data Path (CachePath)

The idea of CachePath can be explained using Figure 1. Suppose node $N_1$ has requested a data item $d_i$ from $N_{11}$. When $N_3$ forwards the data $d_i$ back to $N_1$, $N_3$ knows that $N_1$ has a copy of $d_i$. Later, if $N_2$ requests $d_i$, $N_3$ knows that the data source $N_{11}$ is three hops away whereas $N_1$ is only one hop away. Thus, $N_3$ forwards the request to $N_1$ instead of $N_4$. Note that many routing algorithms provide the hop count information between the source and destination. By caching the data path for each data item, bandwidth and power can be reduced since the data can be obtained through less number of hops. However, recording the map between data items and caching nodes increases routing overhead. In the following, we propose various optimization techniques to improve the performance of CachePath.

In CachePath, a node does not need to record the path information of all passing data. For example, when $d_i$ is forwarded from $N_{11}$ to the destination node $N_1$ along the path $N_5 - N_4 - N_3$, $N_4$ and $N_5$ will not cache the path information of $d_i$ since $N_4$ and $N_5$ are closer to the data source than the caching node $N_1$. Thus, a router node only records the data path when it is closer to the caching node than the data source. Due to mobility, the node which caches the data may move. The cached data may be replaced due to cache size limitation. As a result, the node which modified the route should reroute the request to the original data source after it finds out the problem. Thus, the cached path may not be reliable and using it may adversely increase the overhead. To deal with this issue, a node caches the data path only when the caching node is very close. Techniques to define the closeness can be found in [25].

### B. Cache the Data (CacheData)

In the CacheData approach, the router node caches the data instead of the path when it finds that the data is frequently accessed. For example, in Figure 1, if both $N_6$ and $N_7$ request $d_i$ through $N_5$, $N_5$ may think that $d_i$ is a popular data and cache it locally. Future requests by $N_4$ can be served by $N_5$ directly. Since the CacheData approach needs extra space to save the data, it should be used prudently. Suppose the data source receives several requests for $d_i$ forwarded by $N_3$. The nodes along the path $N_3 - N_4 - N_5$ may found that $d_i$ is a popular item and should be cached. However, it will waste a large amount of cache space if three of them all cache $d_i$. To avoid this to happen, another rule is enforced: *a node does not cache the data if all requests for the data are from the same node*. In the previous example, all requests received by $N_5$ are from $N_4$, which in turn are from $N_3$. With the new rule, $N_4$ and $N_5$ will not cache $d_i$. If the requests received by $N_3$ are from different nodes such as $N_1$ and $N_2$, $N_3$ will cache the data. If the requests all come from $N_1$, $N_3$ will not cache the data, but $N_1$ will cache it. Certainly, if $N_5$ receives requests for $d_i$ from $N_6$ and $N_7$ later, it may also cache $d_i$.

### C. The HybridCache Approach

CachePath and CacheData can significantly improve the system performance. Our analysis [25] showed that CachePath performs better in some situations such as small cache size or low data update rate, while CacheData performs better in other situations. To further improve the performance, we propose a hybrid scheme called *HybridCache* to take advantage of CacheData and CachePath while avoiding their weakness. Specifically, when a mobile node forwards a data item, it caches the data or path based on some criteria. These criteria include the data item size and the number of hops that can be saved due to using CachePath. Detailed information about setting these parameters can be found in [25]. Next, we present solutions to defend against attacks on cache consistency, and then discuss some other security issues.

## III. Defending against Attacks on Cache Consistency

In this section, we identify cache consistency attacks and propose solutions to deal with such attacks.

### A. Cache Consistency

When cache is used, cache consistency issues [4] must be addressed to ensure that clients see only valid states of the data or at least do not unknowingly access data that is stale according to the rules of the consistency model. Problems related to cache consistency have been studied in many other systems such as multi-processor architectures, distributed file systems, distributed shared memory, and client-server database systems.

Two widely used cache consistency models are the weak consistency model and the strong consistency model. In the weak consistency model, a stale data might be returned to the client. In the strong consistency model, after a write completes, no stale copy of the modified data will be returned to the client. The exact definition of the completion of a write varies by the consistency approaches. The commonly used weak consistency mechanism is TTL-based (Time-To-Live), in which a client considers a cached copy up-to-date if its TTL has not expired. For strong cache consistency, *invalidation-based* and *polling-based* approaches are used. In the invalidation-based approach, the server keeps track of the clients that cache the data item, and sends invalidation messages to the clients when the data is changed. In the polling based approach, every time the user requests a data item and there is a cached copy, the cache first contacts the server to validate the cached copy, then returns the copy to the user. Since Polling-based approach may generate significant network traffic [5], the TTL-based approach is widely used for the weak cache consistency model and the invalidation-based approach is used for the strong cache consistency model.

### B. Defend Against Attacks on Cache Consistency

There are two kinds of attacks on the invalidation-based approach. First, the malicious node may stop propagating the invalidation message, and mobile nodes far away from the data source may not be able to receive the invalidation message and may use the stale cache without realizing it. Second, a malicious node may send an invalidation to some nodes and invalidate their caches which are still valid. If the data source signs the invalidation messages with a digital signature, the receiver can authenticate the message and avoid the second kind of attack. Note that a digital signature cannot be used to defend against the first kind of attack, since the mobile node may never receive the signed invalidation message. As for TTL-based approach, since the TTL and the cached data can be protected by digital signature, it does not suffer from this kind of attack. However, TTL-based approach can only provide weak consistency, and it can only be used for data which are rarely updated. In some strategic scenarios such as in the battlefield, accessing stale data (e.g., outdated enemy information) may be life threatening, and hence the strong consistency model should be adopted. In the following, we focus on dealing with attacks to the invalidation-based approach.

To prevent malicious nodes from dropping the invalidation messages, we borrow ideas from the IR-based cache invalidation [1], [4]. In this approach, the server periodically broadcasts an *invalidation report (IR)* in which the changed data items are indicated. The IR consists of the current timestamp $T_i$ and a list of tuples $(d_x, t_x)$ such that $t_x > (T_i - w * L)$, where $d_x$ is the data item $id$, $t_x$ is the most recent update timestamp of $d_x$, and $w$ is the invalidation broadcast window size. In other words, IR contains the update history of the past $w$ broadcast intervals. Based on the value of $w$, clients can still validate their local cache even after missing $w - 1$ IRs. Similar to the invalidation-based approach, clients use the IR to invalidate their local cache. Different from invalidation-based approach, the IR is sent out regularly, and the clients expect the IR at regular time interval. Therefore, if a client maliciously drops an IR, the nodes that are expecting the IR can detect it, and some measures can be taken to address the intrusion. Although flooding can be used to distribute IR to the mobile nodes, due to the high overhead, we assume that a multicast tree [11], [14], with the server as the root, is used to distribute the IRs to the mobile nodes.

### C. Reducing the Authentication Overhead

To authenticate the data source, digital signatures can be used. However, this approach has high overhead both in terms of computation and bandwidth. As shown in [12], asymmetric cryptographic operations (such as digital signatures) are three or four orders of magnitude slower than symmetric cryptographic operations (such as hash functions), and a 1024-bit RSA digital signature is roughly equivalent to the use of a 72-bit key in a symmetric encryption algorithm. To address this issues, researchers apply symmetric cryptographic techniques such as TESLA [21] to secure broadcast multicast, and routing in ad hoc networks. TESLA provides source authentication with message authentication codes (MACs) using only symmetric cryptography, based on delayed disclosure of keys by the sender. Suppose the receiver can loosely synchronize with the sender. Each packet is attached with a MAC computed using a key known only to the sender at that time. A short while later, the sender discloses the key and the receiver is able to authenticate the packet. If the packet arrives at the receiver after the key has been disclosed, it is discarded.

When applying TESLA to a large ad hoc network, different nodes may experience different amount of delay. For example, nodes closer to the data source receive the IR earlier than nodes far away from the data source. As a result, how long should the data source wait before disclosing the key? Waiting until the furthest node receiving the IR may take too long. The authors of TESLA also proposed enhanced approaches to allow immediate authentication at the client side, but this approach requires the server to delay sending the IR at the sender, which also has large overhead. Further, TESLA has synchronization requirements for all nodes in the network. Next, we present a novel solution to reduce the authentication overhead by randomized grouping.

*1) The Basic Idea:* To prevent intruders from modifying the invalidation messages, we enhance the IR-based approach with techniques for damage recovery, intrusion detection and isolation, based on the idea of randomized grouping as follows. The nodes (IR receivers) are randomly distributed into multiple groups, and the data center (server) shares a unique group key with the receivers of each group. The IR is protected by several MACs each with one group key. Since each receiver only knows one of the keys, if an intruder modifies the IR and the MAC using the group key it knows, the modification can be detected by a descendant in a different receiver group.

For example, as shown in Figure 2 (a), in the IR multicast tree, $N_0$ is the server and two group keys $K_0$ and $K_1$ are used. $N_0$ appends two MACs to the IR $< IR, MAC_0, MAC_1 >$,

and sends it to $N_1$, which forwards to $N_2$ and $N_3$. Suppose $N_1$ and $N_3$ know $K_1$ while $N_2$ knows $K_0$. Then, malicious modifications can be easily identified. For example, if $N_2$ is a malicious node and it modifies the IR, $N_3$ will be able to detect this modification by verifying $MAC_1$. After $N_3$ reports this modification to the server, the server can easily find that $N_2$ is the malicious node.

In the ideal case where neighbor nodes use different keys, the malicious node can be easily isolated. However, if two neighbor nodes use the same group key, further isolation may be difficult. For example, if both $N_1$ and $N_2$ both use $K_0$, it will be difficult for the server to find out who is the malicious node. When this happens, a tree reconfiguration may be necessary.

*2) Intruder Identification, Isolation, and Recovery:* When a node detects an intrusion, the detector should report the received IR (attached with the MACs) and the *id* of its parent, say $N_i$, directly to the server. It is possible that this message has to go through the malicious node, which may modify the report, and hence it should be signed by its group key. Certainly, the malicious node may still drop the report. In this case, the detector has to find other routes. The detector should still try until it receives the correct IR from the server. If going through the malicious node is the only way to reach the server, it will be similar to a network partition due to the malicious node, and the node must be aware that the cached data may be stale.

Receiving such a report, the server should ask $N_i$ to report the received IR and its parent $id$. This recursive process continues until the server has enough information to find out the compromised group keys, identify and isolate the intruder. A group key $K_k$ is compromised if there exists a compromised IR in which $MAC_k$ matches the compromised IR, where $MAC_k$ is the $MAC$ of IR using $K_k$. Based on the knowledge of the compromised group keys, the server checks the reports collected through a serious of nodes along a path, to identify the intruder or suspected intruders among these nodes. Suppose $K_j$ is compromised and $N_i$ knows $K_j$. If its direct child uses different key and detects the modification, $N_i$ is the malicious node unless its direct parent has the same group key. In which case, both will be suspects.

After finding the intruder, the server broadcasts this information to all nodes, which may reconfigure their multicast tree to remove this intruder from the tree. If several nodes are possible suspects and there is no way to isolate them, the server should remember their $id$s and inform them to reconfigure their multicast tree or assign new group keys to some of them so that two neighbor nodes can use different keys. Then, the server reconstructs and retransmits the IR so that no node will use the modified IR. Certainly, this new IR should not be sent to nodes that already received it correctly; i.e., the nodes that are parents of the malicious node.

By increasing the number of groups and keys, it is most likely that neighbor nodes will use different keys and hence intrusion detection and isolation can be easily accomplished. However, increasing the number of groups also increases the IR size and consumes more bandwidth. Some preliminary results are shown in Figure 2 (b). In the experiment, we assume the multicast tree is a binary tree with a height of 10, and the number of nodes is 1024. The figure shows the impacts of group number on the probability of being a cache consistency attack victim (cc-victim). As can be seen, when the number of groups increases, the probability of being a victim drops. Even when only two groups are used, the probability of being a victim is only about 3% when 6 attackers are randomly placed in the network.

## IV. Other Security Issues

When caching is used, data from the server is replicated in the caching nodes. One of the major concerns is the duplication of sensitive data, which the owner might want to restrict access. One solution is to define different levels of security for the data, with regard to duplication and storage in node caches. The data server can specify the level of security for each data item. Based on the security level, some data may not be cached, or only cached by a limited number of nodes. In the case of most sensitive data, the data server only sends the encrypted version to a number of nodes. Those trusted nodes will be able to get a shared key from the data server to decrypt the data. Future research should focus on designing mechanisms that would provide the owner a handle to control the scope of caching, and yet would not undermine the flexibility of caching. Note that there is a balance between security strength and system performance. By encrypting or limiting the distribution of the most sensitive data, some benefits of the cooperative caching will be lost.

With cooperative caching, the mobile nodes may return the cached data or modify the route and forward the request to the caching node, and hence, it is very important that the mobile nodes do not maliciously modify the data. One commonly used solution is data authentication which allows a receiver to ensure that the received data is authentic (i.e., it originates from the source and was not modified on the way), even when none of the other receivers of the data is trusted. To authenticate the data source, appending each packet with a message authentication code that is calculated using a shared key does not work since any receiver that has the shared key can forge the data and impersonate the sender. Consequently, it is natural to look for solutions based on asymmetric cryptography, namely digital signature schemes. After the data source signs the data with its private key, mobile nodes can verify the integrity of the data by using the public key of the data source. However, the digital signature approach has high overhead, both in terms of time to sign and verify, and in terms of bandwidth. Thus, we focus on reducing such authentication overhead. For example, if the data has gone through nodes with good reputation, the receiver may not need to verify the signature. This will tradeoff some security strength for system performance. Periodically, the receiver may want to verify the signature, and change the credit rating of other nodes based on the verification results. A mobile node has the option of verifying the signature if the data is very important or if it has enough computation power. This provides the user an option to choose the proper tradeoffs between security and performance; i.e., a mobile node can

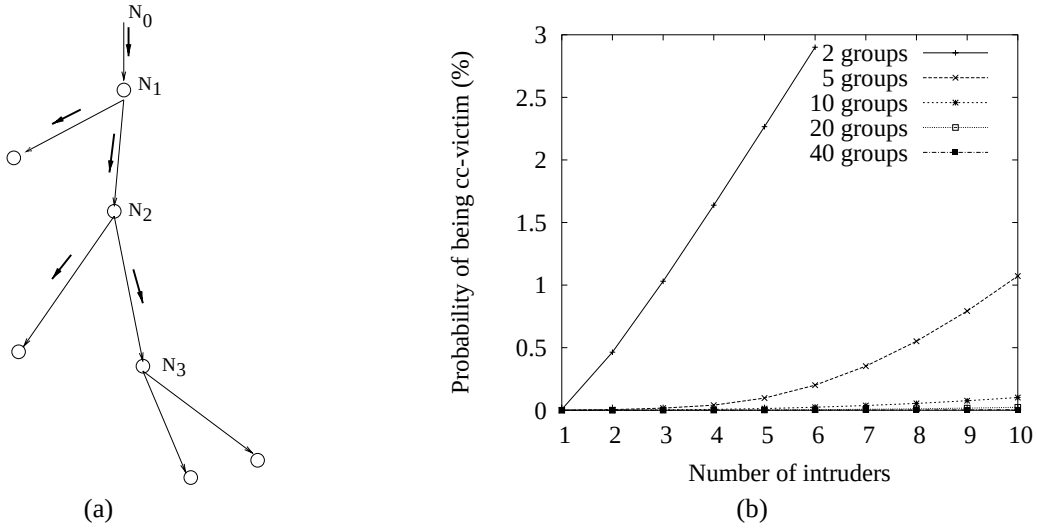


Fig. 2. Reducing the authentication overhead with randomized grouping

tradeoff security strength for performance if the data is not very important and it has less computation power or tradeoff performance for security strength otherwise.

## V. Related work

Cooperative caching [25] has been studied in the Web environment, but little work has been done to apply this technique to ad hoc networks. Due to mobility and resource constraints, techniques designed for wired network may not be applicable to ad hoc networks. For example, most researches on cooperative caching in the Web environment assume a fixed topology, but this may not be the case in ad hoc networks due to mobility. Since the cost of the wireless link is different from the wired link, the decision regarding where to cache the data and how to get the cached data may be different.

Recently, many researchers start to look into security issues in ad hoc networks. Hubaux *et al.* [13] addressed the issues of distributing public keys in ad hoc networks, by proposing to let users issue certificates for each other based on their personal acquaintances. Zhou and Haas [27] proposed a solution based on threshold cryptography. Based on a trusted certificate authority, the authors of [22] proposed a solution to secure the routing protocol of ad hoc wireless networks. In their protocol, nodes get certificates from the CA to identify themselves to avoid spoofing and malicious route updates. To address the high overhead associated with obtaining and verifying the digital certificates, Hu *et al.* proposed a protocol [12] to secure on-demand routing protocols based on TESLA [21], an efficient broadcast authentication scheme that requires loose time synchronization. They also identified the wormhole attack [12], which may make most routing protocols unable to find routes longer than one or two hops. Based on the intuition that a receiver can determine if the packet has traversed a distance that is unrealistic with precise timestamp or location information, they provided a packet leash solution to solve the wormhole attack.

Even with secure routing protocols, the source/destination may still set-up a route, which goes through a misbehaving (malicious) node that agrees to forward packets but fails to do so. Note that such misbehaving nodes can behave normally during the route discovery phase but maliciously drop packets when the time comes to route data. In [20], the authors proposed to use a *watchdog* entity to identify misbehaving nodes and a *pathrater* mechanism to avoid routing packets through misbehaving nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. This can be accomplished by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, it is misbehaving. The pathrater uses this knowledge of misbehaving nodes to choose the network path that is most likely to deliver packets. However, misbehaving nodes are not punished, and thus there is no motivation for the nodes to cooperate. To overcome this problem, Buchegger and Le Boudec [2] defined protocols that are based on a reputation system. In this approach, the nodes observe the behavior of each other and store this knowledge locally. Additionally, they distribute this information in reputation reports. According to their observations, the nodes are able to behave selectively (e.g., nodes may deny forwarding packets for misbehaving nodes). The solution in [23] exploits collaboration among local nodes to protect the network without completely trusting any individual node. To motivate packet forwarding among mobile nodes, Buttyan and Hubaux [3] gave a solution based on a virtual currency, called nuglets: If a node wants to send its own packets, it has to pay for it, whereas if the node forwards a packet for the benefit of another node, it is rewarded.

Other researchers in this field also address the problem of intrusion detection [26] and MAC layer misbehavior [17]. There are also researches on securing sensor networks [24], [28], [10], [7], [19], [9]. To our knowledge, there is no existing work addressing the problem of securing cooperative cache in ad hoc networks, which is the major goal of this paper.

## VI. Conclusions

As mobile nodes in ad hoc networks may perform similar tasks using common data sets, cooperative caching, which allows sharing and coordination of cached data among multiple nodes, can be used to reduce the bandwidth and power consumption. In this paper, we presented a cooperative cache based data access framework, where nodes may cache the data or the path to the data. To secure cooperative cache, we proposed solutions to reduce the data authentication overhead and identified possible attacks on cache consistency. We proposed a solution based on the IR-based cache invalidation strategy, to prevent intruders from dropping or modifying the invalidation message. Although digital signatures can be used to protect the IR, it has significantly high overhead in terms of computation and bandwidth consumption. To address this problem, we proposed randomized grouping-based techniques for intrusion detection, damage recovery and intruder isolation.

Although ad hoc networks have attracted many researchers, most previous research in this area focuses on routing or how to secure routing, and not much work on data access. Our work on cooperative cache based data access and its security issues will stimulate further research along these directions.

## References


[1] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," *ACM SIGMOD*, pp. 1–12, 1994.

[2] S. Buchegger and J-Y. Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes Fairness In Dynamic Adhoc NeTworks," *ACM Mobihoc*, pp. 80–91, June 2002.

[3] L. Buttyan and J. Hubaux, "Stimulating Cooperation in self-Organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, October 2003.

[4] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, September/October 2003 (A preliminary version appeared in ACM MobiCom'00).

[5] P. Cao and C. Liu, "Maintaining Strong Cache Consistency in the World-Wide Web," *IEEE Transactions on Computers*, pp. 445–457, April 1998.

[6] G. Cao, L. Yin, and C. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks," *IEEE Computer*, pp. 32–39, Feb. 2004.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *IEEE Symposium on Research in Security and Privacy*, 2003.

[8] S. Das, C. Perkins, and E. Royer, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *IEEE Infocom*, pp. 3–12, 2000.

[9] W. Du and J. Deng, "A Pairwise Key Pre-distribution Schemes for Wireless Sensor Networks," *The 10th ACM Conference on Computer and Communications Security*, 2003.

[10] L. Eschenauer and V. Gligor, "A Key-management Scheme for Distributed Sensor Networks," *The 9th ACM Conference on Computer and Communications Security*, pp. 41–47, November 2002.

[11] J. Garcia-Luna-Aceves and E. Madruga, "A Multicast Routing Protocol for Ad-Hoc Networks," *IEEE Infocom*, pp. 784–792, 1999.

[12] Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," *IEEE Infocom*, April 2003.

[13] H. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," *ACM MobiHoc*, pp. 146–155, 2001.

[14] J. Jetcheva and David Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," *ACM MobiHoc*, 2001.

[15] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing, Kluwer*, pp. 153–181, 1996.

[16] Y. Ko and N. Vaidya, "Location-aided Routing in Mobile Ad Hoc Networks," *ACM Mobicom*, pp. 66–75, 1998.

[17] P. Kyasanur and N. Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks," *Technical Report, UIUC*, Aug. 2002.

[18] S. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 7, no. 6, pp. 441–453, December 2002.

[19] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *The 10th ACM Conference on Computer and Communications Security*, 2003.

[20] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *ACM MobiCom*, Aug. 2000.

[21] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," *IEEE Symposium on Security and Privacy*, May 2000.

[22] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "A Secure Routing Protocol for Ad Hoc Networks," *IEEE Int'l Conf. on Network Protocols (ICNP)*, Nov. 2002.

[23] H. Yang, X. Meng, and S. Lu, "Self-Organized Network-Layer Security in Mobile Ad Hoc Networks," *ACM Workshop on Wireless Security*, Aug. 2002.

[24] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-route Filtering of Injected False Data in Sensor Networks," *IEEE Infocom'04*, March 2004.

[25] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Infocom*, 2004.

[26] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad-Hoc Networks," *ACM Mobicom*, pp. 275–283, Aug. 2000.

[27] L. Zhou and Z. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, November/December 1999.

[28] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," *IEEE Symposium on Security and Privacy*, 2004.