Computer Networks 87 (2015) 78–88

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A-CACHE: An anchor-based public key caching scheme in large wireless networks☆

Lin Yao[a], Jing Deng[b,*], Jie Wang[a], Guowei Wu[a]

[a] School of Software, Dalian University of Technology, Dalian 116023, China
[b] Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC 27412, U.S.A

ARTICLE INFO

Article history:
Received 4 March 2015
Revised 7 May 2015
Accepted 1 June 2015
Available online 10 June 2015

Keywords:
Asymmetric encryption
Public key
Caching
Large wireless networks

ABSTRACT

When asymmetric cryptography is used in wireless networks, public keys of the nodes need to be made available securely. In other networks, these public keys would have been certified by a certificate authority (CA). However, the existence of a single CA in large wireless networks such as mobile ad hoc networks and wireless sensor networks can lead to a communication hotspot problem and become an easy target for attacks. In this work, we propose a distributed technique, termed A-CACHE, to cache the public keys on regular nodes. One salient feature of our scheme is that some anchor nodes with larger cache memories are exploited. Due to the limited memory size that each node is allowed to dedicate for key caching, only a limited number of keys will be cached. Access to the public keys of other nodes is possible based on a chain of trust. In addition, multiple copies of public keys from different chains of trusted nodes provide fault-tolerant protections and guard against malicious attacks. We explain our technique in detail and investigate its prominent features in this work. Through analysis and evaluations, we observe the existence of an optimum ratio to cache the keys of local nodes.

1. Introduction

The importance of wireless networks with large number of nodes can never be overestimated. In mobile Ad hoc networks (MANETs), networked nodes form multi-hop topologies instantaneously in order to exchange information among themselves. In wireless sensor networks (WSNs), micro-sized sensor nodes are placed on different fields to detect anomaly and to deliver such results to data sinks. In internet of things (IOT), RFID devices help to link almost every piece of hardware together to form a large wireless network. While many research on large wireless networks have focused on network performance such as throughput, energy conservation, and network lifetime [2–5], many other recent research works investigated the security issues of such networks [5,6].

For instance, passive and active security attacks could be launched from outside by malicious nodes or from inside by compromised/misbehaving nodes. Without appropriate security protections, critical information of the networks can be leaked to adversaries.

Asymmetric cryptography techniques can be used to support information security. Using an asymmetric key scheme, each node in the network has a pair of keys: public key and private key. The public key should be known by all nodes in the network; the private key should only be held by the node itself. In order to achieve information confidentiality, a node uses the public key (of a receiver) to encrypt the message and sends it to the receiver through the wireless network. The encrypted message can only be decrypted by the intended receiver that holds the matching private key. In

☆ An early version of the proposed technique [1] was published and presented in IEEE GLOBECOM '10, CISS symposium, Miami, FL, U.S.A., December 2010.

* Corresponding author. Tel.: +1 3362561112.
E-mail addresses: yaolin@dlut.edu.cn (L. Yao), jing.deng@uncg.edu (J. Deng), andrew340621@163.com (J. Wang), wgwdut@dlut.edu.cn (G. Wu).

http://dx.doi.org/10.1016/j.comnet.2015.06.001
1389-1286/

order to achieve information/message authenticity, a node can use its own private key to sign the message. The receiver then authenticates the message by trying to decrypt the received message using the sender's public key.

However, such public keys may not be readily available or certifiable. Usually, there exists a certificate authority (CA) that will issue certificates for every node. Each certificate, signed by the CA, contains a public key and the identifier of a node. Unfortunately, such a CA is not suitable for large wireless networks. On the one hand, a CA can become an easy target for attackers. For example, the traffic toward the CA can be mis-routed with the use of worm-hole attacks or black-hole attacks. Jamming attacks can be launched by the adversary to blackout all wireless communications in the CA's neighborhood, effectively crippling the security of the entire network. On the other hand, large wireless networks are usually formed by nodes joining and leaving, without any pre-existing infrastructure; therefore, it is unlikely that such a CA exists in large wireless networks.

It is possible to store such public keys on the nodes themselves. A straightforward solution is to store the public keys of all nodes on every node so that every public key is easily accessible. However, there are two difficulties of doing so: the limited resource (such as memory space that can be dedicated to key storage) of such wireless network devices and the network dynamic (e.g., joining nodes, leaving nodes, as well as sleeping/awaking nodes). The limited memory space forbids nodes to store many public keys as they wish. The possibility of having newly joined nodes, nodes that have left, or nodes that might have been sleeping points to methods of storing public keys of only some nodes.

In this work, we propose an anchor-based caching scheme, A-CACHE. We deploy or recruit some nodes to serve as the anchor nodes, which dedicate more memory storage to key caching. These anchor nodes serve their neighborhood for queries for different nodes. We demonstrate that the inclusion of such anchors significantly improve the chance of success in finding the queried keys as well as reduce the query cost.

In addition, in order to increase the chance of query success, nodes should cache a good mixture of the keys of local nodes as well as the remote nodes, where the local nodes are defined as those in the neighborhood of the caching node and the remote nodes are outside of the neighborhood. The cache for local nodes ensures that nodes can be securely connected or trusted based on the cached public keys; the cache for remote nodes ensures that it is possible to find public keys for remote nodes from the queried neighborhood.

Our technique has the following salient features:

- Anchor nodes are used in the network to support the efficient key storage/query process. These anchor nodes are expected to dedicate more memory space for key caching.
- Every node stores two categories of public keys, the keys of local nodes (those in the node's neighborhood) and those of remote nodes. For example, node A can cache one key from m different nodes. Some of these m nodes are local nodes, while the other nodes are referred to as remote nodes since they are located outside of node A's neighborhood. A local caching ratio is assigned to balance the number of these two categories of public keys. We observe that there exists an optimal local ratio to maximize the overall public key availability in different network settings.
- We also design a key update strategy that allows nodes to update their public key caches according to the optimum local ratio. The update process balances the cache for local/remote nodes dynamically and will ensure the high availability of node public keys.

The paper is organized as follows: Section 2 describes recent related works. In Section 3, our public key caching scheme is explained in detail. We analyze the chance of key query success and optimize the local ratio in Section 4. Simulations are performed to evaluate our scheme in Section 5. In Section 6, we summarize our work and discuss future works.

2. Related work

Solutions to the problem of public key management in wireless networks have already been proposed. Key management schemes can be classified into two categories: certificate authority (CA) schemes and distributed schemes. In CA schemes, a trusted central authority has to stay on-line to cope with the changes of public key such as revoking or generating public keys periodically. Because the CA is responsible for the security of the entire network, it is a vulnerable point of the network. To solve this problem, the responsibility of a CA can be distributed [7]. We mainly discuss distributed public key management schemes below.

In distributed schemes, no CA is assumed or established. Instead, trust is usually propagated through a trust graph. Often times, it can be established with the help of the so-called web-of-trust [8]. The trusted nodes share the responsibility of collectively providing the CA functionality for the network to manage and distribute certificate keys. In the credential distribution scheme for key updates [9], users can issue public key certificates and authentication can be performed via the path generated by most trustworthy nodes. In [10], a trust-based threshold cryptography scheme for MANETs is proposed, allowing private keys to be generated by key shares obtained from a set of trustworthy 1-hop neighbors. In the novel certificate-less on-demand public key management (CLPKM) protocol [11,12], end-to-end trust value is used to select the most trusted route to verify public keys. There are other works exploiting the web-of-trust techniques, for example, cluster-based [13,14], binary-tree based [15], composite keys [16], and stable keys [17].

In distributed schemes, it is non-trivial to discover or query the certificate chain. In [18], Mohri et al. proposed a new distributed algorithm to find the certificate-chain and get the public keys. In the technique based on trust graph and threshold cryptography [19], users can issue public key certificates, authenticate each other via certificate chains. In [20], every node generates its own public-private key pair, issues certificates to neighboring nodes, and provides on-demand authentication services by means of gathering certificate chains towards a target node. This model is able to authenticate public keys by selecting the most trustworthy path in certificate chains.

Threshold cryptography scheme is also used in public key management. In [21], mobile certificate authorities

(MOCAs) are established for heterogeneous MANETs, where some nodes are more reliable and resourceful than others. These MOCAs share the responsibility of collectively providing the CA functionality for the network, using threshold cryptography. In the MOCA, a client requiring certificate service broadcasts a certificate request message and waits for responses from at least k out of the n MOCAs. With such k responses, the certificate can be fully reconstructed and the certification process succeeds. Other works [19,22,23] used similar approaches.

Our work differs from these related work in the sense that we focus on the use of limited memory space to cache the public keys of different nodes. We further design a public key search technique and a cache update algorithm to achieve optimum caching ratio of public keys from local/remote nodes. Distributed caching, in which data, instead of keys, are stored on the wireless nodes distributively [24]. Cooperative caching has been investigated with the help of the underlying routing protocols [25–27]. Several works [28,29] developed lightweight cooperative cache management algorithms aimed at maximizing the traffic volume served from cache and minimizing the communication cost in wireless P2P networks. In [30], Nikos et al. proposed a collaborative cache management strategy to achieve a communication optimization among the sensors so as to serve data in short latency and with minimal energy consumption. Compared to these works which focused on the caching of a commonly shared data, our paper differs from the perspective that we investigate a network where each node has its data (public key in our discussions) that needs to be accessed later on by other nodes and hence needs to be cached.

Our approach is also related to in-network caching [31–33], which often focus on contents cached on Internet routers. For instance, Psaras et al. [31] investigated methods to reduce caching redundancy in order to achieve better utilization of resource along the delivery path. A dynamic group caching scheme was proposed in [34], allowing group masters to manage all local cache contents and ensure caching and energy consumption efficiency. In order to improve the efficient of content delivery, Lee et al. [35] used Bloom filter to exchange the information of the cached contents among caching nodes, lowering delivery latency and traffic volume and balancing load among different links. Hash-routing schemes [33] are also designed to exploit in-network caches without requiring network routers to maintain per-content state information, achieving the trade-off between increased internal load and reduced latency. Several recent research works on information-centric networks and their sustainability [36] and social community based cooperative caching in mobile social wireless networks [37]. These work focus on contents but not public keys. Compared to these works, we focus on the problem of caching public keys of wireless nodes among the MANET nodes and using the web-of-trust to retrieve the keys.

3. Public key caching scheme

3.1. System and threat models

We assume that, in a WSN with N number of nodes with limited on-board storage. Each node carries its own public key and has some cache space for cache $m < N$ public keys of other nodes. There exists no public key infrastructure or authentication center. When there is a need to establish secure communications with the nodes whose public keys are not cached, they will be requested through other trusted nodes through the chain of trust.

Our scheme works on the assumption that nodes on the chain of trust will not modify contents on the keying messages. We assume that the following attacks from other nodes are possible in the network:

1. Attackers may receive, read, record, and retransmit the key request/response packets as they are passing through.
2. Attackers may send fabricated keying messages.
3. Attackers may drop keying messages passing through.

We first introduce our notations and variables

- PU_i: public key of node i;
- PR_i: private key of node i;
- $\{pt\}_{PR_i}$: message pt signed by PR_i;
- cm: crypted message;
- $\mathcal{C}_i$: set of nodes whose public keys are cached in node i;
- $\mathcal{N}_i$: set of physical neighbors of node i;
- $\mathcal{R}$: list of nodes that involve in the sequence of KREQ message transmission;
- m: total number of public keys that each regular node caches;
- m': total number of public keys that each anchor node caches;
- ϵ: extensive ($\epsilon = 1$) or simple ($\epsilon = 0$) search;
- γ: ratio of the numbers of keys for local/remote nodes cached by one node;
- TTL: the maximum number of hops a KREQ message may travel.
- MAC: message authentication code.

3.2. Operational details of the proposed scheme

An illustration of public key caching is shown in Fig. 1, in which node S needs to find the public key of node Q.

Assume that every node carries the public keys of some other nodes. Such information can be obtained through pre-deployment key caching or through cache update, which will be discussed later. Newly joined nodes can either obtain them based on pre-deployment phases or through other channels of initial bootstrap operations [38].

We first define two control messages that will be sent among nodes for public keys. These messages are called Key REQuest (KREQ) message and Key REPly (KREP) message. The KREQ message will be transmitted from the node requesting the public key of another node. The format of KREQ message is shown in Fig. 2. In this figure, Source and Destination represent the node requesting the public key and the node's identifier whose public key is being requested; the list of routers ($\mathcal{R}$) represents the nodes who have been passing along the request. MAC is signed by the private key of the message sender. In addition, KREQ messages are only processed by those nodes caching the public key of the message sender; other nodes will just ignore them.

The format of KREP message is similar (see Fig. 2). The KREP message contains the public key of the destination and

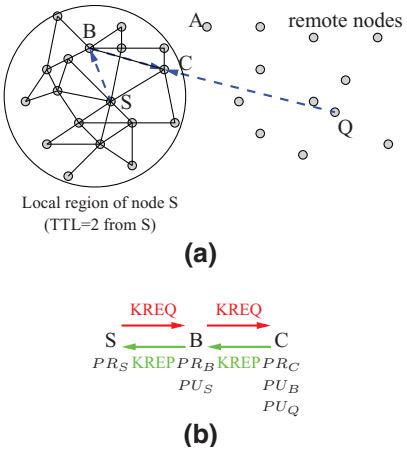

Fig. 1. Illustration of public key caching. In (a), Node S caches the public keys of some local nodes (inside the solid circle) and some remote nodes (outside of the solid circle). When node S needs to find a chain of trust toward another node, node Q, it sends out a query message. Only trusted nodes will forward the message, until finding the queried key or the query TTL expires). In (b), we show the messages as well as the public/private key storage.

KREQ Message Format

Source (S)	Destination (Q)	TTL	List of Routers (R)	$\{MAC\}_{PR_f}$

KREP Message Format

Source (S)	Destination (Q)	PU_Q	List of Routers (R)	$\{MAC\}_{PU_r}$

Fig. 2. Message format of the KREQ and the KREP packets. The MAC on each of the messages are encrypted by either the private key of the forwarder (KREQ message) or the public key of the immediate receiver (KREP message). This is to avoid message fabrication and alteration by nodes outside of the link of trust.

the entire message is encrypted by the immediate receiver's public key, which has been cached by the sender of the KREP message.

Note that both of these two keying messages carry MACs with them, hence eliminating the chance of message tempering. There is a subtle difference: MAC is encrypted by the private key of the forwarder in the KREQ message since all potential receivers (forwarders) of the KREQ message are expected to carry the public key of the current sender; MAC in the KREP message is encrypted by the public key of the next-step receiver since it is cached by the current sender. The beauty of the setup is that such a protection comes from the inherent key structure that is required for the keying process, i.e., some nodes carrying the public keys of other nodes and thus signed information can be trusted.

We explain the operational details of the KREQ and the KREP messages with an example. Suppose node S needs to obtain node Q's public key. It sends a KREQ=$\{S, Q, TTL, \mathcal{R} = \Phi\}$ to itself. Each of those nodes receiving the KREQ message with the public key of the last forwarder should check whether it has cached Q's public key. If the queried public key is not cached on this node, it will attach its own ID to the end of the router list, $\mathcal{R}$, generate a new MAC based on its own private key and the new KREQ message, and rebroadcast it to its 1-hop neighborhood. In addition, any node receiving a KREQ message should check whether it caches the public key of the last broadcasting node. The KREQ message will be processed only when this is the case. Otherwise, the KREQ message will be silently dropped. This is basically to establish a link of trusted nodes.

If the queried public key is found, the node will prepare a KREP message and return it to the querying node through $\mathcal{R}$ that has been stored in the received KREQ message. Note that we do not encrypt the queried public key because the key query process is based on a fundamental trust on the chain. Therefore, in Fig. 1b, node B as well as node C is free to tamper with the queried public key PU_Q, and we leave the detection of such attacks to other procedures. For instance, the authenticity of the received public key can be verified with a simple message exchange with node Q.

The processing of KREP message is simple: a node receiving a KREP message should first decrypt the message using its private key. The decrypted public key copy of the queried node is either stored or forwarded toward the original querying node, node S, with the help from $\mathcal{R}$.

There is a parameter ϵ, which controls whether a node caching Q's public key should still forward the KREQ message [1]. When $\epsilon = 1$, our scheme operates with extensive search. All nodes will forward the KREQ message. When $\epsilon = 0$, our scheme operates with simple search: only those nodes without Q's public key in their cache will forward the KREQ message.

The details of the above operations are presented in Algorithm 1. Nodes receiving KREP messages should proceed according to Algorithm 2.

Algorithm 1 Algorithm to process KREQ message.

```
Node i receives KREQ={S, Q, TTL, R, MAC}
f ← last node ID in R
if f ∉ C_i OR MAC does not match then
    drop message and exit
end if
R ← {R; i}
if Q ∈ C_i then
    Prepare KREP as {S, Q, PU_Q, R}_{PU_f}
    Node i sends KREP to node f
    if ε = 0 then
        exit
    end if
end if
TTL ← TTL − 1
if TTL > 0 then
    Prepare KREQ as {S, Q, TTL, R, MAC_{PR_i}}
    Broadcast KREQ
end if
```

3.3. Update policy

An indispensable component in our key caching scheme is the cache update policy. While the network is in operation, a node gets to know the public keys of other nodes. Such new public keys can be obtained through interactions with other nodes. Once a node sees the public key of a new node, the following question arises: should it replace a currently cached

Algorithm 2 Algorithm to process KREP message.

1: Node i receives a KREP=$\{S, Q, PU_Q, \mathcal{R}\}_{PU_i}$
2: Decrypt message using PR_i
3: **if** $i = S$ **then**
4: Obtain PU_Q and exit
5: **else**
6: $f \leftarrow$ previous node ID in $\mathcal{R}$ ahead of i
7: Prepare KREP as $\{S, Q, PU_Q, \mathcal{R}\}_{PU_f}$
8: Node i sends KREP to node f
9: **end if**

key with the new key? If so, which of the currently cached keys should be removed?

We categorize the public keys that are cached on each node into two groups: local and remote, which represent the public keys of the local nodes that are direct neighbors of the node itself and remote nodes, respectively. "Local" nodes are defined as those nodes within *TTL*-hop and the value of *TTL* is a system parameter. All those nodes that are more than *TTL*-hop away are considered as "remote" nodes.

In our scheme, there is a pre-defined local public key ratio γ. If the current local public key ratio in cache is lower than γ and the public key of a local node is received, the public key will be used to replace one of the remote public keys. Similarly, if the current local public key ratio in cache is higher than γ and the public key of a remote node is received, the public key will be used to replace one of the local public keys. [1] Different methods can be used to choose the key to be replaced, such as last-in-first-out, first-in-first-out, and most rarely used. In this work, we choose a random selection method among the keys in the same local or remote category. Compared to conventional content caching techniques, which usually consider content popularity, our scheme focuses on the node popularity, i.e., how often the node is being requested for a public key. Therefore, a more optimal cache update policy may take into consideration the node popularity.

This cache update algorithm is described in detail in Algorithm 3.

Algorithm 3 Algorithm to update cache.

Require: received new key PU_j at node i
Ensure: Up-to-date $\mathcal{C}_i$
1: **if** $j \notin \mathcal{C}_i$ **then**
2: $\gamma_{cur} \leftarrow |\mathcal{C}_i \cup \mathcal{N}_i|/m$
3: **if** $\gamma_{cur} < \gamma$ and $j \in \mathcal{N}_i$ **then**
4: Update cache by replacing a randomly selected key that belongs to remote nodes with PU_j
5: **end if**
6: **if** $\gamma_{cur} > \gamma$ and $j \notin \mathcal{N}_i$ **then**
7: Update cache by replacing a randomly selected key that belongs to local nodes with PU_j
8: **end if**
9: **end if**

[1] Care needs to be taken to avoid an oscillation effect, in which the key cache is frequently updated with the local ratio fluctuating slightly above and below γ. We leave this to our future work.

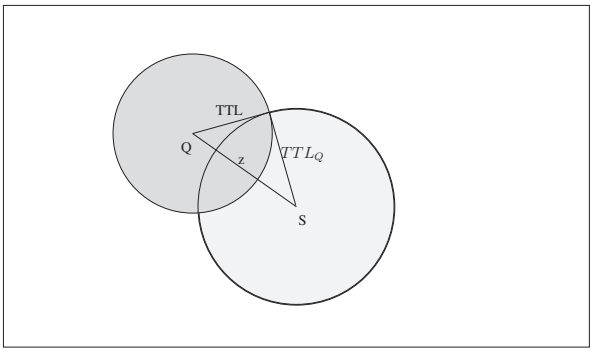

Fig. 3. Node S queries for the public key of node Q. Query messages are sent within TTL_Q hops; while nodes within TTL are considered local.

3.4. Defense against attacks

Due to page limit, we can only briefly discuss our scheme's defense against different attacks. Since all messages are protected with encrypted MAC, attackers cannot change the contents of legitimate keying messages. Fabricated keying messages will be ignored after a mismatch is found between the message and the encrypted MAC. Dropping keying messages can only reduce the chance of finding a closer public key, but with the availability of multiple copies, one copy will eventually be found so long as the number of attackers in the system is not overwhelming.

4. Performance analysis

We investigate the performance of our design in terms of the chance of finding at least one copy of the public key of a destination node, Q, and use the probability to optimize γ.

We make the following assumptions: we assume that the keys are cached on the nodes independently and each regular, anchor, node carries γm, $\gamma' m$, keys of local nodes and $(1-\gamma)m$, $(1-\gamma')m$, keys of remote nodes. [2] with different Local nodes are defined as those within TTL-hop from the requesting node. First we compute the node density of the network, ρ, by ignoring border effects:

$$\rho = \frac{N}{XY}, \tag{1}$$

where XY represent the size of the region.

Therefore, on an average, each node has n_T local nodes (that are within TTL-hop) if we approximate the size of the region as a circle with a radius of TTL $\cdot R$:

$$n_T \approx \rho\pi\,(\text{TTL}\ \cdot R)^2 - 1, \tag{2}$$

where we have assumed $n_T < N$.

Suppose node S queries the public key of node Q (see Fig. 3), if node Q is the local node of one of the flooded nodes for node S, the probability that it is not cached is

$$P_L = 1 - \frac{\min(\gamma m, n_T)}{n_T}, \tag{3}$$

[2] Intuitively, this is can be explained by the fact that nodes with random set of prior-trusted nodes are joining the network in random order.

if the node is a regular node; The probability is

$$P'_L = 1 - \frac{\min(\gamma m', n_T)}{n_T}, \tag{4}$$

if the node is an anchor node.

Similarly, if node Q is the remote node of one of the flooded nodes for node S, the probability that it is not cached is

$$P_R = 1 - \frac{\min((1-\gamma)m, N - n_T - 1)}{N - n_T - 1}, \tag{5}$$

if the node is a regular node; The probability is

$$P'_R = 1 - \frac{\min((1-\gamma)m', N - n_T - 1)}{N - n_T - 1}, \tag{6}$$

if the node is an anchor node.

Node S floods its query message within a TTL_Q hop of its neighborhood (see Fig. 3). Assume that the distance between nodes S and Q is z. Assume there are $N_R(z)$ and $N_L(z)$ nodes that have node Q as remote nodes and local nodes (and $N_R(z) + N_L(z)$ being all the nodes within TTL_Q hop of the requesting node), respectively. Then the probability of node Q's public key not being cached by all of these nodes at all is

$$1 - P_s(z) = \text{Prob (none of } N_R(z) \text{ nodes caches it)} \cdot \text{Prob(none of } N_L(z) \text{ nodes caches it)} \tag{7}$$

Therefore, the chance of success in node S' query is

$$P_s(z) = 1 - \text{Prob(none of } N_R(z) \text{ nodes caches it)} \cdot \text{Prob(none of } N_L(z) \text{ nodes caches it)} \tag{8}$$

We still need to compute $N_L(z)$ and $N_R(z)$. First, we have

$$N_R(z) + N_L(z) = N_Q, \tag{9}$$

where N_Q being the number of nodes queried by node S. An approximation is

$$N_Q \approx \rho\pi\,(TTL_Q \cdot R)^2. \tag{10}$$

Next we observe that locality is mutual: if node A is a local node of node B, node B should be a local node of node A. This is due to the property of wireless bidirectional links. Therefore, we consider all the N nodes in the network and distinguish those local nodes that are within TTL_Q hop of node S. For example, for a node Z that is far away from the region, $N_L = 0$; for a node Z' that is close to the requesting node, $N_R = 0$.

Now we are ready to estimate $N_R(z)$ and $N_L(z)$. As the choice of node Q changes from right next to node S to far away from it, we observe that $N_L(z)$ decreases from N_Q to 0 and $N_R(z)$ increases from 0 to N_Q.

As can be observed from Fig. 3, those nodes that are local to node Q and will be queried by node S should reside in the intersect of two circles: one centered at node Q with a radius of TTL; the other centered at node S with a radius of TTL_Q.

Assume that the distance between S and Q is z. The unit is in hops as we approximate distance in hops. Simple geometry gives us the size of the intersect region

$$S(z, r_1, r_2) = r_1^2 acos\left(\frac{z^2 + r_1^2 - r_2^2}{2zr_1}\right) + r_2^2 acos\left(\frac{z^2 + r_2^2 - r_1^2}{2zr_2}\right) - \frac{\sqrt{(-z+r_1+r_2)(z+r_1-r_2)(z-r_1+r_2)(z+r_1+r_2)}}{2}$$

when $z < r_1 + r_2$; otherwise $S(z, r_1, r_2) = 0$. Since $r_1 = TTL$ and $r_2 = TTL_Q$, we simplify the denotation of $S(z, r_1, r_2)$ as $S(z)$. $N_L(z)$ and $N_R(z)$ can be expressed as

$$N_L(z) = \rho \cdot S(z) \tag{11}$$

and

$$N_R(z) = N_Q - \rho \cdot S(z) \tag{12}$$

If we ignore border effects and assume a perfect circle outside of source node S with radius of T hops, the chance of node Q at distance z toward node S can be expressed based on the following accumulative probability function.

$$Pr(x \le z) = \frac{\pi z^2}{\pi T^2} \tag{13}$$

when $0 \le z \le T$. Thus, the probability density function of z is

$$f(z) = \frac{2z}{T^2}, \tag{14}$$

when $0 \le z \le T$. The network is actually rectangle, so an estimate for T is necessary:

$$\pi T^2 \approx X \cdot Y \tag{15}$$

Therefore,

$$T \approx \sqrt{\frac{XY}{\pi}} \tag{16}$$

We can then approximate the probability of success as

$$\overline{P_s} = \int_0^T P_s(z) f(z) dz \tag{17}$$

In order to compute (17), we need to express the two unknown terms in (8) for $Ps(z)$. We start with

$$\text{Prob(none of } N_R(z) \text{ nodes caches it).} \tag{18}$$

The complication is that some of these $N_R(z)$ nodes can be anchor nodes and the probabilities of these anchors caching node Q's public keys are different to those of regular nodes. Assume there are L anchor nodes among N nodes. On an average, the number of anchor nodes among $N_R(z)$ nodes is

$$\ell_R(z) = \frac{L}{N} \cdot N_R(z), \tag{19}$$

which may not be an integer.

We treat the $N_R(z)$ nodes as an average set of nodes that contain $\ell_R(z)$ anchors and $N_R(z) - \ell_R(z)$ regular nodes.[3] Therefore,

$$\begin{aligned} &\text{Prob(none of } N_R(z) \text{ nodes caches it)} \\ &= (P_R)^{N_R(z) - \ell_R(z)} (P'_R)^{\ell_R(z)} \\ &= (P_R)^{\frac{N-L}{N} N_R(z)} (P'_R)^{\frac{L}{N} N_R(z)} \end{aligned} \tag{20}$$

[3] A more accurate and complex analysis is to derive the probabilities of having different anchor nodes among these $N_R(z)$ nodes.

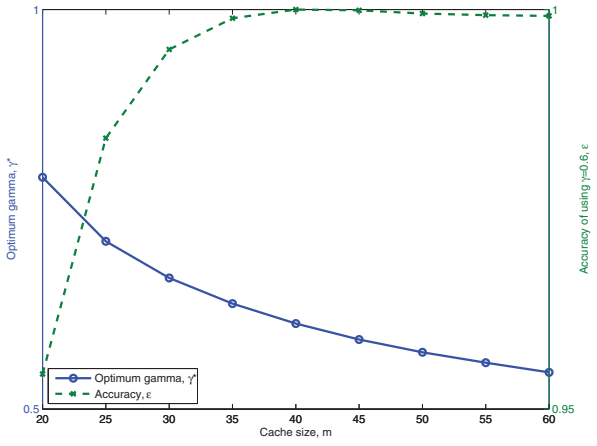

Fig. 4. Numerical results on γ^*, the optimum γ.

Similarly, we have

$$\text{Prob(none of } N_L(z) \text{ nodes caches it)} = (P_L)^{\frac{N-L}{N}N_L(z)}(P'_L)^{\frac{L}{N}N_L(z)} \quad (21)$$

Thus, the probability of success can be expressed as

$$\overline{P_s} = \int_0^T \left[1 - (P_R)^{\frac{N-L}{N}N_R(z)}(P'_R)^{\frac{L}{N}N_R(z)} \cdot (P_L)^{\frac{N-L}{N}N_L(z)}(P'_L)^{\frac{L}{N}N_L(z)}\right] \cdot \frac{2z}{T^2}dz, \quad (22)$$

where T, $N_R(z)$, $N_L(z)$, P_R, P'_R, P_L, P'_L are given by (16), (12), (11), (5), (6), (3), and (4), respectively.

With $\overline{P_s}$, we can optimize it based on γ using numerical methods. The results of γ^* are presented in Fig. 4. The network area is 2000 × 2000 m^2 and the transmission range is $R = 250m$ m. There are $N = 400$ nodes, including $L = 20$ anchors. In Fig. 4, the value of γ^* is shown for different m. The accuracy of using $\gamma = 0.6$ is also presented as between 95% and 100% throughout the range of m that we investigated. ϵ is defined as the success probability of $\gamma = 0.6$ (the value of γ that will be used in Section 5) divided by the maximum success probability achieved by γ^*.

5. Performance evaluation

Our performance evaluation includes two components: one implemented in MATLAB and the other implemented in NS2. We discuss the results in two subsections.

We evaluated the following metrics:

- Success rate. This is the probability of the querying node successfully obtaining the public key of the queried node within the pre-defined TTL_Q hop.
- Number of key copies: This is the number of key copies that has been found for the queried node. It represents the redundancy of the key cache in the TTL_Q-hop neighborhood.
- Overhead: This is the number of control packets for each key that has been found, on an average. It represents the overhead caused by the caching scheme.

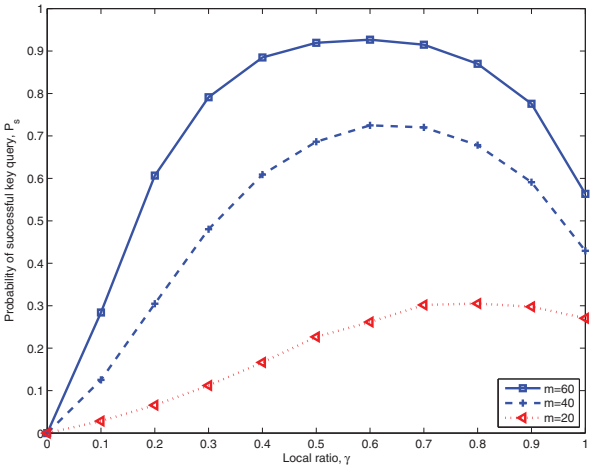

Fig. 5. Probability of successful key query within TTL_Q-hop of the source node. The cache size of the anchor nodes is assumed to be $m' = 2m$.

Unless specified otherwise, MATLAB simulations were set up as follows: $N = 400$ nodes were randomly distributed in a network of size 2000 m by 2000 m. The radio transmission range was assumed to be 250 m. Initially each regular node carried the public keys of a randomly chosen set of nodes (size $m = 40$) in the network. $L = 20$ anchor nodes (counted in the N nodes) were also randomly distributed in the network. The cache size of each anchor node is $m' = 2$m. Local region hop count, TTL, within which hop-count a node would be considered a "local" node, was assumed to be 2. The search region hop count, TTL_Q, was 1. The local ratio is $\gamma = 0.6$.

Then we randomly selected some source/destination pairs throughout the network. In each of the source/ destination pairs, the source node needed to obtain the public key of the destination node. Therefore, a key query process was initiated.

5.1. MATLAB simulations

We first present our simulations in MATLAB to evaluate the A-CACHE scheme with different γ and TTL values. We argue that MATLAB simulations are adequate because packet level simulations do not affect the results for key caching and key query, especially in static networks. Packet level simulations and dynamic network results are presented in Section 5.2.

In Fig. 5, we compare the probability of successful key query within TTL_Q-hop of the source node. The benefit of a well-chosen γ is clear. When γ is small, only a small portion of the local nodes carries the public keys of the forwarding node, limiting the scope of the key query. On the other hand, when γ is too close to 1, the public keys of remote nodes are not cached, reducing the chance of successful key query. The optimum γ can be seen as close to 0.6.

The effects of the anchor nodes are demonstrated in Fig. 6, in which we compare the success probability of our scheme with different L. The cache size of each anchor node is assumed to be maximally possible, i.e., $m' = N$. From Fig. 6,

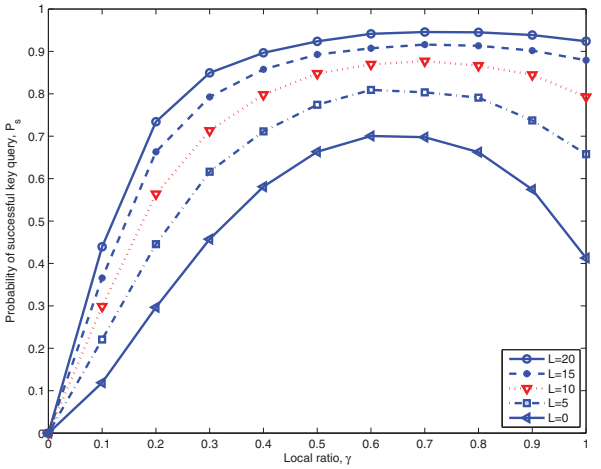

Fig. 6. Probability of successful key query within TTL_Q-hop of the source node. The cache size of the anchor nodes is assumed to be $m' = N$, i.e., maximum.

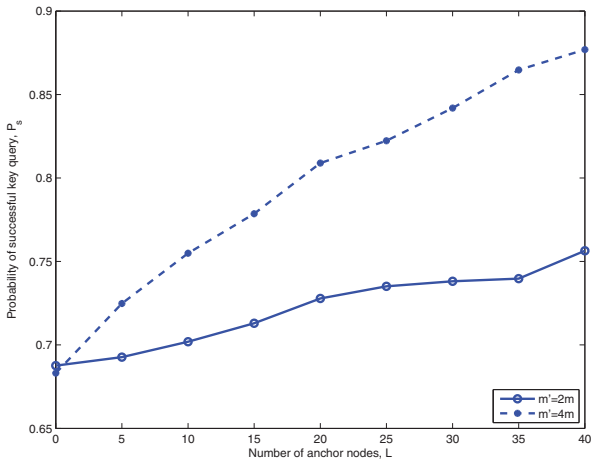

Fig. 7. Probability of successful key query within TTL_Q-hop of the source node (TTL = 1, m = 40, $\gamma = 0.6$), with different L. The cache size of the anchor nodes is either $m' = 2$m or $m' = 4$m.

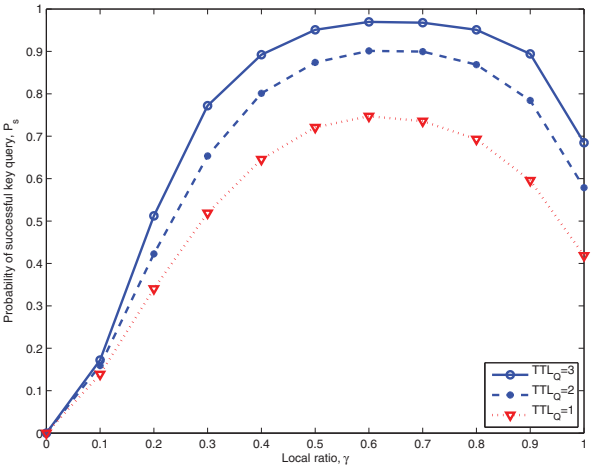

Fig. 8. Probability of successful key query with different TTLs.

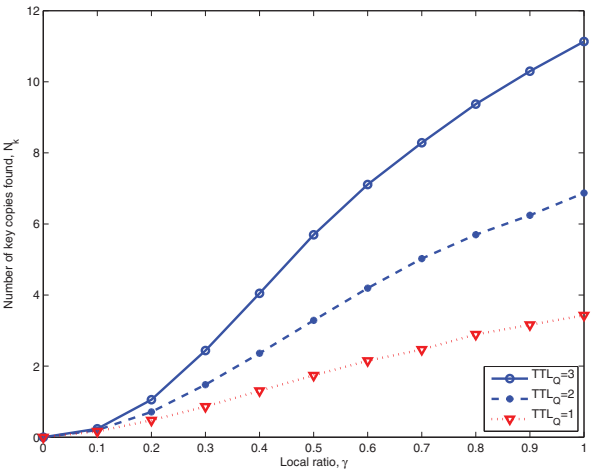

Fig. 9. Number of key copies found in each key query. Such numbers represent the redundancy of the queried public keys.

the impact of the anchor nodes is obvious as we introduce a few anchor nodes. As more and more anchor nodes are introduced, the chance of successful key query is above 90%; therefore, the gain of adding more anchor nodes becomes smaller.

In Fig. 7, we compare the success ratio of different L and different anchor cache sizes. As can be observed in Fig. 7, the success probability improves with L. With larger anchor caches, the improvement is higher. Another way to measure the return of adding such extra cache to the network is the percentage of success probability increase. As we add 10% more cache space (L=20 and $m' = 4$m, compared to $m' = 2$m), P_s increases from 0.72 to 0.82, which is a 14% gain, underscoring a good return of the extra cache space.

In Fig. 8, we show the query success probability for different TTL_Q values. As TTL_Q increases, the chance of success is better, coming with a higher overhead. The optimum $\gamma = 0.6$ is again demonstrated.

We present the average number of key copies, N_k, found in each key query in Fig. 9, from which we can observe that the number increases with γ as well as TTL_Q. While it is easy to explain the increase of the number of key copies with TTL_Q, the relation between N_k and γ is more complex: with a large γ, more neighboring nodes are participating in the search because only those nodes carrying the public key of the last sender of the key query message will rebroadcast it. Therefore, the impact of such participating neighbors overshadows other effects of increasing γ.

The overhead evaluation is shown in Fig. 10, where we compare the number of broadcast messages per found key, N_c. The value is computed as the total number of broadcast messages divided by the total keys found for each query. Naturally, the overhead for large TTL_Q is higher as more nodes are queried and recruited to broadcast the query message further. The rather flat bottom of the curves suggests that the overhead of using γ in the range of 0.6–0.9 is about the same.

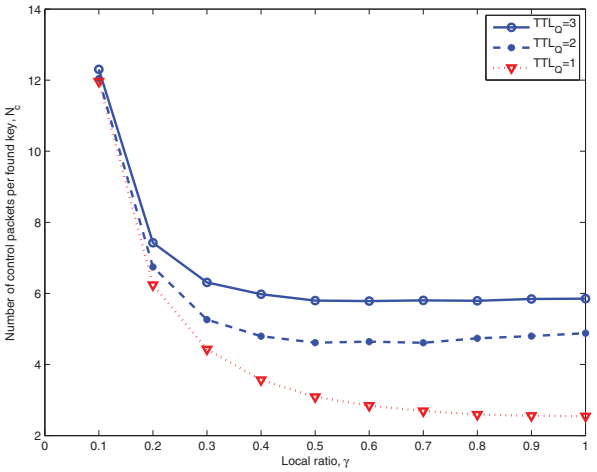

Fig. 10. Comparison of message broadcast overhead as the number of message transmissions for each found key, on an average.

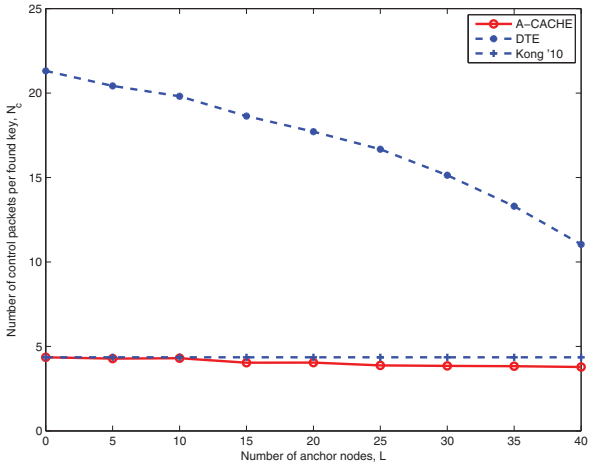

Fig. 11. The average number of control packets for each key found in the queries. The network is static.

Note that the energy expenditure is usually closely related to the above overhead, i.e., the number of control packets transmitted leading to one successfully found key. This is because, if we treat normal operation of the nodes as the floor of the energy consumption, the processing and transmission of a packet poses an extra component for energy consumption.

5.2. NS2 simulations

We implemented the A-CACHE scheme and two related work, the dynamic trust establishment (DTE) scheme [39], and the Kong'10 scheme [1] that does not use anchor nodes, and evaluated their performance. We simulated static as well mobile networks with different maximum speeds.

The overhead of A-CACHE, DTE, and Kong'10 schemes is compared in Fig. 11. The A-CACHE scheme requires much few control packets compared to the DTE scheme, although as L increases, relatively smaller overhead can observed (in both DTE and A-CACHE). Such smaller overhead are due to the increasing number of anchor nodes, i.e., better key availability.

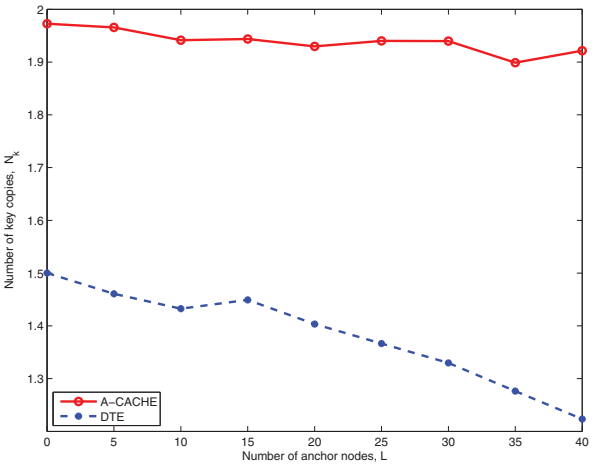

Fig. 12. The average copies of keys for each successful key query. The network is static.

Table 1
Redundancy of found keys in mobile networks.

Scheme	$V_{max} = 0$	$V_{max} = 10$ m/s	$V_{max} = 20$ m/s
A-CACHE	1.93	2.19	2.20
DTE	1.40	1.73	1.77

In Fig. 12, we compare the redundancy of queried keys of the DTE and the A-CACHE schemes. The comparison is made on the metric of the average copies of keys that are found on a successful key query, N_k. The A-CACHE scheme returns about 25% more keys than the DTE scheme does, offering better redundancy and potentially shorter turn-around time.

In Table 1, we further compare the key redundancy of the A-CACHE and DTE schemes in networks with different speeds. The mobility model is random way-point with zero pause time. Node speeds were chosen randomly between 0 and V_{max}. The number of key copies for each successful key query is shown in Table 1. It can be seen that both schemes have increased redundancy as nodes move and the A-CACHE scheme enjoys a high redundancy than the DTE scheme. Such an increase might have been caused by the network dynamics, with nodes caching more diverse public keys.

6. Conclusion

Asymmetric cryptography requires an authenticated/certified copy of the public key from the other party. These public keys are usually certified by a CA in many networks. In large wireless networks, such CAs may not exist. In this work, we have proposed to use the nodes to serve as the peer CAs. A node requesting the public key of another node may obtain multiple copies from different chains of trust. In order to achieve this goal, we have designed the A-CACHE scheme to allow nodes using their limited memory space to cache some of the public keys obtained securely. We have also designed a public key search technique through secure multi-hop paths.

In the investigation of our scheme, we have observed that there exists an optimum caching ratio for the public keys of

local nodes. This is due to the effect of secure broadcast as well as public key availability.

We have presented details of the proposed A-CACHE scheme with theoretical analysis as well as extensive numerical and simulation results for performance evaluation. We have found that the A-CACHE scheme enjoys high success rate and low control message overhead. Our investigation also suggests that a local ratio of 0.6 can be used to achieve close to optimum caching in the network scenarios that we studied.

We conclude our work to explain the difference of our work with distributed content caching, which has some striking similarities with our approach. The inherent difference between these two is actually the unique directional connection among nodes if we treat the relationship of carrying another node's public key as a uni-directional connectivity. None of the content caching strategies has such a connectivity.

In terms of future work, we note that the cache replacement policy can be an interesting topic for a future work. In fact, the random replacement may not be optimum. For instance, some caches are being used/accessed more often should be kept instead of being replaced. We leave it to our future work. Furthermore, key revocation is an important aspect of any key infrastructure and our design will need to include such a procedure as well.

References

[1] Y. Kong, J. Deng, S.R. Tate, A distributed public key caching scheme in large wireless networks, in: Proceedings of the IEEE Global Telecommunications Conference - Communication and Information System Security (GLOBECOM '10), Miami, FL, USA, 2010, pp. 1–5.

[2] S. Weber, J.G. Andrews, N. Jindal, An overview of the transmission capacity of wireless networks, Commun. IEEE Trans. 58 (12) (2010) 3593–3604.

[3] J.G. Andrews, R.K. Ganti, M. Haenggi, N. Jindal, S. Weber, A primer on spatial modeling and analysis in wireless networks, Commun. Mag. IEEE 48 (11) (2010) 156–163.

[4] M. Di Francesco, S.K. Das, G. Anastasi, Data collection in wireless sensor networks with mobile elements: a survey, ACM Trans. Sens. Netw. 8 (1) (2011) 7:1–7:31.

[5] X. Zhou, R.K. Ganti, J.G. Andrews, A. Hjorungnes, On the throughput cost of physical layer security in decentralized wireless networks, Wireless Commun. IEEE Trans. 10 (8) (2011) 2764–2775.

[6] S. Vasudevan, D. Goeckel, D.F. Towsley, Security-capacity trade-off in large wireless networks using keyless secrecy, in: Proceedings of the eleventh ACM International Symposium on Mobile Ad hoc Networking and Computing, 2010, pp. 21–30.

[7] J.H. Koo, B.H. Kim, D.H. Lee, Authenticated public key distribution scheme without trusted third party, in: Proceedings of the Embedded and Ubiquitous Computing–EUC 2005 Workshops, Springer, 2005, pp. 926–935.

[8] K. Hamouid, K. Adi, Efficient certificateless web-of-trust model for public-key authentication in manet, Comput. Commun. 63 (2015) 24–39.

[9] A. Boukerche, Y. Ren, A trust-based security system for ubiquitous and pervasive computing environments, Comput. Commun. 31 (18) (2008) 4343–4351.

[10] H. Dahshan, J. Irvine, A trust based threshold cryptography key management for mobile ad hoc networks, in: Proceedings of the IEEE 70th Vehicular Technology Conference (VTC 09-Fall), 2009, pp. 1–5.

[11] S. Maity, R.C. Hansdah, Certificate-less on-demand public key management (clpkm) for self-organized manets, in: Information Systems Security, 2012, pp. 277–293.

[12] S. Maity, R.C. Hansdah, Self-organized public key management in manets with enhanced security and without certificate-chains, Comput. Netw. 65 (2014) 183–211.

[13] A. Rachedi, A. Benslimane, Trust and mobility-based clustering algorithm for secure mobile ad hoc networks, in: Proceedings of the International Conference on Systems and Networks Communications (ICSNC'06), 2006.72

[14] R. Azarderakhsh, A. Reyhani-Masoleh, Z.-E. Abid, A key management scheme for cluster based wireless sensor networks, in: Proceedings of the International Conference on Embedded and Ubiquitous Computing, (EUC'08). IEEE/IFIP, vol. 2, 2008, pp. 222–227.

[15] G. Kambourakis, E. Konstantinou, S. Gritzalis, Binary tree based public-key management for mobile ad hoc networks, in: Proceedings of the IEEE International Symposium on Wireless Communication Systems (ISWCS'08), 2008, pp. 687–692.

[16] J.-H. Cho, K.S. Chan, I.-R. Chen, Composite trust-based public key management in mobile ad hoc networks, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013, pp. 1949–1956.

[17] M. Suguna, P. Subathra, Establishment of stable certificate chains for authentication in mobile ad hoc networks, in: Proceedings of the International Conference on Recent Trends in Information Technology (ICRTIT), 2011, pp. 234–239.

[18] H. Mohri, I. Yasuda, Y. Takata, H. Seki, Certificate chain discovery in web of trust for ad hoc networks, in: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 2, 2007, pp. 479–485.

[19] M. Omar, Y. Challal, A. Bouabdallah, Reliable and fully distributed trust model for mobile ad hoc networks, Comput. Security 28 (3) (2009) 199–214.

[20] P. Memarmoshrefi, R. Seibel, D. Hogrefe, Bio-inspired self-organized public key authentication mechanism for mobile ad-hoc networks, in: Bio-Inspired Models of Network, Information, and Computing Systems, Springer, 2012, pp. 375–386.

[21] R.K. Seung Yi, Moca: mobile certificate authority for wireless ad hoc networks, in: In Proceedings of the 2nd Annual PKI Research Workshop (PKI 03), 2003, pp. 65–79.

[22] J. Luo, J.-P. Hubaux, P.T. Eugster, Dictate: distributed certification authority with probabilistic freshness for ad hoc networks, Depend. Secure Comput. IEEE Trans. 2 (4) (2005) 311–323.

[23] Y. Zhang, W. Liu, W. Lou, Y. Fang, Securing mobile ad hoc networks with certificateless public keys, Depend. Secure Comput. IEEE Trans. 3 (4) (2006) 386–399.

[24] X. Fan, J. Cao, H. Mao, Y. Liu, Gossip-based cooperative caching for mobile applications in mobile wireless networks, J. Parallel Distrib. Comput. 73 (5) (2013) 653–663.

[25] L. Yin, G. Cao, Supporting cooperative caching in ad hoc networks, Mobile Comput. IEEE Trans. 5 (1) (2006) 77–89.

[26] J. Taylor, B. Tang, M.B. Yildirim, Steady status study of distributed data caching in ad hoc networks, in: Proceedings of the 22nd International Conference on Computer Communications and Networks (ICCCN), 2013, pp. 1–6.

[27] A. Sengupta, S. Amuru, R. Tandon, R.M. Buehrer, T.C. Clancy, Learning distributed caching strategies in small cell networks, in: Proceedings of the 11th International Symposium on Wireless Communications Systems (ISWCS), 2014, pp. 917–921.

[28] J. Zhao, P. Zhang, G. Cao, C.R. Das, Cooperative caching in wireless p2p networks: design, implementation, and evaluation, Parallel Distrib. Syst. IEEE Trans. 21 (2) (2010) 229–241.

[29] N. Golrezaei, A.G. Dimakis, A.F. Molisch, Wireless device-to-device communications with distributed caching, in: Proceedings of the IEEE International Symposium on Information Theory Proceedings (ISIT), 2012, pp. 2781–2785.

[30] N. Dimokas, D. Katsaros, L. Tassiulas, Y. Manolopoulos, High performance, low complexity cooperative caching for wireless sensor networks, Wireless Netw. 17 (3) (2011) 717–737.

[31] I. Psaras, W.K. Chai, G. Pavlou, Probabilistic in-network caching for information-centric networks, in: Proceedings of the second edition of the ICN workshop on Information-centric networking, 2012, pp. 55–60.

[32] K. Cho, M. Lee, K. Park, T.T. Kwon, Y. Choi, S. Pack, Wave: popularity-based and collaborative in-network caching for content-oriented networks, in: Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2012, pp. 316–321.

[33] L. Saino, I. Psaras, G. Pavlou, Hash-routing schemes for information centric networking, in: Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking, 2013, pp. 27–32.

[34] A. Rao, P. Kumar, N. Chauhan, Edgc: efficient dynamic group caching technique for mobile ad hoc networks, Int. J. Comput. Appl. 41 (13) (2012) 25–30.

[35] M. Lee, K. Cho, K. Park, T. Kwon, Y. Choi, Scan: Scalable content routing for content-aware networking, in: Proceedings of the IEEE International Conference on Communications (ICC), 2011, pp. 1–5.

[36] S. Gitzenis, G.S. Paschos, L. Tassiulas, Enhancing wireless networks with caching: asymptotic laws, sustainability & trade-offs, Comput. Netw. 64 (2014) 353–368.

[37] M. Taghizadeh, S. Biswas, Community based cooperative content caching in social wireless networks, in: Proceedings of the Fourteenth ACM International Symposium on Mobile Ad hoc Networking and Computing, 2013, pp. 257–262.

[38] D. Balfanz, D.K. Smetters, P. Stewart, H.C. Wong, Talking to strangers: authentication in ad-hoc wireless networks, in: NDSS, 2002.
[39] C. Papageorgiou, K. Birkos, T. Dagiuklas, S. Kotsopoulos, Dynamic trust establishment in emergency ad hoc networks, in: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, 2009, pp. 26–30.

Lin Yao received her Ph.D. degree from Dalian University of Technology, PR China in 2011. She is now an associate professor at the School of Software, Dalian University of Technology (DUT), PR China. Dr. Yao visited the Department of Electrical and Computer Engineering at Stony Brook University in 2014. Her research interests include wireless sensor networks and opportunistic networks.

Jing Deng is an associate professor in the Department of Computer Science at the University of North Carolina at Greensboro (UNCG), Greensboro, North Carolina, USA. Dr. Deng visited the Department of Electrical Engineering at Princeton University and the Department of Electrical and Computer Engineering, WINLAB at Rutgers University in Fall of 2005. He was with the Department of Computer Science at the University of New Orleans from 2004 to 2008. He served as a Research Assistant Professor in the Department of Electrical Engineering & Computer Science at Syracuse University from 2002 to 2004. He received his Ph.D. degree from School of Electrical and Computer Engineering at Cornell University, Ithaca, New York, USA in January, 2002. He received his M.E. and B.E. degrees in Electronic Engineering at Tsinghua University, Beijing, PR China, in 1994 and 1997, respectively.

Dr. Deng is an editor of IEEE Transactions on Vehicular Technology. He is the co-recipient of the 2013 Test of Time award by the ACM Special Interest Group on Security, Audit, and Control (SIGSAC). Dr. Deng's research interests include wireless network and security, information assurance, mobile ad hoc networks, and social networks. His research webpage is at http://www.uncg.edu/~j_deng/

Jie Wang received his B.E. degree from Dalian University of Technology, PR China. Currently, he is an M.E. candidate at the School of Software at Dalian University of Technology. His research interests include wireless sensor networks, opportunistic networks, and network coding.

Guowei Wu received his Ph.D. degree from Harbin Engineering University, PR China. He was a Research Fellow at INSA of Lyon, France, from September 2008 to March 2010. He is now a professor at the School of Software, Dalian University of Technology (DUT). His research interests include embedded real-time system, cyber-physical systems (CPS), and wireless sensor networks.