Trusted Computing for Protecting Ad-hoc Routing

Michael Jarrett and Paul Ward
Electrical and Computer Engineering, University of Waterloo
mjarrett@ieee.org, pasward@ccng.uwaterloo.ca

Abstract

Ad-hoc networks rely on participation and cooperation of nodes within the network to transmit data to destinations. However, in networks where participating nodes are controlled by different owners, nodes may choose to act in their own interest to the detriment of the network. Current solutions either exact high overheads on the network and nodes, or only operate in specialized scenarios and prevent a small selection of attacks.

Trusted computing provides additional security in open computing environments by allowing software to prove its identity and integrity to remote entities. We propose using trusted computing to prevent misconfigured or malicious nodes from participating in the network. We extend AODV to ensure that only trustworthy nodes participate in the network. The protocol exacts less overhead on the network than many other approaches and can be applied in a wide variety of scenarios.

Keywords: ad-hoc networks, routing security in ad-hoc networks, denial of service, trusted computing

1. Introduction

Ad-hoc networks are networks where participating nodes are responsible for routing traffic. Such networks are often built using mobile wireless devices, where the communication range of an individual node is insufficient to reach all other nodes in the network. Nodes therefore rely on other nodes to relay traffic to destinations beyond their transmission range. Central to the concept of mobile ad-hoc networks are the ideas that nodes are constrained in computational and battery power, and that the network lacks universally-available supporting infrastructure.

Some ad-hoc networks only include devices from a single administrative domain (*e.g.*, a military sensor network). However, often more interesting to civilians is the opposite scenario, where each device is individually administered by its owner - the extreme of multiple administrative domains. A typical example of such a scenario is a community wireless mesh network, where houses are each equipped with wireless nodes capable of forwarding traffic towards a wired Internet connection, and in turn may also make use of this connection [1].

A difficulty for ad-hoc networks in scenarios involving multiple administrative domains is that there is very little motivation for a node of one administrative domain to honestly participate in the routing of packets for another. A node can act selfishly, transmitting and receiving traffic over the network, but not establishing routes or forwarding traffic for other nodes. A node can also act maliciously, purposely drawing traffic to it for modification or inspection. A node could even corrupt routing information in other nodes, effectively disrupting communications over the wider network.

The area of *trusted computing* offers a solution. Trusted computing brings similar security guarantees as secure coprocessors [2] to standard computing architectures. One key ability of trusted computing is the ability to perform *remote attestation*, which allows a device to certify to a remote entity that it is running some known combination of hardware and software, and that such software and hardware has not been tampered with. An ad-hoc network could use this to identify which nodes will adhere to the rules of conduct within the network.

In this paper we describe a system using trusted computing that prevents selfish or malicious nodes from participating in the ad-hoc network. Section 2 itemizes some of the threats against routing in an ad-hoc network. Section 3 describes current attempts to solve the issues of selfish or malicious nodes in ad-hoc networks. Section 4 discusses the basic concepts of trusted computing. Section 5 presents our trusted-computing-based solution, and describes situations where it would be effective. Section 6 compares our approach to existing solutions, demonstrating the benefits of our technique. Finally, Section 7 presents conclusions and future work.

2. Threats

There are several threats that an unprotected ad-hoc network faces, falling into two broad categories.

Selfish nodes are nodes which are self-interested: they have no particular desire to hamper communications in the network, but are unwilling to commit their own limited resources to forwarding traffic not directly related to their own communication. This can be accomplished by not responding to route requests, responding with unfavourable route metrics, or even responding accurately to routing requests but then not forwarding traffic on that route.

A second threat of a selfish node is that it can ignore node-level bandwidth restrictions of the network. Experiments have shown that if some nodes use more than their fair share of bandwidth on the network, that other nodes will be starved for bandwidth, and network capacity as a whole will diminish [3].

Malicious nodes act to the detriment of the network by manipulating routing. Many routing protocols use hop count as a metric - a node can falsely claim a low hop count to a destination, enabling it to intercept traffic for that destination. Node identities are not authenticated, so a node can claim to be the destination of a route, or, by switching identities, set up routing loops (despite protections against such occurrences in most ad-hoc routing protocols). Finally, a node can erroneously generate error messages to break formed routes, either as a legitimate node on that route, or by impersonating a legitimate node. Several such attacks are discussed by Sanzgiri *et al.* [4].

3. Current Solutions

Most proposals address either the problem of selfish nodes (encouraging nodes to participate in routing) or secure routing (ensuring nodes route correctly), but rarely both. In particular, most protocols used to prevent selfish nodes assume a underlying secured routing protocol.

One such secure routing protocol is Authenticated Routing for Ad-hoc Networks [4] (ARAN). In ARAN, all routing messages are signed using public-key cryptography at each hop, and messages which are not validly signed are dropped. This, combined with requiring the use of round-trip time as a routing metric, prevents many forms of attack on the routing infrastructure.

A technique described in the proposal of secure AODV (SAODV [5]) separates nodes by their relative 'security level', and gives each security level a shared secret with which to encrypt routing headers. Routes can then be found which are guaranteed to only travel through nodes at this security level. Unfortunately, this requires coordination and trust among all nodes of a particular security level.

A monitoring system by Marti *et al.* [6] places a 'watchdog' in each node. This watchdog monitors the next hop of a source route to ensure that the node in fact forwards packets correctly. If the node misbehaves, the watchdog sends a warning back to the source, where the 'pathrater' module in each node incorporates the knowledge into routing decisions. This may help to avoid routes through untrustworthy nodes, but does not in any way discourage selfish behaviour. Note that this system cannot easily work with routing protocols like AODV, since the technique requires source routing.

Similar to the watchdog approch is the CONFIDANT [7] protocol, which presents a detailed architecture for detecting and reacting to errant nodes. More aggressive than the watchdog proposal, this system works to actively deny access to the network to errant nodes, giving selfish nodes motivation to co-operate.

The AD-MIX [8] protocol targets systems where a node may accept routes but refuse to forward traffic by encrypting each packet between zero and two times, each encryption using the public key of another node. The packet is sent to the node whose key was used for the final encryption, which will decrypt it and relay it to the next point on the route. A node which drops a packet it is requested to forward risks dropping a packet destined for itself. This gives incentive to forward packets as well as providing anonymity and privacy.

Various economic approaches have been proposed to counter selfish nodes. One, using a virtual currency called nuglets [9], has the sender pay for transmission, and each node that forwards the packet takes some portion of this payment. The node's nuglet balance must be maintained by a secure coprocessor. A similar approach [10] by Hubaux *et al.* uses base stations in an ad-hoc-extended cellular network to reward forwarders with real currency on their bill. It is also a very efficient approach in that it only uses symmetric cryptography.

4. Trusted Computing

Trusted computing [11] (TC) is a general term used to describe a set of features in hardware and software to enhance the security of open computing platforms. An open computing platform is one in which arbitrary hardware and software can be deployed, and is the more common configuration for personal computing devices. This is contrasted with the area of closed computing devices, such as game consoles, embedded systems, *etc.*, where only very specific hardware and software combinations are allowed. The latter provides additional security, since

a malicious user cannot run their own software or hardware on the devices. However, closed computing devices do not have the flexibility that computer users have come to appreciate and require.

TC allows for guarantees similar to that of a closed computing device to be made while still using an open computing device. TC systems are normally described to have four key features.

1. Process isolation
2. Secure I/O
3. Sealed storage
4. Remote attestation

Process isolation guarantees that a secure process, in some literature referred to as a *trusted agent*, is protected from inspection or modification by another process. Secure I/O allows this trusted agent to communicate with hardware without fear of modification or inspection of the communications channel. Sealed storage allows data to be stored encrypted to an insecure medium like a hard disk, while only allowing chosen trusted agents to decrypt it. The most interesting aspect for our work is that of remote attestation, which allows a trusted agent to demonstrate to a remote entity that it, and the platform on which it relies, has certain properties. Used in combination, these four features can allow a trusted agent on a device to operate without interference, and prove that it does so to remote entities.

Remote attestation combines an assurance that a trusted computing environment is running, and that the remote party is communicating with a trusted agent with particular properties. Most implementations require the environment loading the trusted agent to calculate a hash of the executable before loading it into memory [12]. However, more advanced implementations have proposed attestations with proof of high-level properties of the agent's behaviour rather than the exact identity of the software binary [13], greatly simplifying the issues of software upgrade and interaction of different software brands.

Central to a TC implementation is a *trusted module*. This is normally a hardware cryptographic processor, operating similarly to a smartcard. The processor is capable of performing standard public-key encryption/signing, symmetric-key encryption, and hashing. It is also capable of generating and storing private keys such that they cannot be retrieved by software means. It also allows measurements of system state to be stored securely inside it and used in reports made in remote attestation.

Trust is extended in layers, with a trusted BIOS performing initial measurements on the system, and taking measurements of the operating-system loader. The operating-system loader should, when loaded, have access to its secure storage and be able to perform remote attestation based on the measurements made by the trusted module. If it is designed for trusted computing, it can then perform measurements of the operating system kernel. This process can continue until there is a set of measurements that ensure the integrity of all software between a trusted agent and hardware [14].

These capabilities must be combined with platform enhancements to support secure I/O and stronger process isolation. It is generally assumed that most of the software on the system, including a large portion of the operating-system services, are not verified by the trusted computing environment, so the trusted agents must have their memory and communications protected from interference originating from other processes on the system.

Remote attestation requires some form of public-key infrastructure. Each trusted module has a keypair protected in internal memory. The public component of this keypair can be verified by a trusted third party (TTP). This TTP will issue certificates for keys generated by the trusted module after it verifies the public component of the internal key. These additional keypairs are also protected in the trusted module, but allow the user of a TC system to maintain anonymity by having multiple keypairs available.

The main operation performed in our protocol is that of *sealed-signing* [15], often referred to as *quoting*. This operation uses a key within the trusted module to sign data, but also includes in this signature a nonce and a chosen set of measurements from the platform and software.

The infrastructure required to support trusted computing applications is already widely available. TPM hardware is already installed in many laptop computers [16], and will likely become more common in more computers and devices in the future. Likewise, TC-enabled bootloaders, Linux kernel support, and support libraries and applications have already been developed by the open source community [17]. Upcoming Microsoft operating system Windows Vista is also expected to support trusted computing.

5. TC to Protect Ad-hoc Networks

Given the properties of trusted computing described above, we propose its use in ad-hoc networks to provide guarantees that nodes can be neither selfish nor malicious. Our design for this purpose consists of two key components. First, we describe how the various software components (specifically, the routing agent and the wireless device driver) must operate in the trusted-computing context. Second, we describe an extension to AODV to take advantage of the trusted-computing facilities. When we

detail that protocol extension, we will show how the inclusion of trusted-computing support within nodes of a mobile ad-hoc network is used to prevent several forms of attack.

First, we presume the software and operating-system components responsible for managing routing in the ad-hoc network run as a trusted agent. Given this, the routing agent should be largely standardized, to ensure that remote nodes can easily verify attestations from it. This routing agent software has three key responsibilities. First, it must obey all routing rules of the ad-hoc routing protocol chosen: it will not attempt to impersonate any other node, misrepresent any routing metric it reports, issue invalid error messages, transmit faster than agreed-upon rate limits, or ignore legitimately-received requests for a route. Second, it must require an attestation from the wireless network driver, to verify that the driver will fulfill its obligations, listed below. If the trusted agent cannot verify the driver, it must refuse to participate in the ad-hoc network. Third, it must monitor reports from the wireless driver as to the physical wireless device's performance, and should metrics drop below a certain threshold, refuse to participate further in the network.

The driver for the wireless network device has three key obligations, and must attest to these to the routing agent. First, it must not have the capability to filter traffic at the driver level when such traffic is received from a trusted agent. Second, it must communicate with the physical network device using secure I/O, to prevent a fake virtual device from being used. Third, it must maintain an accurate record of the device's performance within its capability of measurement. If the device fails to transmit, or becomes physically powered down, the driver must record and report these facts.

We assume for the purposes of this paper that there is a limited set of trusted third parties which sign the public keys of each node, and that all nodes trust each of these signers. However, any public-key infrastructure capable of asserting trust in a public key for a specific purpose will suffice, for example the one discussed for ad-hoc networks by Yi and Kravets [18].

5.1. Protocol

We now describe our enhancements to the Ad-hoc On-Demand Distance Vector [19] (AODV) protocol to prevent network abuse by selfish and malicious nodes. We call this protocol Trusted Computing Ad-hoc On-Demand Distance Vector (TCAODV). We note, however, that our techniques could be adapted to design trusted-computing-based secure protocols using other routing protocols, such as Dynamic Source Routing (DSR).

Each node uses a public-key certificate stored within the TPM trusted root for the purposes of routing in an ad-hoc network. This certificate is broadcast during HELLO messages. Neighbours receiving this certificate should verify it through the signature of the issuer, and store it as the broadcaster's public key if it is valid. If it is not valid, it is silently dropped.

When a node A wishes to establish a route to node B, A will broadcast an RREQ. This is done identically to AODV, except the entire packet is signed with a sealed signature by A, using integrity metrics from its routing module.

A node receiving an RREQ first verifies the signature, using the key previously received for that node in a HELLO message, and determines if the measurements provided are trustworthy. If no such key is available, or the verification fails for any reason, the packet is silently dropped. Otherwise, the packet is processed. If the RREQ's destination is not the receiving node, and the receiving node does not have a route to the destination in its cache, it will forward the RREQ. It first strips off the signature made by the forwarder, replacing it with its own signature and integrity measurements. It adds the forwarder to its routing table as the reverse route to A.

If the receiving node is the destination, or has a cached route to the destination, it generates an RREP message, unicasting it back along the already-established reverse route. This operates similarly to the RREQ, in that each node will sealed-sign the packet with its integrity metrics. Each node can then verify the signatures of the forward route and establish the route in the routing table.

Each valid RREP that the source receives corresponds to an accurately-represented route through trusted nodes to the destination. However, this alone does not guarantee that the node will continue to be well-behaved. An untrusted device could simply broadcast traffic on the route once it was established. Therefore, a per-route symmetric-encryption key is established to ensure that only trusted nodes along the path can use the route. Part of the process required for a routing agent to be considered trusted by other routing agents (and thus, have a signature verified that is accepted as a trusted routing agent) is that routing agents must protect the symmetric key from being read by other processes or revealed to any outside source.

After receiving a RREP, or any time the route between A and B changes, A will transmit a new packet, called RKEY, along the route. A randomly generates a symmetric key and a route identifier, attaches a timestamp, encrypts them with the public key of the next hop[1] in the

[1] Some public-key cryptosystems require encryption to be performed using a separate keypair to the keys used to sign. If such a system is used, the 'public key' will simply be a pair of public keys, with one valid

route, and sends it along the route to B. The next node in the route decrypts the key, stores it as the routing key associated with the route identifier, re-encrypts it with the public key for the next hop, and relays it. When this key reaches B, the entire route has knowledge of a symmetric key for the route.

All traffic sent along the route is encrypted using this symmetric key. This includes all routing-layer headers, with the exception of the route ID. When a node receives traffic to forward, it looks up the route ID, and if it has a key for that route, decrypts and verifies the headers of the packet. Should the headers be valid, the packet is forwarded unmodified. If verification fails, or should the route ID not be known to the node, the packet is silently dropped.

Route errors are signalled using the RERR message. RERR messages are encrypted using the route key, and transmitted along the route. A node receiving a valid route error message should discover a new route as described above, but should attempt to reuse the route key and ID if possible. An RKEY packet must still be sent along a new route even if the key is reused, to advise any new nodes of the key and route ID.

6. Comparison

We present a discussion comparing the basic properties of TCAODV to existing protocols described in Section 3.

6.1. Security Challenges

We propose several properties that we would like to ensure are true of nodes participating in the ad-hoc network, that covers a variety of selfish and malicious behaviour.

1. **Routing Participation:** Nodes must always participate in the routing process.
2. **Routing Honesty:** Nodes must report metrics honestly, present their correct identity, and accurately report link errors.
3. **Traffic Forwarding:** Nodes must forward traffic as required by the routing protocol.
4. **Bandwidth Allocation:** Nodes must not transmit faster than established bandwidth limits.
5. **Confidentiality:** Nodes must protect the content and headers of communication from eavesdropping.
6. **Authentication:** Nodes must not impersonate another node in the network.

for signing, and one valid for encryption.

	1	2	3	4	5	6
TCAODV	X	X	X	X	X	X
ARAN		X				X
SAR					X	
Watchdog		X	X	X		
CONFIDENT		X	X	X		
AD-MIX	X		X		X	
Nuglets	X		X			
Cell	X		X		X	X

Table 1. Security Challenges Addressed by Each Protocol

In Table 1 we list the six security challenges identified above, and which protocols are able to offer some level of protection for them. Each protocol is only taken in isolation. In some cases multiple protocols could be combined to address more threats, at the expense of the overhead of multiple security protocols. We note that our trusted-computing solution addresses all of the listed threats effectively. None of the other protocols described can address all of these threats on its own concurrently.

6.2. Overhead

While our approach can address a wider selection of plausible threats than existing protocols, we must examine the cost of doing so. A security protocol is of limited value if it covers more threats but does so at prohibitive cost.

We first observe that TCAODV adds the requirement of a public-key signature operation on each RREQ and RREP packet transmitted or forwarded, and a signature verification on each such packet received. The RKEY adds a public-key encryption and decryption operation for each node in the route and increases the time required to establish the route. Each data packet must be encrypted at the source using symmetric-key cryptography, and decrypted at the destination. Each node on the route must decrypt enough of the packet to verify the packet is authentic. While it might be considered that these cryptographic costs could be overwhelming to an ad-hoc node, we note that any trusted computing platform must, by its very nature, have cryptographic hardware as a component, and therefore the execution time for performing these operations is likely not excessive.

Each routing packet will be significantly larger, by at least tens of bytes. The size increase of each data packet is minimal. If all nodes are trusted, the resulting routes are unaffected by our protocol.

AD-MIX adds at most three public-key encryptions and decryptions, but does so for every data packet instead

of just routing messages. Furthermore, routes are chosen specifically to be longer than necessary. Additional routes may occasionally need to be discovered to locate 'poles.' We therefore believe that our solution is notably less costly than is AD-MIX.

ARAN has similar costs to TCAODV, with public-key operations being required along the established route.

The protocol underlying Nuglets requires each node to establish a symmetric key when they become neighbours for the first time. This requires several public-key operations at each meeting of nodes, and could become extremely expensive for a rapidly changing network. However, for a largely static network, this allows symmetric-key cryptography to be used at each node, which is a far less expensive proposition. Thus, in the dynamic case, nuglets is likely more costly than our solution, while it would be approximately equal in the static case. Our approach does not suffer from this issue, being equal in cost whether the network is static or dynamic (subject to route failure).

SAR and the cellular approach both use pre-distributed symmetric keys and thus suffer very little overhead. The SAR approach may additionally increase route length, but does so by design to avoid forwarding through nodes that could potentially perform traffic analysis. It must be noted, however, that while both of these approaches are potentially less costly than our solution, neither are feasible in a general ad-hoc network where pre-distribution of symmetric keys is not possible. Any solution that resolved this problem of pre-distribution would likely add the same public-key cost that our solution requires.

6.3. Key Infrastructure Requirements

TCAODV requires a public-key infrastructure (PKI) capable of verifying that a public key is from a trusted hardware module. This (or other) infrastructure should also be able to determine trust in a set of integrity metrics for a routing module. This is a significantly easier problem than that of certifying an identity, since the responsibility rests with the hardware manufacturer rather than end users. Further, the cost is a function of the number of hardware variations, rather than the number of users, and thus will be dramatically lower.

The AD-MIX and ARAN protocols both require the use of public-key cryptography with a PKI capable of establishing trust between any pair of nodes, which, again, is a harder trust problem, for the reason noted above. The nuglets approach uses a secure co-processor, requiring similar PKI to our system.

The cell-based system avoids the need for a PKI, but requires a full cell-based infrastructure, and requires the base station to share a key with each node. This requirement is similar to SAR, which requires pre-distributed shared keys. Neither approach is suitable in the general case of multiple administrative domains where there may not be an authority universally trusted to assign these keys.

CONFIDANT does not require significant infrastructure, but is insufficiently specified on various critical points, such as the 'friends' structure. In particular, we believe it would require a PKI to resolve these issues.

6.4. Vulnerabilities

TC ad-hoc routing is dependent on the security of the underlying trusted-computing infrastructure. A node that is able to compromise a trusted module could use the keys contained within to act arbitrarily, while still being trusted by other nodes. The node could misbehave until such compromise is detected, and key-revocation methods invoked. The nuglets system has a similar vulnerability, with its effectiveness predicated on the security of the underlying secure coprocessor.

Our approach further relies upon the ability of the wireless network adapter's driver to correctly monitor the performance of the physical transmitter, which even with secure I/O, is not guaranteed if the device can be subtly modified.

Many proposals have been circulated for the addition of "owner override" abilities to trusted computing hardware. Such a mechanism would allow a TPM owner to explicitly choose to report incorrect measurements in a remote attestation. Such a mechanism would be fatal to our approach, since it would allow a selfish user to pretend to be trustworthy.

None of the systems can address physical- or medium-access-control-layer attacks. In particular, if it is possible (outside the detection of the wireless driver) to disable transmission, a node could behave selfishly. It is likely that any such attack is expensive to orchestrate. Blocking neighbours from transmission is also possible by broadcasting noise or using the MAC layer to continuously force other nodes to delay transmission. These issues are alleviated somewhat in economic models, since the node has motivation to cooperate. Monitoring techniques can detect such behaviour, but cannot address it since it can generally be difficult to detect the true source of interference.

Monitoring approaches (watchdog, CONFIDANT) do not effectively address collaborating malicious nodes, since a monitoring node cannot detect that a collaborating node was ignoring the misbehaviour of their partner malicious nodes unless it can itself witness the behaviour of the malicious nodes. Collaboration is not feasible in our TC-based method.

6.5. Side Effects

TCAODV does not lead to any significant negative consequences, except for the requirement that only trusted nodes can take part in the network. By contrast, the ARAN protocol makes several sacrifices. It requires the use of round-trip time as the routing metric, tying routing to the security implementation. This is a significant problem, as there are several alternative routing metrics that are more effective in ad-hoc networks. Further, the ability to cache a route is lost, since the destination itself must sign each route reply.

The nuglets protocol has difficulty with starvation, since the mechanism for introducing nuglets to the society is not defined, and the loss of packets results in the nuglets it carries being lost as well. Furthermore, edge nodes will be starved of nuglets. While starvation is not as large an issue in the cellular approach where the currency is backed by money, users may become annoyed if consistently billed more than peers closer to a base station.

7. Conclusions

By using the security guarantees of trusted computing, we have proposed an approach to ad-hoc networking that is resistant to many forms of unsocial nodes, including selfish nodes which refuse to forward traffic and nodes that attempt to maliciously manipulate the routing process. Our routing protocol operates at a very low cost compared to existing secure techniques, and addresses many major security challenges that exist in ad-hoc networks.

7.1. Future Work

We do not foresee that all computers will be trusted devices immediately; rather, their deployment will likely be gradual. We would therefore like to make use of untrusted, but honest, nodes in a trusted-computing ad hoc network. If there is a choice between routing through untrusted nodes and not establishing a path, it may well be worthwhile to allow the node to participate. The protocol would have to change to accommodate this. Another possible avenue would be to examine a mixed trusted-untrusted network, where untrusted nodes could participate while monitored by a trusted node.

One issue that is not effectively addressed is that of physical-layer manipulations. Trusted computing could also be used to enforce a reputation system, which could compliment secure routing by giving additional motivation to nodes to participate.

Finally, an implementation is required to determine the exact characteristics and overhead of the protocol in practice. In particular, the interaction of the latencies of the trusted hardware with routing, and the actual size of additional data needs to be studied.

References

[1] E. Batista, "Mesh less cost of wireless," Wired News, Feb. 2003. [Online]. Available: http://www.wired.com/news/business/0,1367,57617,00.html

[2] "Secure cryptoprocessor," Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Secure_cryptoprocessor

[3] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE Wireless Communications*, Oct. 2003.

[4] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *proc. IEEE Intl. Conf. on Network Protocols*, Nov. 2002.

[5] S. Yi, P. Naldurg, and R. Kravets, "Security-aware ad-hoc routing for wireless networks," in *proc. 2nd ACM Intl Symp. on Mobile ad hoc networking & computing*, 2001.

[6] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *proc. Mobile Computing and Networking*, 2000.

[7] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks," in *proc. 3rd IEEE/ACM Intl. Symp. Mobile Ad Hoc Networking & Computing (MobiHOC)*. IEEE, June 2002.

[8] S. Sundaramurthy and E. M. Belding-Royer, "The ad-mix protocol for encouraging participating in mobile ad hoc networks," in *proc. 11th IEEE Intl. Conf. Network Protocols (ICNP'03)*, 2003.

[9] J.-P. Hubaux and L. Buttyán, "Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks," Swiss Federal Institute of Technology, Tech. Rep. DSC/2001/001, 2001.

[10] N. B. Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson, "A charging and rewarding scheme for packet forwarding in multi-hop cellular networks," in *proc. MobiHoc '03*, June 2003.

[11] "Trusted computing," Wikipedia: The Free Encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Trusted_computing

[12] P. England, B. Lampson, J. Manferdelli, M. Peinado, and B. Willman, "A trusted open platform," *IEEE Computer*, July 2003.

[13] V. Haldar, D. Chandra, and M. Franz, "Semantic remote attestation - a virtual machine directed approach to trusted computing," in *proc. USENIX Virtual Machine Research and Technology Symposium*, May 2004.

[14] P. England and M. Peinado, "Authenticated operation of open computing devices," in *proc. 7th Australasian Conf. on Information Security*, Aug. 2002.

[15] *TCG Specification Architecture Overview*, Version 1.2 ed., Trusted Computing Group, Apr. 2004.

[16] J. Burt and M. Hachman, "National semiconductor unveils 'trusted' chip for pcs," eWeek, Sept. 2004. [Online]. Available: http://www.eweek.com/article2/0, 1759,1646895,00.asp

[17] "Ols: Linux and trusted computing," LWN, July 2005. [Online]. Available: http://lwn.net/Articles/144681/

[18] S. Yi and R. Kravets, "Practical pki for ad hoc wireless networks," University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2002-2273, Aug. 2001.

[19] *RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing*, Network Working Group, July 2003.