This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2006 proceedings.

Fully Self-Organized Key Agreement for Ad-Hoc Wireless Networks

Bartlomiej Sieka
Computer Science Department
University of Illinois at Chicago
bsieka@cs.uic.edu

Ajay D. Kshemkalyani
Computer Science Department
University of Illinois at Chicago
ajayk@cs.uic.edu

***Abstract*— This paper proposes a self-organizing bootstrapping protocol for establishing authenticated channels as well as secure identifiers in peer-to-peer networks. Specifically, the paper makes the following contributions. (1) It proposes a fully self-organized protocol that establishes an authenticated communication channel between nodes of a wireless ad-hoc network. This authenticated channel can then be used to establish a secret communication channel between nodes. This is the main contribution. (2) The protocol design also provides a secure identifier framework that is resilient to impersonation. The authentic identifiers it establishes can be used to associate network (and upper) layer identifiers to prevent spoofing. They can also serve as a reliable basis for reputation management protocols. The self-organized bootstrapping is a useful feature for designing autonomic systems.**

I. Introduction

A. Motivation

Creating a secure communication channel between two nodes in an ad-hoc network is an important problem because it forms the basis of all secure protocols in a distributed system. A secure channel is a channel that has the following properties.

- (secrecy): the channel is immune to eavesdropping,
- (integrity): the messages passed can not be altered without that being noticed by their respective receivers, and the communication parties are mutually authenticated,
- (availability): a malicious entity can not disrupt the operations of the channel.

In this work we will focus only on the two first properties, i.e., on channel secrecy and channel authentication. It is worth noting that the problem of providing an authenticated channel underlies the problem of providing a secret channel. In most practical scenarios, channel secrecy is provided by symmetric cryptography. This approach requires that the two communicating parties share a common secret (the secret key). Thus some method of providing the secret key is needed – that is the purpose of key establishment protocols. Key establishment can be performed as *key transport* or *key agreement*.

- *Key transport.* Here, one entity generates the key and transports it securely to the other party, which makes it clearly not a viable solution for ad-hoc networks.
- *Key agreement.* Here, both parties wishing to establish a shared secret key contribute to the computation and communication.

Note that agreement protocols rely on the existence of a small-size authenticated channel ([1]). Consider a scenario where public key cryptography is used. Here, the authenticity of the public key must be verified by some out-of-band communication channel. Even though the size of that prerequisite out-of-band authenticated channel is very small compared to the size of the authenticated (and possibly secure) channel that results from the use of cryptographic methods, observe that the out-of-band channel must itself be authenticated. The main purpose of the out-of-band communication is to authenticate the identifier of the communication party, i.e., to know the identity of the sender. It is clear then, that we need to provide an authenticated channel before we can think about other services and protocols.

B. Related Work

The existence of secure communication channels is crucial to many aspects of an ad-hoc network operation, probably the most important one being secure routing. The problem of securing ad-hoc networks is an active area of research. The proposed schemes fall into three categories.

1) *Trusted third party protocols.* These protocols rely on the existence of a *trusted third party (TTP)*. The role of the TTP can be played by a Certificate Authority, a base station, a selected node, etc. This approach implies centralization of vital network services and thus does not seem well suited for the ad-hoc scenario.
2) *Protocols using prior context.* It appears that it is easier to secure ad-hoc networks once we assume that the nodes share some *prior context* before the network operation begins. A prominent family of methods using this approach are the protocols that assume an off-line secret key pre-distribution phase. Such methods have received a lot of research attention recently, especially in the sensor networks context ([2], [3], [4], [5]). Note that this approach requires that the nodes have a shared prior context before the deployment. This is not always practical and we seek to provide a solution that is free of this limitation (i.e., it is fully self-organized).
3) *Self-organizing protocols using out-of-band channels.* The most natural approach to take for ad-hoc networks is *self-organization*. In this approach, there are no special nodes, no infrastructure, no centralized configuration

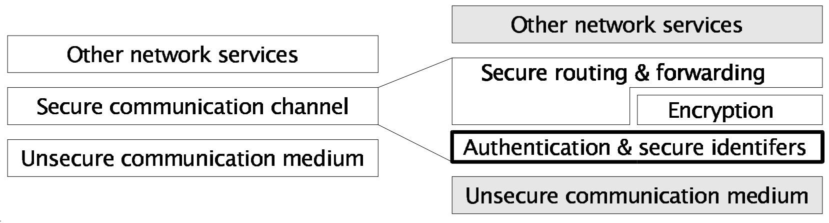
Other network services
Secure communication channel
Unsecure communication medium
Other network services
Secure routing & forwarding
Encryption
Authentication & secure identifers
Unsecure communication medium

Fig. 1. Network service layers.

point, no shared prior context. There has been a lot of research in this area, but unfortunately many proposals have a limiting assumption, that there exists an *out-of-band authenticated communication channel* ([6], [7], [8], [9], [10], [11], [12]).

C. Contribution

It should be clear by now, that methods for establishing authenticated communication channels are important in the ad-hoc peer-to-peer scenario.

- An authenticated channel is a necessary prerequisite for key establishment protocols, that are used to build secure communication channels.
- The authenticated channel can also be used as a building block for a secure routing algorithm. Once we have a secure routing in place, we can implement any other network services we wish.

The *main contribution* of this paper is the self-organized protocol to establish authenticated channels in a peer-to-peer ad-hoc network. This protocol has none of the drawbacks of the three categories of protocols (1)–(3) above, and exhibits the following **key properties**.

1) No trusted third party.
2) No requirement to share a prior context.
3) No out-of-band authenticated channels, no privileged side channels, no prior security associations, no prior bootstrapping of authenticated identifiers, no a priori known public master key, no tamper-proof hardware.

Let us underline two important aspects of this work. One is that the proposed method stresses on the self-organization principle. Second is that the protocol's goal is to provide secure communication. Both these aspects are important for designing autonomic systems.

Although the protocol is simple, its importance stems from the fact that it is the first general protocol to satisfy the above key properties, and it can serve as the base for the development of further protocols exhibiting *all* the above properties. The protocol to implement the authenticated communication channel uses an unsecure communication medium, as depicted in Fig. 1. Any network service, including secure routing and forwarding, can use the authenticated channel. However, even with authenticated channels, the authenticity of communicating parties must be considered in the context of their identities. Therefore, as part of our solution, we also address the issue of the identity model of the network, proposing the use of *secure identifiers*. This is the *second contribution*. Thus, while establishing authenticated channels, our solution inherently provides a solution to the problem of providing *secure identifiers*. We propose to use the hash of the public key for the identifier of a node. Although this idea has been used before (e.g., [13], [14]), we remark that the use of the hash of the public key as a secure identifier has always been used in the context of a specific protocol. For example, [13] employs a very similar method to secure binding update messages in the mobile IPv6. However that proposal is very closely tied to a very particular application. On the other hand, our approach does not make strict assumptions about the properties of adjacent protocol layers, thus being suitable for incorporation in many contexts. We solve the problem of providing secure identifiers in a *protocol-independent manner*. *Any* higher layer protocol (see Fig. 1), including routing and forwarding, can use these secure identifiers. Note again, that we make no assumptions as to the available network services. We consider the scenario where we must work with an unsecure and untrusted communication medium.

We now describe a result that formulates our problem in a very similar way. Bobba et al. in [15] start by presenting the cyclic dependency between the secure routing and security services. This cyclic dependency is depicted in Fig. 2.

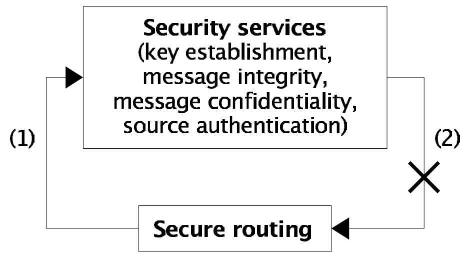

Fig. 2. Dependency cycle between secure routing and security services. Figure is adapted from [15] to reflect our approach of removing dependency (2) to break the cycle.

Then they propose a self-organized approach for bootstrapping security for the routing layer. They break the cyclic dependency by proposing a secure routing protocol that does not depend on any security services. Thus, they remove dependency edge (1) to remove the cyclic dependency. Their solution is embedded in the routing protocol and does not establish secure identifiers for the higher layers. We take a complementary route to breaking the cyclic dependency: we aim at removing dependency edge (2) in Fig. 2, by showing how to implement security services without relying on the routing. Bobba et al. [15] had stated that "Removing dependency (2) would be impractical because it would require that the nodes implementing security services be reachable by all other nodes in the network by a fixed set of routes." However, our approach does remove dependency (2) because we do not rely on the existence of a routing protocol in our approach; hence the nodes need not be reachable by a set of routes.

II. Problem Statement

We are interested in devising a means for authenticated communication within a peer-to-peer ad-hoc network. As the authenticity of communication is very closely related to the identity of communicating parties, we also aim at a method for providing network identifiers.

The problem we solve is defined as follows. Given a set of nodes, we want them to be able to establish an authenticated network (*AN* for short). More formally, for any given node j in the *AN* we want the following property.

$$Key\ Property:\ \forall i \in AN, PK_i^j = PK_i^i \qquad (1)$$

PK_i^j is the public key of node i as known by node j. Note that PK_i^i can be seen as the authentic public key of node i. In other words, the above property states that all nodes in the *AN* know the authentic public keys of all other nodes in the *AN*. The authenticity of the public key must be considered in the context of the entity that is in the possession of the corresponding private key. Hence we define PK to be an authentic public key of node i if and only if (a) node i possesses the corresponding private key and (b) the identity of node i can be verified to be bound to that key. Note that we do not require that the node i generates both the public and private keys. In case they are generated by some other party, we implicitly assume that there exists a trust relationship between the key generator and that node. It should be noted that this assumption is very subtle and should be carefully verified when considering a larger context.

Observe that the problem stated above is in its essence the problem of bootstrapping the network. We assume that network layer functions are not available yet, i.e., nodes do not forward packets and do not implement any routing algorithm. Thus the only communication means that can be used is the radio broadcast in a wireless network or communication with neighbors using the incident links in a point-to-point network.

III. Solution

Let us begin with a description of the identity model we use. In this context, we would like to emphasize a quote from [12]: "in an ad-hoc network there may be no *a priori* reason to distinguish between the different nodes" (Section 2.2, last sentence). In this spirit we propose that the identifiers are self-appointed. We start by assuming that every node has a (private, public) key pair. Following a method as in [13], [14], we propose that the node identifier be a hash of the node's public key. For example, for node i we have $id_i = hash(PK_i)$ where id_i is the assumed identifier of the node, and PK_i is its public key.

Observe that if the Key Property (Equation 1) holds, then all the nodes in the *AN* know each others' identifiers.

Let us now turn to the description of our protocol. The notation we use to describe protocol messages is as follows. When the contents of the message are relevant, we use $MSG(contents)$ to denote the message, where $contents$ lists the actual fields of the message. When we want to refer to a given message more generally, we use the shorter `TYPE` notation, where `TYPE` is a place-holder for the type of message in question (e.g., `JOIN` or `ACCEPT` or `UPDATE`). The type of message is usually the first field of its contents – it can be an integer number or a string and must be unique across different message types. PK_i and SK_i are used for node i's public and private key, respectively. To indicate that the integrity of the message M is protected by a digital signature, we use the notation $S_{SK}(M)$, where SK is the private key used for signing. When describing the flow of messages in the protocol, we use the $\rightleftharpoons$ symbol to denote radio broadcast.

The following data structures are used. Every node maintains a key table containing the mapping between the identifiers and public keys, as well as some additional information. For every node i in the network, the following data are maintained for each node j.

- id_i: Node identifier. In our approach, the identifier is a hash value of the public key of the node, i.e., $id = hash(PK)$. Note that the node identifier is a fixed-length sequence of bits.
- PK_i: Public key of the node.
- seq_i: An integer that gives the sequence number copied from the last `JOIN` message from i.
- $time_i$: An integer that gives the local time when the most recent message from node id_i was seen.

Node j initializes its key table with one row containing its identifier and public key. Note that both the seq_j and $time_j$ are not relevant in this case. Table I illustrates the format of a node's key table. The number of entries in the key table of node j is denoted as N_j.

node	id	PK	seq	time
1	id_1	PK_1	seq_1	$time_1$
2	id_2	PK_2	seq_2	$time_2$
…	…	…	…	…
j	id_j	PK_j	seq_j	$time_j$
…	…	…	…	…
N_j	id_{N_j}	PK_{N_j}	seq_{N_j}	$time_{N_j}$

TABLE I

A key table maintained by a node j.

We introduce the following terms: $KeyTable_j$ and $KeyTableDelta_j$. $KeyTable_j$ denotes two columns of node j's key table: the PK column and the seq column. $KeyTableDelta_j$ denotes the set of (PK, seq) entries from

j's key table, that have been modified since the last time a UPDATE message was sent by node j.

The proposed protocol to build an *AN* is outlined in Table II, and is organized around two main scenarios.

- *Node Join:* A node outside of an *AN* wants to join *AN*. This node may or may not be a member of some other AN. If it is a member of another AN, then the joining scenario is equivalent to a merge of two networks. The entire network AN can be viewed as having been formed from a succession of subnetwork merges. In each merge, one node initiates the protocol for the joining of two (or more) sub-networks by sending a JOIN message. The nodes that receive a JOIN respond with an ACCEPT message. A single JOIN may trigger more than one ACCEPT, one from each node within radio coverage. Note that the nodes that send the ACCEPT messages in response may belong to different ANs in the general case. Let the sub-network of the node that initiates the JOIN be denoted as AN_{join}, and let the sub-network(s) that accept this initiated merge be denoted as AN_{accept}. When $|AN_{join}| = 1$, we have the special case when a single node wants to join an existing AN on awakening or recovering.
- *Key Update:* The contents of the key table change over time. Entries can be added as a result of receiving the JOIN, the ACCEPT, or the UPDATE message. Upon a change in the key table, the node should notify others by broadcasting the UPDATE message itself. For node j the following message should be used.
$j \rightleftharpoons MSG(\texttt{UPDATE}, KeyTableDelta_j, S_{SK_j}(\texttt{UPDATE}, KeyTableDelta_j))$

We now discuss the details of the protocol, distinguishing the following events.

1) *Send* JOIN: This event pertains to a node in AN_{join}. When a node i wants to join another *AN*, it should generate a pair of keys (PK_i, SK_i), where PK_i is the public key and SK_i is the corresponding secret key and then broadcast the JOIN message. (If it has already generated this pair before, but is now joining a new AN due to mobility or other reasons, it does not have to regenerate a different key pair). The seq_i field of the message is a sequence number that is guaranteed to be increasing with time for a given node and the PK_i is the public key generated.
2) *Receive* JOIN: This event pertains to a node that is not a member of AN_{join}. When a node j receives a JOIN message, it should first verify the validity of the digital signature of the message. If the signature is invalid, the message should be discarded and no further action should be taken. If the signature is valid, j should compute the hash value of the public key $id_i = hash(PK_i)$ and check if there exists an entry with id_i for node i in its (i.e. j's) key table.
    a) If the id_i entry does not exist, then the new entry should be added with the computed id_i and both PK_i and seq_i values copied from the JOIN message. The $time_i$ field should be set to node j's local time. The node should then broadcast the ACCEPT message, and also broadcast the UPDATE message (see step 5).
    b) If the id_i entry does exist, let k be the index of that entry in the key table; hence $id_i = id_k$. Also the corresponding public key and sequence number are denoted by PK_k and seq_k, respectively. Let us now consider three cases.
        i) $PK_i \neq PK_k$: There is a collision in the hash function. The JOIN message should be discarded. Dealing with collisions is discussed in Section IV.
        ii) $PK_i = PK_k$ and $seq_i < seq_k$: This might indicate an attempt to mount a reply attack. (Note that the JOIN messages are not subject to the regular network routing and forwarding, hence this case does not indicate routing loops or other network-layer problems.) This can also mean that node i sequence counter has wrapped around. The JOIN message should be discarded.
        iii) $PK_i = PK_k$ and $seq_i \geq seq_k$: This may indicate that the node i is sending spurious JOIN messages. The sequence number seq_k should be updated to seq_i, the ACCEPT message should be broadcast, and then the UPDATE message should be broadcast (see step 5).
3) *Send* ACCEPT: This event pertains to a node that is not a member of AN_{join}. On receiving a JOIN message from node i (Step (2)), when a member node j determines that i can be admitted to its *AN* without an identifier conflict, it broadcasts the ACCEPT message. The $KeyTable_j$ field of the message should contain all the (PK, seq) pairs from the key table of node j.
4) *Receive* ACCEPT: This event pertains to the specific node in AN_{join} that sent the JOIN message. After broadcasting the JOIN message, when the node receives a corresponding ACCEPT message, it is considered to be a member of the $AN_{join+accept}$ network. The node should check the signature of the message and drop the message if the check fails. If the signature is valid, the node should add entries from the $KeyTable$ field of the ACCEPT message to its key table. Then the node broadcasts an UPDATE message so that any other AN also within its range can learn of AN_{accept} (see step 5). A node that did not send the JOIN message should drop any ACCEPT message received.
5) *Send* UPDATE: This event pertains to a node that is a member of the AN_{join} or AN_{accept}. New entries are added to the node's key table in the following cases.
    - *Step 2a:* node in AN_{accept} sends UPDATE.
    - *Step 2(b)iii:* node in AN_{accept} sends UPDATE.
    - *Step 4:* node in AN_{join} sends UPDATE.

(1)	When node i wants to join another *AN*, it broadcasts the `JOIN` message: $i \rightleftharpoons MSG(\texttt{JOIN}, seq_i, KeyTable_i, S_{SK_i}(\texttt{JOIN}, seq_i, KeyTable_i))$.
(2,3,5)	When node j, a member of the *AN*, receives the `JOIN` message from node i and determines that node i can join the *AN*, it enters i's data into its key table and broadcasts the `ACCEPT` message: $j \rightleftharpoons MSG(\texttt{ACCEPT}, id_j, seq_i, PK_i, KeyTable_j, S_{SK_j}(\texttt{ACCEPT}, id_j, seq_i, PK_i, KeyTable_j))$. Then node j broadcasts the `UPDATE` message: $j \rightleftharpoons MSG(\texttt{UPDATE}, KeyTableDelta_j, S_{SK_j}(\texttt{UPDATE}, KeyTableDelta_j))$.
(4,5)	When a node i that has sent a `JOIN` receives the corresponding `ACCEPT` message, it updates its key table. Then the node broadcasts an `UPDATE` message: $i \rightleftharpoons MSG(\texttt{UPDATE}, KeyTableDelta_i, S_{SK_i}(\texttt{UPDATE}, KeyTableDelta_i))$.
(6,5)	When a node j inside the AN receives a `UPDATE` message, it updates its key table using $KeyTableDelta$. If new entries are added to its key table, the node broadcasts the `UPDATE` message: $j \rightleftharpoons MSG(\texttt{UPDATE}, KeyTableDelta_j, S_{SK_j}(\texttt{UPDATE}, KeyTableDelta_j))$.

TABLE II

OUTLINE OF PROTOCOL TO ESTABLISH AUTHENTICATED CHANNELS.

- *Step 6:* node in AN_{join} or AN_{accept} sends `UPDATE`.

When new entries are added to the node's key table, the node should broadcast the `UPDATE` message. The $KeyTableDelta$ field of the message should contain all the (PK, seq) pairs that have been updated since the last time the `UPDATE` message was sent.

6) *Receive* `UPDATE`: This event pertains to a node that is a member of AN_{join} or AN_{accept}. When a node receives the `UPDATE` message, it should check the signature of the message. The message should be dropped and no further action should be taken if the signature is invalid, otherwise the node should add entries from the $KeyTableDelta$ field to its key table. It then executes step (5). A node outside of the *AN* should drop the `UPDATE` message.
7) *Key Timeout:* Every node should maintain a timestamp associated with every entry in its key table (field $time$ in the key table). Node i should update the timestamp to its current time for entry j every time it receives a message sent by node j. Note that the message need not be addressed to node i. An entry should be deleted from the key table if the timestamp is older than a specified threshold value. The default can be set to a high value to minimize overhead under normal operation.

IV. Security Analysis

Let us now focus on the security of the proposed solution. Recall that we assume no prior context shared between nodes and still want them to be able to perform some mutually desired interaction. This is somewhat akin to a real-life situation in which we meet a complete stranger with whom we want to interact in some way. Note that in general all we can know about a person comes from that person himself, e.g., we can learn the identity of that person from him saying "Hello, I am John Smith". It is of course prudent not to completely trust a stranger, but to build the trust over time instead, associating its level, as well as other attributes, with the name John Smith. If after some time all the interactions allow us to achieve our goals, and the person has been well-behaving so far, we do not really care if that person's name is John Smith, or something else. All that is important is that initial communication phase and the subsequent process of associating trust.

This parallel is useful in understanding of our proposed scheme. We observe that in the absence of prior context and lack of infrastructure, we cannot really do better that to trust the other communication party with what he says. In our protocol, the initial trust is restricted to the identity (and the public key), as communicated by the other party. Under this assumption, a *man-in-the-middle* attack looses its meaning. Consider a malicious node that is in between us and some other node and just relays that node's communications to and from us. Since all the messages are signed, the attacker has no other choice than to use his own private key and the corresponding public key as the identifier (otherwise messages we receive would fail the integrity check). Note that from our point of view, all the messages appear as coming from the attacker, since his identity can be bound to the message by checking of the message signature. Observe that if such interactions are satisfactory to us, then it is really irrelevant who the sender of the message is.

In our identity model, impersonating a node is equivalent to being able to generate a signature using that node's private key. The facts that each message carries a digital signature, and that a node's identity is bound to its public key, make *impersonation attacks* impossible. Further, the *replay* and *reflection* attacks against the joining phase are thwarted by the use of sequence numbers.

It is conceivable that two nodes will generate the same public key. However, the probability of such an event is extremely small, i.e. $1/2^{1536}$, assuming a 1536-bit public key. Furthermore, even if public keys are different, a collision in the hash function is still possible. However – by the birthday paradox – if we assume 32-bit identifiers, then there would have to be an average of $1.2 * 2^{16}$, i.e. about $7.86 * 10^4$ nodes in the network for the probability of collision to exceed $1/2$. If we assume 64-bit identifiers, then about $5.15 * 10^9$ nodes are required for the probability of a collision to exceed $1/2$.

It should be noted that all practical applications will require the identifier to be bound to some other, higher-level information (a name of the person, an IP number, a MAC address, an organization name, etc). Note that the existence of an authenticated channel provided by our protocol allows for those higher-level associations to be established in a secure manner using cryptographic techniques (both symmetric and public key schemes can be used). Let us address the *Sybil* attack in the context of that higher-level information. On the low level, there is nothing that would prevent an entity from generating multiple (public, private) key-pairs and thus assuming multiple identities. This is due to the assumed characteristics of the communication medium, i.e., its unreliability and its broadcast nature. Thus, mechanisms on higher levels must be used to address the *Sybil* attack threat, for example, reputation management approaches mentioned below.

One specific example of a higher level data that can be associated with the identify of the node is the reputation. There exists a large body of research pertaining to managing reputations of nodes in peer-to-peer and ad-hoc networks ([16], [17], [18], [19]). These approaches can be combined with our protocol to provide a mechanisms to increase (or decrease) the trust level between nodes. A crucial observation here is that, to our knowledge, the reputation management schemes rely on the existence of a secure outside communication channel, which our scheme provides.

We would like to stress the fact that our protocol is fully self-organized. In that respect it differs from the family of identity-based cryptosystems, where a trusted third party is required to compute the private key in the setup phase.

V. Conclusions and Future Work

The contribution of this paper is two-fold. We propose a fully self-organized protocol that establishes an authenticated communication channel between nodes of a wireless ad-hoc network. The protocol does not rely on the existence of a Trusted Third Party, the nodes do not need to share a prior common context, and no out-of-band communication channel is required. The scheme is independent of the upper layer protocols, in particular, it is not an extension to any existing routing protocol. The resulting authenticated channel can be further used to establish a secret communication channel between nodes. The protocol also provides a secure identifier framework that is resilient to impersonation. Our authentic identifiers can be used to associate network (and upper) layer identifiers to prevent spoofing. They can also serve as a reliable basis for reputation management protocols. Although the protocol presented is for an ad-hoc wireless network which is a peer-to-peer broadcast network, it can be easily adapted to all other ad-hoc networks that communicate via point-to-point channels.

The future research directions are to extend the protocol so that it can handle multiple independent authenticated networks. This would be useful in a scenario where more than one ANs need to share the same bandwidth. It also allows for one node to be the member of multiple ANs. This is useful not only from the application point of view, but also facilitates the implementation of some network functions, for example, tunneling between different *AN*'s and firewalling.

References

[1] S. Blake-Wilson and A. Menezes, “Authenticated Diffie-Hellman key agreement protocols.” in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, S. E. Tavares and H. Meijer, Eds., vol. 1556. Springer, 1998, pp. 339–361.

[2] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks.” in *ACM Conference on Computer and Communications Security*, V. Atluri, Ed. ACM, 2002, pp. 41–47.

[3] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge.” in *INFOCOM*, 2004, pp. 587–597.

[4] D. Liu, P. Ning, and R. Li, “Establishing pairwise keys in distributed sensor networks,” *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 1, pp. 41–77, 2005.

[5] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, “A pairwise key predistribution scheme for wireless sensor networks.” *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 2, pp. 228–258, 2005.

[6] F. Stajano and R. J. Anderson, “The resurrecting duckling: Security issues for ad-hoc wireless networks.” in *Security Protocols Workshop*, ser. Lecture Notes in Computer Science, B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds., vol. 1796. Springer, 1999, pp. 172–194.

[7] D. Balfanz, D. Smetters, P. Stewart, and H. Wong, “Talking to strangers: Authentication in adhoc wireless networks,” in *Symposium on Network and Distributed Systems Security (NDSS ’02)*, San Diego, California, USA, February 2002.

[8] J.-P. Hubaux, L. Buttyán, and S. Capkun, “The quest for security in mobile ad hoc networks.” in *MobiHoc*. ACM, 2001, pp. 146–155.

[9] S. Capkun, L. Buttyán, and J.-P. Hubaux, “Self-organized public-key management for mobile ad hoc networks.” *IEEE Trans. Mob. Comput.*, vol. 2, no. 1, pp. 52–64, 2003.

[10] S. Capkun, J.-P. Hubaux, and L. Buttyán, “Mobility helps security in ad hoc networks.” in *MobiHoc*. ACM, 2003, pp. 46–56.

[11] H. Deng and D. P. Agrawal, “TIDS: threshold and identity-based security scheme for wireless ad hoc networks,” *Ad Hoc Networks*, vol. 2, no. 3, pp. 291–307, July 2004.

[12] A. Khalili, J. Katz, and W. A. Arbaugh, “Toward secure key distribution in truly ad-hoc networks,” in *IEEE Workshop on Security and Assurance in Ad-Hoc Networks, 2003. Proceedings*, 2003, pp. 342–346.

[13] G. O’Shea and M. Roe, “Child-proof authentication for MIPv6 (CAM),” *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2, pp. 4–8, 2001.

[14] G. Montenegro and C. Castelluccia, “Crypto-based identifiers (CBIDs): Concepts and applications,” *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, 2004.

[15] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh, “Bootstrapping security associations for routing in mobile ad-hoc networks,” in *GLOBECOM’03, IEEE Global Communications Conference, December 1-5, 2003, San Francisco, USA*, vol. 3, December 2003, pp. 1511–1515.

[16] K. Aberer and Z. Despotovic, “Managing trust in a peer-to-peer information system,” in *Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 2001*, 2001, pp. 310–317.

[17] S. Braynov and T. Sandholm, “Incentive compatible mechanism for trust revelation,” in *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, July 2002*, 2002, pp. 310–311.

[18] M. Gupta, P. Judge, and M. Ammar, “A reputation system for peer-to-peer networks,” in *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California, USA, June 2003*, 2003, pp. 144–152.

[19] B. Yu and M. Singh, “An evidential model of distributed reputation management,” in *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, July 2002*, 2002, pp. 294–301.