Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 171 (2007) 43–55
www.elsevier.com/locate/entcs

Secure Node Discovery in Ad-hoc Networks and Applications[1]

Giovanni Di Crescenzo[2]

Telcordia Technologies
One Telcordia Drive, 1K325, Piscataway, NJ, 08854, USA

Abstract

Designing secure protocols over ad-hoc networks has proved to be a very challenging task, due to various features of such networks, such as partial connectivity, node mobility, and resource constraints. Furthermore, their lack of physical infrastructures deprives their users of even basic network functions such as message routing, for which nodes are themselves responsible.
In this paper we consider a very basic network function, node discovery, in ad-hoc networks, where a node with limited network information would like to establish a session with a given number of other nodes in the network (of which the node may not be aware about). We formally define correctness, security and efficiency properties of node discovery protocols, and investigate the problem of designing such protocols under appropriate network topology assumptions. Here, the security of these protocols is against Byzantine adversaries that can corrupt up to a limited number of nodes in the network and make them arbitrarily deviate from their protocol. After presenting some secure node discovery protocols, we show their application to secure service architectures in ad-hoc networks.

Keywords: Secure node discovery, client-server architectures, threshold cryptography, mobile ad hoc networks

1 Introduction

Ad-hoc networks are a collection of wireless nodes, communicating among themselves without the help of any infrastructure such as base stations or access points. As the development of wireless services such as cable TV, secure audio conferencing, visual broadcasts, military command and control grows, so does the number of civilian, financial and military applications of research on ad-hoc networks. As it is expected from the importance of these applications, security plays a key role for the success of the development of such wireless services, and the research area of

[1] Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

[2] Email: giovanni@research.telcordia.com

doi:10.1016/j.entcs.2006.11.014

security and cryptography over ad-hoc networks is currently growing at a very fast pace.

As a consequence of the lack of infrastructure in ad-hoc networks, even basic communication functions such as message routing are a burden on the wireless nodes themselves. Furthermore, limited resource of wireless devices may significantly constrain the connectivity among nodes, and messages of any type of communication session have to necessarily go through multi-hop paths before reaching destination.

In this paper we consider *node discovery* protocols in ad-hoc networks. In a node discovery protocol a node with limited network information would like to establish a session with a given number of other nodes in the network (of which the node may not be aware about). Here, we model the process of establishing a session by a simple query-reply interaction, being unique with respect to the identities of the parties involved. Node discovery over ad-hoc networks should be considered a very basic network functionality, even more than message routing, or route discovery (a typical component of message routing). As a result, we expect node discovery subprotocols to be useful for several protocols over ad-hoc networks, including protocols for client-server architectures, which we exemplify in this paper. We formally define correctness, security and efficiency properties of node discovery protocols, as follows: by *correctness*, we mean that when all parties are honest, an initiator node can discover enough nodes in the network; by *security* (against a Byzantine adversary corrupting up to τ nodes, for some known parameter τ), we mean that in the presence of such adversary, the initiator node can still discovery enough nodes in the network; finally, the main *efficiency properties* we focus on are the number of session establishment attempts from the initiator node, and the amount of interaction between the initiator node and the rest of the network. (Both metrics crucially affect standard efficiency metrics in ad-hoc networks, such as latency, bandwidth, battery life, energy consumption, etc.) We then investigate the problem of designing such protocols under appropriate (and as weak as possible) network topology assumptions. We present 3 protocols, the most interesting requiring reasonably weak topology assumptions and presenting satisfactory efficiency properties. This work builds on previous work we did in [5,6], where variants of some protocols presented there may be rephrased as node discovery protocols secure against static adversaries under appropriate topology assumptions. In this paper, we surface and formalize the notion of node discovery, we present new protocols secure against adaptive adversaries (namely, adversaries corrupting nodes at any time during the protocol) and we suggest a way to analyze and compare the efficiency of such protocols. We conclude by showing applications of secure node discovery protocols to secure service architectures in ad-hoc networks, under symmetric or asymmetric cryptographic primitives.

2 Model and Definitions

We first describe our modeling of a mobile ad-hoc network, including assumptions on the network topology and on the knowledge of parties. We then define our notion

of secure node discovery protocols in mobile ad-hoc networks.

2.1 Network modeling

MANETs put severe constraints on the connectivity among parties and, in general, network connectivity among all parties cannot be guaranteed. Because of the wireless nature of the parties' devices, it is not guaranteed that all parties are in the radio range of a given party; in addition, the shape of a radio range of a given party can differ according to the location, time, device power, and device energy. Therefore we model the connectivity among the parties with a *connectivity graph* $\mathcal{G}$, where each node in $\mathcal{G}$ is associated with a different party, and an edge between any two nodes implies that the two associated parties are within radio range of each other (for simplicity, we only consider the case of bidirectional connectivity). In addition, graph $\mathcal{G}$ can vary in time according to *party mobility* or *unavailability*, due to factors such as geographical changes, power disruption or low battery. These events trigger changes in graph $\mathcal{G}$ modeled as *failure* and *creation* of *nodes* and *edges*; therefore the structure of graph $\mathcal{G}$ varies according to such changes (for notational simplicity we omit a time-dependent subscript in the notations related to $\mathcal{G}$ and we assume without loss of generality that n is an upper bound on the number of parties). We also make the simplifying assumption of absence of contention or transmission errors, as these can be dealt with in a modular fashion and at different layers.

Assumptions on the network topology. In the rest of the paper we will use the following assumptions on the topology of graph $\mathcal{G} = (V, E)$ during the entire life of the protocols executed over the network modeled as $\mathcal{G}$. (Here, $\rho, \sigma, \tau_1, \tau_2$ are parameters in $\{1, \ldots, n\}$).

(i) *ρ-neighbor assumption*: at any time, every node in V has at least ρ neighbors

(ii) *σ-connectivity assumption*: at any time, for any σ nodes $N_1, \ldots, N_\sigma$ in V, the graph $\mathcal{G}'$, obtained by removing $N_1, \ldots, N_\sigma$ and all edges incident to them from $\mathcal{G}$, remains connected.

(iii) *(τ_1, τ_2)-reachability assumption*: at any time, for any node $N \in V$ and any τ_1 nodes $N_1, \ldots, N_{\tau_1}$ in $V \setminus \{N\}$, the graph $\mathcal{G}'$, obtained by removing $N_1, \ldots, N_{\tau_1}$ and all edges incident to them from $\mathcal{G}$, contains at least τ_2 nodes reachable from N.

We note that the σ-connectivity assumption implies the ρ-neighbor assumption, for $\rho = \sigma + 1$, and any $\sigma \in \{0, \ldots, n-1\}$ (or otherwise there exists a node with at most $\rho - 1 = \sigma$ neighbors, and removing such neighbors will disconnect the graph). We also note that the (τ_1, τ_2)-reachability assumption, when $\tau_2 \geq n - 1 - \tau_1$, is identical to the the σ-connectivity assumption, for $\sigma \leq \tau_1$.

Assumptions on the knowledge of parties. We will assume that each party can be associated with a unique address; say, a 64-bit IPv6 address, which we will, from now on, simply refer to as *ID*. We will then assume that each party is aware of the her neighbor nodes' ID, but is not required to have additional knowledge about the

ID of other parties or about the remaining topology of graph $\mathcal{G}$ (other than the direct neighbors).

2.2 Secure node discovery in MANETs

Defining node discovery. Informally speaking, in a node discovery protocol a single user at a network node would like to learn about the existence and network reachability of a number of other network nodes. As node discovery can be used in different applications, we will not focus on modes of operation with respect to a specific application, but abstract its mode of operation as a generic 'session establishment'. Specifically, in any such protocol, the goal of a node N will be that of establishing a reliable session with one or more other nodes $N_1, \ldots, N_u$ in the graph. For simplicity, we assume that a single query-reply interaction, where node N sends the query and any node, say, N_i, among nodes $N_1, \ldots, N_u$ sends the answer, is sufficient to establish a reliable session between nodes N and N_i.

Let 1^k be a security parameter in unary and let $\mathcal{G}$ be the connectivity graph of a mobile ad-hoc network. Formally, we define node discovery protocols as a pair of subprotocols: a preprocessing subprotocol, where preliminary information is exchanged among the parties in the network, and a discovery subprotocol, where the preliminary information is used to guarantee successful discovery of any desired number of nodes from any given node. Then an execution of a node discovery protocol is denoted as NDP$\equiv (\Pi_{pre}, \Pi_{dis})$, and consists of one execution of a preprocessing protocol Π_{pre} and polynomially (in k) many executions of a discovery protocol Π_{dis}. Both protocols Π_{pre} and Π_{dis} are run among all parties in the network; protocol Π_{dis} is started by a node, which we will refer to as the *initiator*. For any positive integer t, we require that a *t-node discovery protocol* NDP$\equiv (\Pi_{pre}, \Pi_{dis})$ *in a mobile ad-hoc network* $\mathcal{G}$ satisfies the following requirement:

- *Correctness:* If parties $P_1, \ldots, P_n$ honestly run all executions of protocols Π_{pre} and Π_{dis}, then with probability 1 at the end of each execution of protocol Π_{dis}, the party acting as initiator, on input 1^t, returns in that execution t distinct identities $ID_1, \ldots, ID_t$.

If all nodes honestly follow their protocol, then a trivial node discovery protocol is achieved by a distributed implementation of the following reachability algorithm: in the preprocessing protocol, all nodes discover their neighbors; in the discovery protocol, the initiator P_j discovers its neighbors, and asks them to recursively discover previously undiscovered nodes and forward their identity to P_j, until P_j has discovered at least t distinct nodes (that is, until P_j has received at least t distinct ID's). The problem of node discovery becomes non-trivial in the presence of adversaries that can corrupt some of the parties involved in the execution of protocols Π_{pre} and Π_{dis}.

Defining security of node discovery. We first characterize the power of a (Byzantine) adversary whenever a node discovery protocol is run over a mobile ad-hoc network. A first, basic, attacking strategy for a Byzantine adversary is that of corrupting nodes that participate to session establishment; in particular, if an

initiator is trying to establish a session with a corrupted node, we will assume that the initiator may not be successful in discovering the corrupted node. In addition to corrupting parties that are supposed to be discovered, an adversary can corrupt parties that are responsible for routing ID's to initiators. Even more, an adversary can physically move parties in the neighborhood of an initiator before the protocol starts, so that the client must rely on a smaller number of neighbors for routing purposes, or even replicate each corrupted party into additional parties with the same ID (although, as we will see this latter kind of adversarial strategies is easily dealt with). More formally, we consider a Byzantine adversary that can corrupt up to τ parties at any different times during the execution of the protocol NDP, and can arbitrarily modify their programs for the rest of the protocol, including stopping message routing, moving them and replicating them.

We now study schemes allowing any initiator in a mobile ad-hoc network to discover any given number of nodes, even in the presence of Byzantine adversaries that corrupt up to τ network nodes different from the initiator (or otherwise the node discovery problem trivializes).

Recall that we denote an execution of a node discovery protocol as NDP$\equiv$ (Π_{pre}, Π_{dis}), consisting of one execution of a preprocessing protocol Π_{pre} and polynomially (in k) many executions of a discovery protocol Π_{dis}. For any positive integer t, we require that a *τ-secure t-node discovery protocol* NDP$\equiv$ (Π_{pre}, Π_{dis}) *in a mobile ad-hoc network* $\mathcal{G}$ satisfies the following requirement:

- *Security.* Let $P_j \in V(\mathcal{G})$, for $j \in \{1, \ldots, n\}$, be an initiator in an execution of Π_{dis}, and let A be a probabilistic polynomial time algorithm, called the (Byzantine) *adversary*, that at any time during the protocol can choose up to $\leq \tau$ indices $i_1, \ldots, i_\tau \in \{1, \ldots, n\} \setminus \{j\}$ and play as $P_{i_1}, \ldots, P_{i_\tau}$ until the rest of the protocol, arbitrarily modifying their program (including physically moving them), or even replicating them. We say that an execution of protocol NDP is *successful* if at the end of the execution of protocol Π_{dis}, party P_j, on input 1^t, returns t distinct identities of parties in $V(\mathcal{G}) \setminus \{P_j\}$ that received a session establishment request from P_j, and such that with at least $t - \tau$ of them the session establishment protocol was successfull. Then, for any adversary A, the probability that an execution of protocol NDP is not successful, is negligible.

Some immediate observations include: (1) it is trivially impossible to obtain a τ-secure t-node discovery scheme for any $t > n - 1 - \tau$ (in the rest of the paper, for simplicity, we focus on the most interesting case $t = 2\tau + 1 \leq n$); (2) when $t \leq n - 1 - \tau$, the problem of designing a τ-secure t-node discovery protocol has a trivial solution *on wired networks* where any two nodes are connected and the adversary cannot manipulate the routing among any two nodes, as a node can just contact a new neighbor to discover a new node. The problem becomes non-trivial for ad-hoc networks because of their partial topology. In fact, it is simple to exhibit graph topologies for which even an adversary corrupting a single node can prevent the design of a secure 1-node discovery scheme.

Topology assumptions. The possibility and performance of protocols over ad-hoc

networks crucially depends on the topology assumptions on the network graph. Proving properties of a protocol under stronger topology assumptions makes such property less likely to hold in real-life ad-hoc networks, even more so in the presence of node mobility. We are therefore especially interested in *minimizing the topology assumptions* under which node discovery protocols are proved correct and secure.

Efficiency measures. We will also be particularly interested in the following two efficiency measures: the number of *initiator requests*, defined as the number of session establishment attempts made by the initiator N on input 1^t; the number of *initiator rounds*, defined as the number of communication rounds between the initiator N and nodes in the rest of the network. Standard efficiency metrics for protocols over ad-hoc networks, such as latency, bandwidth, battery power, energy consumption, etc., are all directly affected by the two defined efficiency measures.

3 Discovering Nodes over Ad-Hoc Networks

In this section we present node discovery protocols over ad-hoc networks, that remain secure in the presence of a Byzantine adversary that can corrupt at any time during the exection of the protocol up to a limited number τ of network nodes. We present 3 protocols under appropriate topology assumptions, the main design goals being (1) minimizing the topology assumption, and (2) minimizing efficiency metrics, both being defined in Section 2. Our most interesting protocol is a τ-secure t-node discovery protocol, for $t = 2\tau + 1$, which, under the $(\tau, 2\tau + 1)$-reachability assumption, only requires $O(\ell)$ initiator rounds and $O(\tau \cdot \ell)$ initiator requests, for $\ell = \log \tau$.

Common setup in all protocols. We assume that access to the ad-hoc network is granted by an *off-line* trusted authority, who also acts as a certification authority which gives to each user (who has not registered yet) a certificate that matches the user's ID and signature verification key. We note that this assumption is quite realistic in military ad-hoc networks (the main setting motivating this work). During a node discovery protocol, each party will sign her messages using the signature verification key certified by the trusted authority, and attach to them the certificate obtained when joining the network. Some properties implied by the above setting, which we will use in the proof of all protocols, are as follows. First, since the users' IDs are bound to their signature verification keys by the authority's certificates, there is no gain in an adversary to replicate a corrupted node, since all replicas will at most establish multiple sessions with the same ID, and therefore won't alter the total number of nodes discovered by an initiator node (thus, all our protocols are secure against "identity-replication attacks"). Secondly, attempts from an adversary to register with different IDs are fruitless, as the trusted authority is required to check the veridicity of IDs, and will therefore not issue two certificates to the same party claiming to have different ID's (thus, all our protocols are secure against "Sybil attacks"). Finally, we note that since all session establishment messages are signed and authenticated, any changes to their content or to the original sender's ID are detected while they are forwarded over multiple hops on the ad-hoc network.

3.1 A protocol under the $(2\tau+1)$-neighbor assumption

As a warmup, we consider the following protocol $\mathrm{NDP}_1 = (\Pi_{pre}, \Pi_{dis})$. Subprotocol Π_{pre} has no special definitional requirement, and subprotocol Π_{dis} is defined as follows. The initiator N, in input 1^t, for $t = 2\tau+1$, asks to establish a session with any t of its neighbors $N_1, \ldots, N_u$. At the end of the protocol, N returns the ID's of all nodes among $N_1, \ldots, N_u$, with which the session establishment protocol was successful.

The analysis is trivial: by the $(2\tau+1)$-neighbor assumption, the initiator N has $u \geq 2\tau+1$ neighbors and the initiator can ask t of them to establish a session; since the adversary can corrupt at most τ neighbors, the session establishment protocol is successful with at least $t-\tau$ of them.

Thus NDP_1 is a τ-secure t-node discovery protocol, for $t = 2\tau+1$, with $O(1)$ initiator rounds and $O(\tau)$ initiator requests, under the $(2\tau+1)$-neighbor assumption.

3.2 Two protocols under the $(\tau, 2\tau+1)$-reachability assumption

We would like to reduce the topology assumption associated with protocol NDP_1, and consider protocols that work for wider families of network graphs. We note that the $(\tau+1)$-neighbor assumption is necessary to obtain secure node discovery protocols against adversary that corrupt up to τ nodes, or otherwise there exists a node N such that all its neighbors can be corrupted by the adversary and N, when acting as an initiator, may not be allowed by the adversary to establish a session with any other node in the network graph. As a consequence, we now consider the problem of finding node discovery protocols under assumptions weaker than the $(2\tau+1)$-neighbor assumption, but not weaker than the $(\tau+1)$-neighbor assumption. Specifically, we will focus on the $(\tau, 2\tau+1)$-reachability assumption. We show two protocols secure under this assumption: the first one with $O(1)$ initiator rounds and $O(\tau^2)$ initiator requests, and the second having $O(\log \tau)$ initiator rounds and $O(\tau \log \tau)$ initiator requests.

A protocol with satisfactory initiator rounds. We now consider the following protocol $\mathrm{NDP}_2 = (\Pi_{pre}, \Pi_{dis})$. It has a satisfactory number of initiator rounds, but a large number of initiator requests.

Protocol description. Subprotocol Π_{pre} has no special definitional requirement, and subprotocol Π_{dis} is defined as follows. We consider an initiator N that takes as input 1^t, for $t = 2\tau+1$, and has u neighbors $N_1, \ldots, N_u$, and we set $u' = \min(u, t)$. For $i = 1, \ldots, u'$, N asks neighbor N_i to forward N's session establishment request to t distinct nodes that are reachable from N_i (including N_i but not including N), and to forward to N such nodes' session establishment replies. At the end of this process, N returns the ID's of at least $t-\tau$ distinct nodes reachable from any among $N_1, \ldots, N_{u'}$, with which the session establishment was successful.

Protocol analysis. The correctness of NDP_2 is simply seen to hold; in particular, we consider simple query-reply interactions to be sufficient to establish a session between any two communicating parties, and then by forwarding query-reply inter-

actions along the graph, it is possible to have N establish a session with any of the t nodes that are reachable from N.

For the security, we first note that for any τ nodes corrupted at any time by the adversary among the neighbors $N_1, \ldots, N_{u'}$ of the initiator N, by the $(\tau, 2\tau+1)$-reachability assumption, there exist $2\tau+1$ nodes that are reachable from N (even if τ corrupted nodes and their adjacent edges are disconnected from the network graph). Thus, $t = 2\tau+1$ nodes will receive the session establishment request from N. Since we assume that query-reply interactions cannot be modified by intermediate parties, it is possible to have N establish a session with any of the $2\tau+1$ nodes that are reachable from N. As a consequence, the protocol returns at least $t - \tau$ IDs of nodes with which N successfully established a session.

We note that with respect to NDP_1, protocol NDP_2 reduces the topology assumption, but significantly increases the number of initiator requests. In the worst case, NDP_2 makes $O(\tau^2)$ session establishment attempts, while NDP_1 only makes $O(\tau)$ of them.

A protocol with satisfactory initiator rounds and requests. We now consider the following protocol $\mathrm{NDP}_3 = (\Pi_{pre}, \Pi_{dis})$. In addition to reducing the topology assumptions, it has satisfactory performance both in the number of initiator rounds and in the number of initiator requests.

Protocol description. Subprotocol Π_{pre} has no special definitional requirement, and subprotocol Π_{dis} is defined as follows. We consider an initiator N that takes as input 1^t, for $t = 2\tau+1$, and has u neighbors $N_1, \ldots, N_u$ and we set $u' = \min(u, t)$. The protocol is divided into phases; at each phases N obtains a number of identities of new nodes with which the session establishment protocol was successful (this number being dependent on the adversary's behavior); finally, the last phase is defined as the phase where the number of all sessions successfully established by N becomes at least $t - \tau$.

In the first phase, for $i = 1, \ldots, u'$, in parallel N asks neighbor N_i to forward N's session establishment request to $t_1 = 2$ distinct nodes that are reachable from N_i (including N_i but not including N), and to forward to N such nodes' session establishment replies. At the end of this process, N partitions the set of neighbors $NS = \{N_1, \ldots, N_{u'}\}$ into two sets $NS_{y,1}$ and $NS_{n,1}$, where $NS_{y,1}$ contains the set of neighbors in NS that forwarded to N identities of 2 distinct nodes with which the session establishment with N was successful, and $NS_{n,1}$ is defined as $NS \setminus NS_{n,1}$. In this first phase, N has received $|NS_{y,1}|$ distinct identities.

In the second phases N sets $t_2 = \lceil 2 \cdot (t - t_1 \cdot |NS_{y,1}|)/|NS_{y,1}| \rceil$, and the same process is repeated with N asking each neighbor N_i in $NS_{y,1}$ to forward N's session establishment request to t_2 distinct nodes that are reachable from N_i (including N_i but not including N), and to forward to N such nodes' session establishment replies. Again, N partitions $NS_{y,1}$ into two sets $NS_{y,2}$ and $NS_{n,2}$, defined analogously to $NS_{y,1}$ and $NS_{n,1}$; specifically, $NS_{y,2}$ contains the set of neighbors in $NS_{y,1}$ that forwarded to N identities of t_2 distinct nodes with which the session establishment with N was successful, and $NS_{n,2}$ is defined as $NS_{y,1} \setminus NS_{n,2}$.

Later, in the third phase N continues sending its queries to set $NS_{y,2}$ of neighbors and by setting $t_3 = \lceil 2 \cdot (t - t_1 \cdot |NS_{y,1}| - t_2 \cdot |NS_{y,2}|)/|NS_{y,2}| \rceil$. The process continues for j_{max} phases, until N has received at least t distinct identities of nodes that received a session establishment request and at least $t - \tau$ distinct identities of nodes with which the session establishment protocol was successfull. (This implies that j_{max} satisfies $t_{j_{max}+1} \leq 0$.)

Protocol analysis. We now analyze the correctness, security and performance properties of protocol NDP_3. First of all, we note that by j_{max} we have denoted the number of phases of protocol NDP_3. It is immediate to see that j_{max} is also an upper bound on the number of initiator rounds, and that $j_{max} \cdot t$ is an upper bound on the number of initiator requests. We would like therefore to compute an upper bound on the value of j_{max}, for each adversary's behavior.

Recall that by t_j we denote the number of identities that N tries to obtain during phase j. Also, we denote as $Idnum_j$ the number of identities that N still would like to obtain at the beginning of phase j (i.e., the max between 0 and the value equal to t minus the number of identities obtained so far). We note that $Idnum_1 = t$, $Idnum_{j_{max}} = 0$ and $t_1 = t$. It is also immediate to see that $t_j \geq Idnum_j$ in the j-th phase, for each j. This guarantees that the correctness properties of NDP_3 is satisfied.

We also note that at each phase the adversary can decide to corrupt some new parties that were uncorrupted in the previous phases. (The only restriction on the adversary being that the number of corrupted nodes is at most τ.) More precisely, the adversary can schedule its corrupted nodes over all phases to either prevent N to successfully establish sessions with t distinct nodes, or to maximize the number of phases. By the $(\tau, 2\tau + 1)$-reachability assumption, there exist t nodes that are reachable from N, for any such schedule from the adversary. Therefore, protocol NDP_3 eventually will allow such nodes to establish a session with N, regardless of the adversary's strategy. This proves the security of NDP_3.

In what follows, we complete the analysis of the performance of NDP_3 by computing an upper bound on j_{max}. We say that a neighbor N_i of N is *corrupted at phase* j if there exists a phase index $j' \leq j$ such that in phase j' the number of identities forwarded from node N_i to node N is smaller than $t_{j'}$. We then denote as $Corrnod_j$ the number of nodes that have been corrupted by the adversary by the end of phase j, and we define $NotYetCor_j$ as $\tau - Corrnod_j$; that is, $NotYetCor_j$ represents the number of nodes that may still be corrupted by the adversary after the end of phase j. Since we consider simple query-reply interactions to be sufficient to establish a session between any two communicating parties, then by forwarding query-reply interactions along the graph, it is possible to have N establish a session with any of the t nodes that are reachable from N. As a consequence, the protocol returns at least t distinct ID's of nodes with which N established a session. The crucial observation we make here is that at the j-th phase either the adversary corrupts at least half of the neighbors of N in set $NS_{y,j}$, or N is able to successfully

establish sessions with at least $t_j/2$ new nodes. This implies that when $j \geq 2$,

$$\text{either } NotYetCor_j \leq NotYetCor_{j-1}/2 \text{ or } Idnum_j \leq Idnum_{j-1}/2.$$

Since the last phase, indexed as j_{max}, happens when $Idnum_{j_{max}} = 0$, by the latter observation, and by recalling that $NotYetCor_1 = \tau$ and $Idnum_1 = t$, we obtain that $j_{max} \leq \log \tau + 2$.

We note that protocol NDP_3 is based on a topology assumption weaker than protocol NDP_1. We then note that with respect to NDP_2, protocol NDP_3 only moderately increases the number of initiator rounds (i.e., from $O(1)$ to $O(\log \tau)$), but it significantly reduces the number of initiator requests (i.e., from $O(\tau^2)$ to $O(\tau \log \tau)$).

4 Applications: Secure client-server Architectures

One interesting research problem about distributed networks is the design and analysis of client-server architectures. This problem has been studied in several specific variants in different research areas, including distributed systems, peer-to-peer networks, etc. In this paper, for sake of generality, we consider the following, very abstract, version of it: in an ad-hoc network, we would like to realize an architecture, or a set of protocols, that allow users to provide and receive services of different kind; any node in the network may or may not be capable of providing some or all desired services (out of a given set of them), and we would like any node in the network to be able, at any time, to request a given service from other parties in the network; this involves finding the nodes in the network that provide the given service and obtaining it from them. The process of requesting and providing services is abstracted as a simple query-reply interaction between any two parties.

There are several security problems in such client-server architectures. In the case of the strongest (known) attacks, Byzantine attacks, we consider an adversary that can corrupt (a limited number of) parties and arbitrarily modify their behavior. In the case of ad-hoc networks, this power is further increased in that the adversary can even choose to move the corrupted party and consequently modify the network topology, or duplicate the party's computing equipment, thus effectively creating new nodes (with the same ID). In the context of protocols within client-server architectures, an adversary corrupting a node may disrupt the execution of the protocol in several ways: it may prevent a service to be routed to the request initiator from other nodes (say, by refusing to provide correct routing of a message from the server to the initiator), it may refuse to respond to an initiator's request, or it may even respond, but still provide an incorrect service.

In the next two subsections we outline two proposals for secure client-server architectures, both based on generic secure node discovery protocols. The first one is based on asymmetric cryptography, and is related to an architecture proposed in [7], which however addressed a more specific client-server architecture problem (our architecture uses any generic secure node discovery protocols, and can be used to improve the architecture for the specific problem from [7]). The second one is new

and is uniquely based on symmetric cryptography, thus resulting in a much more efficient architecture. In both solutions we assume for simplicity that all nodes can act as a server and provide the service requested by the specific request initiator; the extension to the case where only some nodes are capable of providing the requested service can be easily dealt with and is omitted for lack of space.

A Secure client-server architecture using asymmetric cryptography. This architecture uses the following tools:

(i) A secure node discovery protocol, such as those from Section 3, where any initiator node can discover $t = 2\tau + 1$ nodes, where τ is an upper bound on the number of parties ever corrupted by the adversary;

(ii) A verifiably-ID-binding signature scheme [6], based on statistically-unique and cryptographic veriviable identifiers [10,9], allowing any node to sign its messages using a signature that is unforgeably linked to the node's ID;

(iii) A threshold signature scheme over ad-hoc networks [5,6], allowing any node to collect $2\tau + 1$ partial signatures of the same message that can be combined into a single signature, even if τ signatures were generated by an adversary.

The above tools are combined in a secure client-server architecture as follows. The service request initiator starts a secure node discovery protocols, where it tries to discover $2\tau + 1$ servers and forwards to them its service request. The discovered servers compute the service reply message and sign it twice: first, using the partial signature of the threshold signature scheme, to guarantee that the service reply cannot be modified by intermediate nodes routing the answer to the initiator; and second, using the verifiably-ID-binding signature scheme, to guarantee the matching between the service and the server's ID, which guarantees that an adversary corrupting up to τ nodes at most corrupts a minority of the nodes providing the service.

Attractive properties of this architecture include: security against adaptive adversaries (if this type of security is enjoyed by the threshold signatures, as in [6], and thanks to the fact that the secure node discovery protocols from Section 3 also enjoy this type of security); no need for a certification authority (effectively, a public-key infrastructure is not needed, thanks to the properties of verifiably-ID-binding signature schemes); weakened topology assumptions (thanks to the analogue property of the secure node discovery protocols from Section 3); service non-repudiation and short node's private key.

A Secure client-server architecture using symmetric cryptography. This architecture uses the following tools:

(i) A secure node discovery protocol, such as those from Section 3, where any initiator node can discover $t = 2\tau + 1$ nodes, where τ is an upper bound on the number of parties ever corrupted by the adversary;

(ii) A server-based 2-conference key-distribution scheme [1], allowing an off-line authority to distribute private keys to all parties entering the network, so that any 2 parties can compute a shared private key, only given their identities,

which looks random to all other parties.

(iii) A message authentication scheme (MAC).

The above tools are combined in a secure client-server architecture as follows. The service request initiator starts a secure node discovery protocols, where it tries to discover $2\tau + 1$ servers and forwards to them its service request. During this protocol, requests and reply sent between two nodes N_1 and N_2 are authenticated using the MAC algorithm and the key shared by N_1 and N_2, where the responsibility of exchanging the identities of these two parties (whenever they are not neighbors) is of any intermediate nodes. The discovered servers compute the service reply message and authenticate it using the MAC algorithm, where the key used is the key shared by the request initiator and the specific server. This serves both as a way to link the reply to the ID of the server computing it, and as a partial MAC being a component of a threshold MAC, since the initiator receiving several MACs can implement a threshold verification of them. Furthermore, such messages are MAC-ed again by the intermediate routing nodes, by using the MAC algorithm with a key shared by the specific routing node and the request initiator.

Attractive properties of this architecture include: security against adaptive adversaries (if this type of security is enjoyed by the threshold signatures, as in [6], and thanks to the fact that the secure node discovery protocols from Section 3 also enjoy this type of security); weakened topology assumptions (thanks to the analogue property of the secure node discovery protocols from Section 3); and greater time-efficiency, due to the much higher efficiency of symmetric cryptographic primitives.

References

[1] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, *Perfectly Secure Key Distribution for Dynamic Conferences*, in Proc. of CRYPTO '92, LNCS, Springer-Verlag, 1993.

[2] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh, *Bootstrapping Security Associations for Routing in MANETs*, in Institute for Systems Research, ISR Technical Report 2002-44, May 2002

[3] Y. Desmedt and Y. Frankel, *Threshold Cryptosystems*, in Proc. of CRYPTO '89, LNCS, Springer-Verlag, 1990.

[4] Y. Desmedt, *Threshold Cryptography*, in European Trans. of Telecommunications, vol. 5, n. 4, 1994.

[5] G. Di Crescenzo, R. Ge and G. Arce, *Threshold Cryptography over Mobile Ad-Hoc Networks*, in Proc. of SCN '04, LNCS, Springer-Verlag, 2004.

[6] G. Di Crescenzo, R. Ge and G. Arce, *Improved Topology Assumptions for Threshold Cryptography over Mobile Ad-Hoc Networks*, in Proc. of 2005 ACM Workshop on Security for Ad-hoc and Sensor Networks, ACM Press, 2005.

[7] G. Di Crescenzo, R. Ge and G. Arce, *Securing rSerPool against Byzantine Adversaries*, in IEEE Journal on Selected Areas of Communication, 2006.

[8] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, *Secure Distributed Key Generation for Discrete Log Based Cryptosystems*, in Proc. of Eurocrypt 99, LNCS, Springer-Verlag, 1999.

[9] G. Montenegro and C. Castelluccia, *Statistically unique and Cryptographically Verifiable (SUCV) Identifier and Addresses*, in Proc. of 2002 NDSS conference.

[10] G. O'Shea and M. Roe, *Child-Proof Authentication for MIPv6*, in ACM Computer Communication Review, April 2001.

[11] T. Pedersen, *A Threshold Cryptosystem without a Trusted Party*, in Proc. of Eurocrypt '91, LNCS, Springer-Verlag, 1991.

[12] L. Zhou and Z. J. Haas. *Securing Ad-Hoc Networks*, in IEEE Network Magazine, vol. 13, no.6, 1999.