Securing AODV: The A-SAODV Secure Routing Prototype

Davide Cerri and Alessandro Ghioni
CEFRIEL – Politecnico di Milano

***Abstract*—Mobile ad hoc networks pose new kinds of security problems, caused by their nature of collaborative and open systems and by limited availability of resources. In this article we consider a Wi-Fi connectivity data link layer as a basis, and focus on routing security. We discuss our implementation of the Secure AODV protocol extension, which includes some tuning strategies aimed at improving its performance. Namely, we propose an adaptive mechanism that tunes SAODV behaviour. Moreover, we analyse our adaptive strategy and another technique that delays the verification of digital signatures. This paper sums up the experience we collected in the prototype design, implementation and tuning.**

I. Introduction

WIRELESS networks are intrinsically more vulnerable than traditional wired network, because the radio coverage cannot be accurately limited. The defence of wireless managed networks can benefit from the existence of a related authority that can rule and enforce security, having control on network equipment. From a security perspective, *mobile ad hoc networks* inherit the same problems that exist in managed wireless networks (e.g., it is easy to access the data-link layer and to intercept transmitted data, and it is easy to blind the network with denial of service attacks at the physical layer). However, MANETs additionally pose new kinds of security problems, because their nature of collaborative and open systems can be exploited by malicious entities.

MANETs can be cheaply and easily realized by simply making use of Wi-Fi network cards configured in ad hoc mode and of a layer three routing protocol designed for handling the variability of the network topology. Pragmatically, we consider cheap Wi-Fi connectivity as a basis, we focus on routing at a non-geographical scale and thus, following the classification proposed in [1], we opt for a flat network structure.

Many routing protocols for mobile ad hoc networks have been proposed. Up to now, the IETF manet working group produced four experimental RFCs that specify as many flat routing protocols: AODV (Ad-hoc On Demand Distance Vector), OLSR (Optimized Link State Routing), TBRPF (Topology Dissemination Based on Reverse-Path Forwarding) and DSR (Dynamic Source Routing). Another routing protocol, DYMO (Dynamic MANET On-demand), is currently in draft state. However, none of these protocols specifies any security measure, effectively assuming that there are no malicious nodes participating in routing operations. It is worth noting that in an open network that is based on collaboration between nodes, like a MANET, in order to have a reliable infrastructure security issues cannot be overlooked.

Many different proposals can be found in literature aiming at adding security functionalities to existing routing protocols, or proposing new secure routing protocols [2]. Two different kinds of techniques can be identified in these proposals: techniques aiming at guaranteeing authenticity and integrity of routing messages, and techniques that let nodes monitor the behaviour of other nodes in routing operations. Both of these directions imply that some resources (bandwidth, processing power, battery power) need to be spent by nodes for protecting messages or monitoring the network, respectively. As is often the case, the main issue is finding the trade-off between security and performance, taking into account the constraints in terms of limited resources that the nodes participating in a MANET have.

This article describes our experience while realizing a prototype for MANET secure routing and about the related performance analyses and tuning. First, we briefly describe the AODV protocol, the attacks to which it is subject, and a well known security extension proposal. Then, we present our prototype implementation and some tuning strategies.

II. AODV

AODV [3], [4] is perhaps the most well-known routing protocol for MANETs. It is a *reactive* protocol: nodes in the network exchange routing information only when a communication needs to take place, and keep this information up-to-date only as long as the communication lasts.

When a node must send a packet to another node, it starts a *route discovery* process in order to establish a route towards the destination node. Therefore, it sends its neighbours a route request message (RREQ). Neighbouring nodes receive the request, increment the hop count, and forward the message to their neighbours, so that RREQs are actually broadcasted using a *flooding* approach. The goal of the RREQ message is to find the destination node, but it also has the side effect to make other nodes learn a route towards the source node (the "reverse route"): a node that has received a RREQ message with source address S from its neighbour A knows that it can reach S through A, and records this information in its routing table along with the hop count (i.e., its distance from node S following that route). The RREQ message will eventually reach the destination node, which will react with a route reply message (RREP). The RREP is sent as a unicast, using the path towards the source node established by the RREQ. Similarly to what happens with RREQs, the RREP message allows intermediate nodes to learn a route towards the destination node (i.e., the originator of the RREP). Therefore, at the end

of the route discovery process, packets can be delivered from the source to the destination node and vice versa. A third kind of routing message, called route error (RERR), allows nodes to notify errors, e.g. because a previous neighbour has moved and is no longer reachable. If the route is not active (i.e., there is no data traffic flowing through it), all routing information expires after a timeout and is removed from the routing table.

AODV is a "collaborative" protocol, and allows nodes to share the information they have about other nodes. During the route discovery process, RREQ messages need not necessarily reach the destination node: if an intermediate node already knows a route towards the destination it generates a RREP message and does not forward the RREQ any further. This allows quicker replies and limits the flooding of RREQs when it is not needed.

AODV uses *sequence numbers* in order to identify fresher routing information. Each node maintains its own sequence number, incrementing it before sending a new RREQ or RREP message. These sequence numbers are included in routing messages and recorded in routing tables. AODV always favours newer information, thus nodes update their routing table if they receive a message with a sequence number higher than the last recorded one for that destination. Route length (given by hop count) is a less important criterion: newer information (i.e., carried by messages with a higher sequence number) is always preferred, even if it identifies a route that is longer than the previous one.

Being a distance vector routing protocol, AODV does not give nodes a complete view of network topology: each node knows its neighbours, and, for non-neighbouring nodes, it knows only the next hop to reach them and the distance (in hops).

III. Attacking AODV

AODV does not take security into account: AODV messages are neither encrypted nor authenticated nor integrity protected, and basically are always assumed as trusted.

Many kinds of attacks are possible, based on the possibility to forge packets and on the distributed and uncontrolled nature of the network. A malicious node could impersonate a source node by creating fake RREQ messages with its victim's address as originator and by using a sequence number higher than its victim's (similarly, the attacker can impersonate a destination node by creating fake RREPs). The attacker can also generate false error messages, spreading fake information in the network, e.g. in order to announce that a certain destination is not reachable any more. This kind of false information could be spread around as the first stage of a more complex attack, aiming at excluding a target node from the network before some fake RREQs or RREPs are sent to other nodes. Spoofed RREQ and RREP messages can be used in order to redirect some traffic through alternative routes, create loops in the network, segment the network, or perpetrate a denial of service attack. A systematic analysis of AODV security is proposed in [5], where misuse goals that an inside attacker may want to achieve are analysed and classified. The following categories are identified:

- Route Disruption: this means either breaking down an existing route or preventing a new route from being established;
- Route Invasion: this means that an inside attacker can add itself to a route between two endpoints of a communication channel;
- Node Isolation: this refers to preventing a given node from communicating with any other node in the network. It differs from route disruption in that node isolation is aimed at all possible routes, instead of targeting at two specific endpoints;
- Resource Consumption: this refers to consuming the communication bandwidth in the network or storage space at individual nodes.

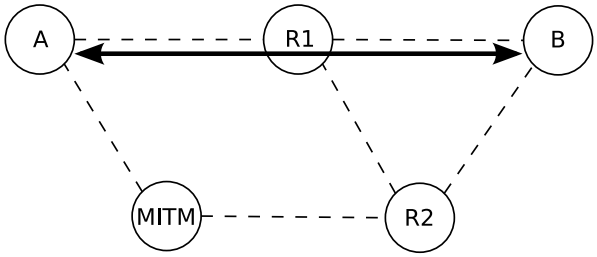
Moreover, AODV misuses can be atomic (i.e., a single routing message is manipulated) or compound (composed of multiple atomic misuses, possibly along with normal uses of the routing protocol).

As an example, we developed a simple man-in-the-middle (MITM) attack tool that redirects victims' traffic through the attacking node, so that the attacker can put himself in the middle of a communication (following the aforementioned classification, this is a route invasion compound misuse). The attack is shown in figure 1. Dashed lines represent direct radio visibility, solid lines represent traffic, arrows with text represent forged routing messages. Before the attack begins, node A sends a RREQ for node B, and B replies. Thus, a route is established between nodes A and B, with traffic flowing through intermediate node R1. As the malicious node (MITM) wants to begin its attack, it first needs to establish its own routes towards nodes A and B using legitimate RREQ messages. Once these routes are established, it can periodically send fake RREP messages to nodes A and B. Fake RREPs towards node A have the destination field set to B's address, so that A believes that they were generated by B and thus that a route towards B is available through the MITM node. Similarly, the attacker sends node B (through node R2[1]) fake RREPs with node A address as destination. R2 thus records a new route to A with MITM as next hop, and B records a new route to A with R2 as next hop. The attacker can therefore poison other nodes' routing tables with false information, and hijack traffic. To be successful, the attacker must put high enough sequence numbers in its forged RREPs, so that other nodes (A, B, R2) consider these as newer and update their routing tables, forwarding traffic through the new route. The malicious node can then eavesdrop, modify or drop the traffic.

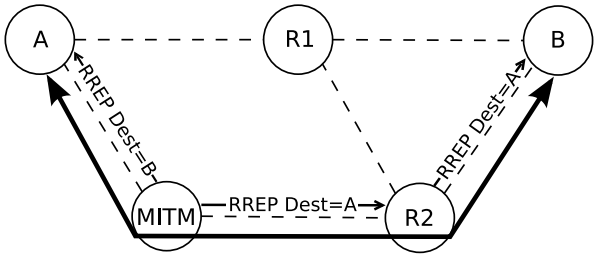
IV. SAODV

SAODV (Secure AODV) [6] is a security extension of the AODV protocol, based on public key cryptography. SAODV routing messages (RREQs, RREPs, and RERRs) are digitally signed, in order to guarantee their integrity and authenticity. Therefore, a node that generates a routing message signs it

[1]Node R2 is not colluding with the attacker. It has a route to B (even if B had not been a neighbour, this route would have been established by the attacker in the previous phase), so it simply forwards the RREP along that route.



(a) Before attack



(b) During attack

Fig. 1. AODV MITM attack by routing table poisoning.

with its private key, and the nodes that receive this message verify the signature using the sender's public key. The hop count cannot be signed by the sender, because it must be incremented at every hop. Therefore, in order to protect it (i.e., not allow malicious intermediate nodes to decrement it), a mechanism based on hash chains is used.

In its basic form, this makes it impossible for intermediate nodes to reply to RREQs if they have a route towards the destination, because the RREP message must be signed by the destination node. In order to preserve the collaboration mechanism of AODV, SAODV includes a sort of delegation feature that allows intermediate nodes to reply to RREQ messages. This is called the *double signature*: when a node A generates a RREQ message, in addition to the regular signature it can include a second signature, which is computed on a fictitious RREP message towards A itself. Intermediate nodes can store this second signature in their routing table, along with other routing information related to node A. If one of these nodes then receives a RREQ towards node A, it can reply on behalf of A with a RREP message, similarly to what happens with regular AODV. In order to do so, the intermediate node generates the RREP message, includes node A's signature it previously cached, and signs the message with its own private key.

SAODV does not require additional messages with respect to AODV. Nevertheless, SAODV messages are significantly bigger, mostly because of digital signatures. Moreover, SAODV requires heavyweight asymmetric cryptographic operations: every time a node generates a routing message it must generate a signature, and every time it receives a routing message (also as intermediate node) it must verify a signature. This gets worse when the double signature mechanism is used, since this may require the generation or verification of two signatures for a single message. In our prototype, described below, we try to mitigate this effect.

V. Our prototype: A-SAODV

In order to test MANET secure routing in real world scenarios we developed *A-SAODV*, a prototype implementation of SAODV based on the AODV-UU implementation by Uppsala University[2]. Unlike AODV-UU, A-SAODV is a multithreaded application: cryptographic operations are performed by a dedicated thread in order to avoid blocking processing of other messages. Therefore, in A-SAODV there are two execution threads: one dedicated to cryptographic operations, and the other to all other functions (routing message processing, SAODV routing table management, timeout management, SAODV message generation, and data packet forwarding). The two threads communicate via a FIFO queue containing all the messages that need to be signed or verified. A-SAODV runs on Linux, and complete source code can be downloaded from our web site[3].

Our prototype includes an experimental feature, the *adaptive reply decision*, which is the reason why it is called A-SAODV (i.e., Adaptive SAODV). This feature is meant to optimize SAODV performance with respect to the double signature option. In AODV, allowing intermediate nodes to generate RREPs on behalf of the destination node has a positive impact on performance, since it does not require heavyweight operations by intermediate nodes themselves. The situation is different in SAODV, because generating such a reply requires the intermediate node to generate a cryptographic signature: nodes may spend much time in computing these signatures, and become overloaded. Moreover, if intermediate nodes have a long queue of routing messages that must be cryptographically processed, the resulting delay may be longer than if the request reaches the destination node. If we remove the double signature mechanism we have an "uncollaborative" protocol, in which only the destination node is allowed to reply to a RREQ message. This is possible, but our simulation results show that if signing time is low, and routes are not very short, performance is worse than SAODV with double signatures. Therefore, our proposal consists in making the double signature feature adaptive: intermediate nodes reply to RREQs only if they are not overloaded.

Each node has a queue of routing messages to be signed or verified, and the length of this queue (with different weights for signature operations and verification operations[4]) can be used to evaluate the current load state of the routing daemon. When a node receives a RREQ message and has the information to generate a RREP on behalf of the destination, it checks the queue length and compares it with a threshold. If the queue length is lower than the threshold the node generates a RREP (collaborative behaviour), otherwise it forwards the RREQ without replying (uncollaborative behaviour). The same mechanism can be applied when generating a RREQ message, in order to decide between a single signature and a double signature. In the simplest case the threshold can be a fixed value; however this would not be very flexible because we

[2] AODV-UU is available at http://core.it.uu.se/core/index.php/AODV-UU

[3] A-SAODV is available at http://saodv.cefriel.it/

[4] A-SAODV currently supports RSA only, and in RSA verification operations are significantly more lightweight than signature operations.

may want to adjust this value depending on external factors (e.g., battery state). In our prototype the threshold value can be changed during execution (two special values allow to have "always reply" behaviour and "never reply" behaviour). Other, more elaborate strategies could be defined in order to estimate the crypto queue delay and consequently decide whether to reply or forward the message. For example, a fixed threshold (based on the timeouts defined by the routing protocol) and a predictor of queuing times could be used. In this way, the algorithm could adapt itself to the situation and the computing power of the node. An additional external parameter could be used in order to take into account the previously mentioned external factors (how much a node is willing to collaborate, e.g. depending on its battery state).

Another little optimization included in our prototype is a cache of latest signed and verified messages, in order to avoid signing or verifying the same message twice.

For key management purposes our implementation relies on a simple keyring stored in a file, but we also developed a minimal graphical key manager that can be used with A-SAODV. This key manager must be used in a bootstrap phase, with all nodes visible within one hop. This is necessary because routing operations cannot be performed before distributing keys (as signatures would not be verified), therefore we must exchange keys without message routing. At this time, each user sends his/her key to all the others using local broadcast (i.e., 255.255.255.255), then reads the key fingerprint aloud. Other users check if the fingerprint that is being read matches the one they received: if so, they add the received key to their keyring. This could be extended with a web-of-trust (PGP-like) mechanism, in order to remove the requirements of initial one-hop bootstrap and explicit approval of all keys by users.

VI. Optimizing SAODV: simulation tests

The adaptive reply decision strategy described in the previous section is a way to optimize SAODV performance. In order to evaluate its effect in different scenarios with many nodes, we ran simulation tests using ns-2 Network Simulator. If routes are very short, the "uncollaborative" strategy appears to be the best one, because the gain given by intermediate nodes replies is not worth the cost. On the other hand, we found that the adaptive strategy is useful when routes are quite long, because in this case routing performance benefits from replies by intermediate nodes.

First, we considered a square scenario of 200 $\times$ 200 m with 50 nodes and a 50 m connectivity radius. Nodes move following a random waypoint model with no pause time, and establish 50 random connections. Such a scenario does not show significant deviations between different strategies (with a slight prevalence of the uncollaborative one), because routes are very short (between 2 and 3 hops on the average). In order to obtain longer routes and test the protocol under more critical conditions, we moved to a rectangular scenario of 1500 $\times$ 50 m, with 100 nodes that establish 100 random connections. Indeed, the average length of established routes in this scenario results in between 4 and 5.5 hops, depending on signing time and adopted collaboration strategy. Considering this rectangular scenario, figure 2 shows the number of successfully established connections and the first data packet delay (this one normalized with respect to route length), for different signing times, in the case of the three different strategies: uncollaborative (never use double signature in RREQs, intermediate nodes never reply), collaborative (always use double signature in RREQs, intermediate nodes always reply if they have information), and adaptive. In this situation, the adaptive variant behaves generally better than the other two, successfully establishing a higher number of routes. Longer routes have a higher probability of not being successfully established, and this can bias results about the delay of the first data packet. If a variant fails more than another one in establishing long routes, it will establish fewer connections but these connections will correspond to shorter routes, thus having a shorter delay for the first data packet. Since we can measure the delay only on successfully established connections, such a variant would appear to have better performance, but this would be a misleading result. This is the reason why we normalize first data packet delay with respect to route length. With this clarification, we can see that in our scenario the adaptive variant behaves better than the other two, having a shorter delay. Finally, other parameters, such as the number of generated routing packets, do not show significant differences between the three considered strategies.

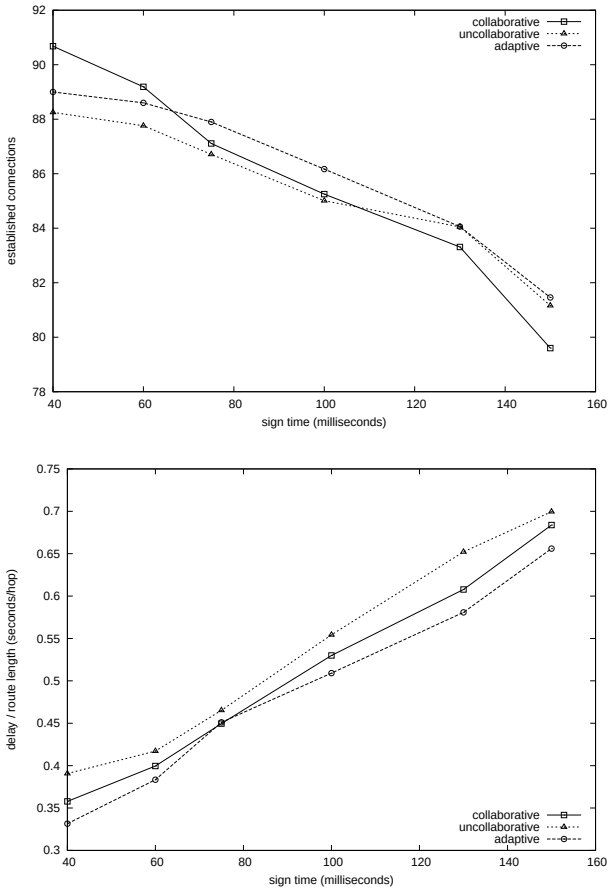


Fig. 2. Simulation results for adaptive reply decision.

In [7], Guerrero Zapata proposes another optimization to improve SAODV performance: the *delayed verification*. With this

optimization, intermediate nodes forward routing messages before verifying their signature. In order to avoid obvious attacks (i.e., accepting fake information), routing information is stored as *unverified*, and is not used before verification. Nevertheless, this strategy allows intermediate nodes to verify signatures in parallel, since routing messages are not blocked at each hop waiting for signature verification. Moreover, intermediate nodes can delay signature verification until that information is needed, and do not verify it at all if it expires without having been needed. This can be quite common with RREQ messages: since they are sent in broadcast many nodes will receive them, but only a fraction of nodes will happen to be on an active route path towards the source node. Our simulation results show that delayed verification improves SAODV performance. Figure 3 shows first data packet delay (normalized with respect to route length) for regular SAODV (normal) and for SAODV with delayed verification, for the same rectangular scenario we described before. Other scenarios (e.g., square) give similar results.

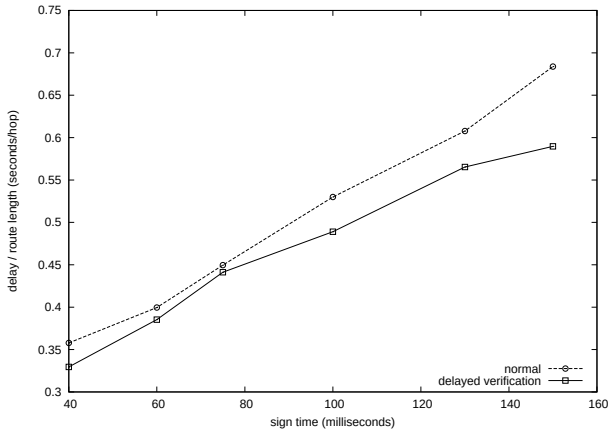


Fig. 3. Simulation results for delayed verification.

Delayed verification is currently not implemented in A-SAODV prototype, also because of a potential problem that in our opinion deserves further investigation. Fundamental security guarantees given by SAODV, i.e. authenticity and integrity of routing information, are not affected by delayed verification, because unverified information is not used for routing. Nevertheless, with delayed verification a malicious node sending RREQs with fake signatures will be able to flood the network, whereas with normal verification neighbouring nodes would recognize the fake signature and would not forward the message. Therefore, delayed verification could open the possibility for more effective (and therefore dangerous) denial of service attacks. It can be argued that, if routing information contained in fake RREQs is not going to be used, with delayed verification fake signatures are not going to be verified, thus not causing much harm (or even causing less harm). However, the attacker can also send fake RREPs or fake data packets with proper source and destination IP addresses in order to trigger signature verifications, at least by neighbouring nodes. An analysis and simulation of this kind of attacks should be performed, in order to test the impact of delayed verification if the network is under attack.

VII. Conclusions

In this article we presented A-SAODV, a prototype implementation of the SAODV routing protocol. SAODV adds security to AODV, but includes cryptographic operations that can have a significant impact on performance. We discussed the adaptive reply decision, an experimental feature we added to our implementation in order to improve SAODV performance. Other possible improvements could be added, e.g. the delayed verification (which seems to have a positive impact on performance), but further investigation is needed. In particular, situations with both “good” and “bad” nodes should be considered in simulation tests, in order to evaluate the behaviour of SAODV and of the proposed optimizations under attack (e.g., denial of service attempt).

Acknowledgements

The authors want to thank Nicola Alagia, Francesco Dolcini and Alex Mufatti for their valuable contributions to this work.

This work was partly supported by the Italian Ministry of Education, University and Research under the MAIS FIRB project.

References

[1] X. Hong, K. Xu, and M. Gerla, “Scalable Routing Protocols for Mobile Ad Hoc Networks,” *IEEE Network*, vol. 16, no. 4, pp. 11–21, Jul/Aug 2002.

[2] P. Argyroudis and D. O’Mahony, “Secure Routing for Mobile Ad hoc Networks,” *IEEE Communications Surveys and Tutorials*, vol. 7, no. 3, pp. 2–21, third quarter 2005.

[3] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, USA, Feb 1999, pp. 90–100.

[4] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing,” RFC 3561, Jul 2003.

[5] P. Ning and K. Sun, “How to Misuse AODV: A Case Study of Insider Attacks against Mobile Ad-Hoc Routing Protocols,” in *Proceedings of the 4th IEEE Information Assurance Workshop*, West Point, NY, USA, Jun 2003, pp. 60–67.

[6] M. Guerrero Zapata and N. Asokan, “Securing Ad hoc Routing Protocols,” in *Proceedings of the 1st ACM workshop on Wireless security*, Atlanta, GA, USA, Sep 2002, pp. 1–10.

[7] M. Guerrero Zapata, “Key management and delayed verification for ad hoc networks,” *Journal of High Speed Networks*, vol. 15, no. 1, pp. 93–109, Jan 2006.

Davide Cerri (d.cerri@ieee.org) obtained his Laurea degree in Telecommunications Engineering at Politecnico di Milano in 2001, with a thesis about the performance of secure communication protocols. Since 2001 he has been working as a researcher at CEFRIEL – Politecnico di Milano, in the Middleware unit, focusing on information and communication security topics. He has been involved in various research projects, both at the national and European levels. His research interests cover security topics in open and distributed systems, in particular mobile ad hoc networks and peer-to-peer overlay networks.

Alessandro Ghioni (ag@ghioni.name) obtained his Laurea degree in Telecommunications Engineering at Politecnico di Milano in 2001, with a thesis about multicast TCP-friendly video distribution. From 2001 to 2007 he worked as a researcher at CEFRIEL, in the Middleware unit, focusing on information security topics. He has been involved in different research projects, both at a national and European level, dealing with security protocols and algorithms in open and distributed environments, such as peer-to-peer and mobile ad hoc networks. Currently, he works as a project manager at Emaze Networks, an Italian company providing services and products in the Information Security field.