This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

Lightweight, distributed access control for wireless sensor networks supporting mobility

Leonardo Maccari*, Lorenzo Mainardi*, Maria Antonietta Marchitti† , Neeli R. Prasad†, Romano Fantacci*
*Department of Electronics and Telecommunications - University of Florence
Telecommunication Network Lab tel. : +390554796467, Florence, Italy
Email: {maccari, mainardi, fantacci}@lart.det.unifi.it
†Center for TeleInFrastruktur (CTIF), Aalborg University
Niels Jernes Vej 12, 9220 Aalborg East,Denmark
Email: {marchit, np}@es.aau.dk

***Abstract*— Wireless sensor networks (WSN) are large scale networks of unattended devices, aimed at monitoring environmental parameters. Their extremely scarce hardware resources constitute a huge limitation to the use of standard security protocols to secure communications, so that custom ones must be designed. In this article we describe the development of a novel access control system for WSN based on a distributed threshold scheme. Our model gives support for mobility and limits the needed communication and consequent energy drain, which is a fundamental parameter for the lifetime of WSN.**

I. Introduction

Wireless sensor networks (WSN) are large networks composed of nodes equipped with minimal hardware capacity and environmental sensors measuring parameters such as temperature, humidity etc. The idea behind WSN is that a large network (from few units to thousands of units) can be used to monitor the environment even in a large area, in which the nodes coordinate themselves in a distributed fashion. A mesh network is formed using distributed MAC and routing protocols and data sensed are generally conveyed to a gateway to be accessed by the manager of the network. WSN are often unattended and battery powered so one of the main issue to take into account when designing suitable protocols is energy efficiency.

In critical applications, such as surveillance or monitoring of medical parameters security services, such as access control and privacy, achieve great importance. Security in a WSN is particularly difficult to obtain, first because it is imagined as a distributed network without a central authority, and secondly for the hardware limitations of nodes, which cannot afford costly crypto functions such as RSA (for an introduction on WSN security see [1]). One more peculiar security issue of WSN is that being them often unattended, it is possible that a node could be stolen and analyzed by an attacker, and the private material (keys) revealed. Since the price of the nodes should be limited as much as possible anti-tampering hardware devices is hardly usable; for these reasons authentication protocols should be resistant to attacks from *insider enemies* at least to some extent.

This work is partially supported by EU Network of Excellence FP6-IST-4-027738-NoE "CRUISE"

An access control protocol is a procedure used to certify that a node is authorized to enter a certain network. Two desirable properties of such a protocol are: mutual authentication, that means that also the newcomer is assured that it is entering the correct network and not a rogue one, and key-establishment, the generation of a fresh shared secret between the node that enters and the node that acts as intermediary with the network. This key will be used to achieve privacy in the following communications through encryption and message authentication codes (MAC). The most simple procedure to perform access control is that both nodes refer to a central authentication server, which behaves as a trusted third party and delivers a shared key to the two nodes after having authenticated the newcomer. This model (as suggested in [2]) is hardly applicable to real WSN for the following reasons:

- there must always be a path to the server
- if the network is composed of thousands of nodes, the path to reach the server could involve tens of nodes, introducing great delays and energy waste.

Other protocols based on key pre-distribution [3] or bivariate polynomials [4] produce only key-establishment between couples of nodes. If no trusted third party is used, each node has no assurance that the other is an authorized node, a malicious node in possession of the right credentials could perform key-establishment with a victim node even without being part of the network, with the consequence of isolating the victim. To have mutual authentication the use of a centralized protocol is still the only solution. Unluckily, a centralized protocol is even less feasible if the WSN supports some form of mobility, in fact a mobile node roaming in a static WSN (it might be a sink collecting information or just a regular node changing position) would have to perform multiple authentications with nodes that it finds in its path. If each one produces a multi-hop exchange with a central server, the overhead would be intolerable in terms of time needed and energy drain.

In this paper we propose a different approach based on threshold cryptography, that consists in delegating the responsibility of access control to a coalition of nodes that allow another node to enter the network. The coalition can be formed

by nodes physically close to the newcomer, so that multi-hop communications is avoided or reduced to the minimum. The authentication is performed at a local level, so that no delay is introduced, energy consumption is minimized and multiple subsequent authentications are possible, supporting mobility. As in any threshold scheme the price to pay is the possibility that a group of malicious nodes could join together and fool the authentication, allowing more malicious nodes to enter the SN. This is possible only if the malicious nodes are already authenticated into the network, so the network is under a distributed attack form internal enemies. We will show how the risk for this attack can be parametrized as wanted by the network manager.

II. Access control with Shamir's secret sharing

While models for distributed authentication that have been proposed for MANET networks [5] are all based on public/private key security schemes, a secret sharing scheme based on polynomial interpolation is described by Shamir in [6]. Basically it permits a number N of actors to share a secret in a way that a subset of at least K of them should join to reconstruct it, we will show how to modify it in order for it to behave as an access control algorithm. In Shamir's scheme the secret is a number (a y value of the polynomial), that could be used as a symmetric key to unlock a larger set of data. It works as follows:

- at start-up a trusted party generates a polynomial $q(x)$ of degree $K-1$ in a way that the secret S is given by $S = q(0)$
- a set of N couples $(x_i, q(x_i))$ is generated and each actor receives one couple (a *share* of the secret)
- if the secret has to be revealed, at least K actors have to join their shares to make interpolation of the original polynomial possible.
- knowledge of a set of less than K shares gives no information about the secret.

Efficient functions for polynomial interpolation are present in literature and can be easily ported to sensor nodes.

A simple access control scheme could be realized as follows: the manager of the network generates a polynomial $q(x)$ of degree K and each node of the network is equipped with one couple $(x_i, q(x_i))$. For simplicity imagine that the network is already bootstrapped and there is a kernel of nodes that already performed mutual authentication, so each of them is in possess of a distinct symmetric key for each of its neighbors and the traffic between nodes is protected from sniffing. Now a new node is added to the network, and the situation depicted in figure 1 is reproduced. N_1 chooses an intermediary, N_2, and sends to it the couple $(x_1, H(q(x_1)))$ where H() is a secure one-way hash function. N_2 contacts its neighbors, which in turn answer with their shares $(x_i, q(x_i))$, if N_2 is able to collect enough shares it is able to interpolate the polynomial and derive $q(x_1)$, then perform the hash and verify that N_1 effectively owns a valid share. The value $q(x_1)$ that has never been transmitted in clear can be used as the symmetrical key for the link between N_1 and N_2. Authentication is mutual, because if N_2 effectively is in possession of $q(x_1)$, then N_1 can be sure that the authentication has involved at least K authorized nodes.

As described, the scheme has several defects, we outline two of them:

- once the intermediary has interpolated the polynomial, it has the power to generate new couples $(x, q(x))$, to redistribute it to more evil nodes and let them enter the network, as well as to derive the secret key $q(x)$ for any other couple of nodes.
- if N_2 has not enough neighbors, authentication can not take place.

The main advantage of this approach is that authentication is performed with a minimal exchange of frames (2K+1), distributed among K distinct nodes in a short time interval.

To avoid the outlined problems we recall the linearity of sum operation over polynomials. For polynomials of degree K-1 stands the following:

$$A(x) = \sum_{i=0}^{K-1} a_i x^i; B(x) = \sum_{i=0}^{K-1} b_i x^i$$

$$A(x) + B(x) = \sum_{i=0}^{K-1} (a_i x^i + b_i x^i) = \sum_{i=0}^{K-1} (a_i + b_i) x^i$$

If r_0 and r_1 are integer constants and we call $C(x) = r_0 A(x) + r_1 B(x)$ and $c_i = (r_0 a_i + r_1 b_i)$ then we have:

$$A_0 := A(x_0); B_0 := B(x_0); C_0 := C(x_0)$$

$$C_0 = r_0 \sum_{i=0}^{K-1} (a_i x_0^i) + r_1 \sum_{i=0}^{K-1} (b_i x_0^i)$$

Once defined $\vec{V_0} = [A(x_0), B(x_0)]^T$ and $\vec{R} = [r_0, r_1]$ then $C_0 = \vec{R}\vec{V_0}$. Summing up, if a polynomial C(x) of degree K is the linear combination of two polynomials (A(x), B(x)) its value in x_0 will be the linear combination of A(x_0), B(x_0). That means that to interpolate C(x) a set of K couples (x_i,$\vec{R}\vec{V_i}$) is needed.

Thi said, our idea is to generalize Shamir's scheme adding three more features:

- a set P of M polynomials of degree K is generated, and each polynomial is sampled at values x_i; each node in the network receives a vector $V_i = [P_0(x_i), P_1(x_i) ... P_{M-1}(x_i)]$
- when the protocol starts, the first two frames are needed to generate a random vector of size M. The coefficients of the vectors are used to generate a new fresh polynomial with linear combination, so no private data is revealed
- to overcome the limit due to the numbers of neighbors each node that receives a request can rebroadcast the request to its neighbors; this procedure is called *delegation*.

A simple scheme for random vector generation is the following:

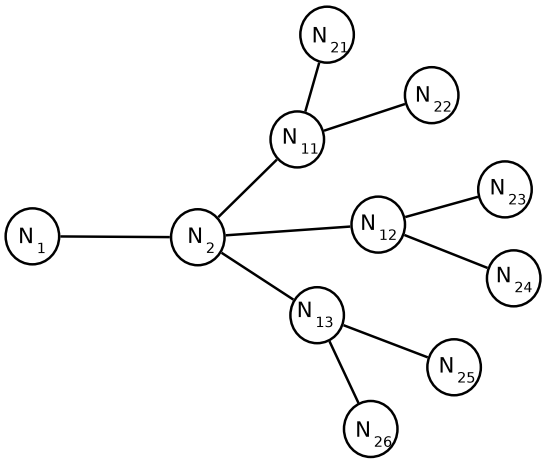

Fig. 1. N_1 is entering the network, N_2 is chosen as intermediary, the others are nodes already authenticated to the network so we assume the communications between them and N_2 is secured.

1) $N_1 \rightarrow N_2$: $[Nonce1]$
2) $N_1 \leftarrow N_2$: $[Nonce2, H(Nonce1)]$
3) both nodes can generate r = H(Nonce1,Nonce2)

From r a vector $R = [r_0, r_1 ... r_{M-1}]$ of arbitrary length can be derived with multiple hashes, discarding zero values.

Upon network entrance, the situation for a new node is depicted in figure 1. Once N_1 enters the network it is in possession of vector V_1, it chooses an intermediary (N_2) and both r and R are generated.

The value r is transmitted to N_2's neighbors, that we call *distributed authentication servers* (dAS), which are able to recreate R. Each dAS computes $v_{ir} = RV_i$ and forwards back the couple (v_{ir}, x_i) to N_2 which in turn is able to compute the whole fresh polynomial C(x), which is the linear combination of the M initial polynomials. N_1 computes v_1, and sends $(x_1, H(v_1))$ to N_2 which can compute $C(x_1)$ and verify the hash that has received from N_1. If the authentication had success the result is that N_1 and N_2 should now be in possession of a fresh secret value $C(x_1)$, but the intermediary has not interpolated any of the initial polynomials but a random linear combination of them. Carefully choosing the size of the integers avoids the reuse of the same linear combination is successive authentications, since R is chosen randomly. From that moment on, N_1 and N_2 can use $C(x_1)$ as a key for encryption and message authentication, thus, N_1 can have a confirmation that N_2 was able to calculate $C(x_1)$.

The problem of the reconstruction of the original polynomials is not completely avoided. An insider attacker could send multiple requests to a certain node X of the network, using distinct seeds r_i, at the end it could invert the matrix composed of $[r_1, r_2, ...r_M]$ and resolve a system that gives v_x as a solution, repeating this procedure with up to K nodes, he could interpolate all the polynomials. To avoid this problem another mean is used, we call it *delegation*, together with *cheating detection* and it also solves the problem of not having enough neighbours to perform authentication.

A. Delegation

We call delegation the possibility of a dAS to forward the request to another node, and forward back the response to the original sender. Using this procedure we resolve two problems, the first one is that if the intermediary can have a larger number of responses than the number of neighbours, the second one is that a node can choose not to answer to a certain request, to avoid revealing its share of the secret. After the generation of R, the protocol continues with the following exchange:

4) $N_2 \rightarrow N_{1x}$: $[r, Depth]$. $Depth$ is a parameter that tells N_2's neighbors wether they have to forward the requests to a second level neighbours. It is greater than 1 if N_2 has less than K neighbors.
5) If $Depth > 1$, each N_{1x} will decrease its value and forward the request to its neighbors N_{2x}. Whenever a node receives a request with Depth = 1, it will not forward the request but will calculate the v_{ir} and answer back. N_{1x} will wait until reception of all the answers (or a specified timeout) then send to N_2 a single unicast packet containing all the shares including its own.
6) When N_2 has received enough shares both N_1 and N_2 should be in possession of $C(x_1)$. The following actions depend on the protocol used afterwards; if the protocol (as advisable) uses some form of message authentication code based on the shared key there is no need for explicit confirmation of success, otherwise two more packets could be used, such as:
7) $N_1 \rightarrow N_2$: $[Nonce3, H(Nonce2, Nonce3, C(x_1))]$
8) $N_1 \leftarrow N_2$: $[Nonce4, H(Nonce3, Nonce4, C(x_1))]$
9) The value $C(x_1)$ is a secret shared between N_1 and N_2 that can be used to generate symmetric keys to secure following communications.

Even when a node receives a request with Depth=1, he can chose not to answer to the request and forward the request to one of its neighbors. If it has no neighbors it can just avoid answering.

Delegation makes the K parameter flexible, to have more shares N_2 could issue a request with $Depth$ larger then needed to have a stronger security, for example if the network size grows and there is a higher probability of having nodes captured.

This procedure can be performed by N_1 with more then one node, so that at the end it could have secure communications with more neighbors.

III. Security considerations

Our protocol achieves bi-directional authentication between a node entering the network and the network itself, guaranteed up to the compromising of K nodes. In the cited key-establishment schemes, if keying material present in one node is stolen, a whole rogue network could be created, made of new nodes all equipped with the same credentials. N_1 could perform key-establishment with the rogue network, and this is possible with the theft or only one node. In our scheme when N_1 performs a successful authentication, it is assured that N_2

is in contact with at least K well behaving nodes, unless at least K nodes have been compromised. As said, K is a lower limit to network security, N_2 could decide to retrieve a higher number of shares, to have a higher security level. To justify the security of our protocol we outline the capabilities of a generic attacker, able to sniff or inject frames into the network.

An *external attacker* can forge the initial nonce, but he will not be able to derive the same value $C(x)$ of the intermediary, so it will not be able to conclude the authentication. The packets exchanged between N_1 and N_2 do not contain information useful for interpolation, so the attacker can only perform a brute force attack on the derived key, which is unavoidable in almost any protocol. If the confirmation frames are not used the attack should be performed on-line, which makes it unfeasible for sufficiently long symmetric keys. The communications between N_2 and the neighbors is ciphered, so it doesn't give information to the attacker, also with delegation.

A single *internal attacker* (i.e. a node that has been stolen and reprogrammed) is able to produce a denial of service on the authentication, answering with a wrong value. A cheating detection technique can be implemented if more than K v_{ir} are collected by the intermediary. If (K+1) values are collected, the polynomial can be interpolated multiple times with only K values, to verify which one is responsible for the failure. It can be easily shown that using Lagrange algorithm, once performed the first interpolation, the computational effort for the following ones can be reduced by re-using the already calculated factors, so that the mathematical operations needed to perform cheating detection for K the shares is just approximately the double of the number needed for simple interpolation. A internal attacker can issue multiple requests to one of its neighbor asking the calculation of various $C(x)$. With delegation a node can refuse to answer to more then K-1 requests from the same node, without generating a failure in the protocol.

A number of *less than K internal attackers* are not able to perform a more efficient attack then a single attacker.

If *more than K nodes* are physically stolen, the network security is compromised, meaning that an attacker could introduce into the network as many nodes as it wants. Also, this allows interpolation of all the polynomials, so that an evil node could derive any $C(x)$ value. About the first issue we can say that when a node is stolen, its credentials are compromised, and any number of evil nodes can be equipped with the same credentials. The only way to avoid this issue is by using anti-tampering hardware, or bound physical parameters with crypto keys (similarly to what happens with WiMax, where the MAC address of the interface is authenticated with a digital certificate). In WSN there is normally no such possibility, so the theft of a node implicitly means that if the attacker is in possession of compatible hardware he can pollute the network with misbehaving nodes. About the second issue, this is intrinsic of any threshold scheme, and it still presents an advantage compared to a centralized scheme based on symmetrical keys; since the key is freshly generated, the attacker should be physically close to the nodes performing the authentication, while in a centralized scheme, without the use of public key cryptography, if a node close to the authentication server is compromised, it would be able to intercept a high number of authentication events even if they are happening far from its physical position (an example of this attack can be found even in WiMax authentication in mesh mode [7]).

A. Efficiency

If we imagine a network where nodes are disposed in a squared grid of size LxL and each node communicates with the 8 nodes adjacent, the first broadcast request reaches 8 nodes; if $K > 8$, then 8 more requests are sent, and 25 nodes are involved, each one sends exactly one response, so a total of 35 packets (2 for nonce exchange, 9 requests and a total of 24 responses) are sent.

If a gateway or authentication server is placed in the center of the grid (the best case) the average distance from the center is approximately one forth of the diagonal length, $\frac{L\sqrt{2}}{4}$. To have a bi-directional authentication using shared keys, the most practical way is that each side should perform a challenge and receive an answer, if the operation is started by the client, at least a 4-way handshake is needed [8], so a total of $L\sqrt{2}$ frames are sent. If we set K=25, then the same number of packets will be sent as in a centralized authentication with $L = 35/\sqrt{2} \approx 25$. If we use a 3-hops distributed authentication we can set K=49, equivalent to a centralized authentication in a network of size approximately 37x37 nodes. The main difference with a centralized model are:

- Each node sends at most 2 frames, while in a centralized system each involved node sends 4 frames. The 8 nodes that are neighbors of the server have 1/8 probability of being involved in *any* authentication that takes place *anywhere* in the network, so they will send an average of 0.5 packets per authentication, thus they are prone to a high energy consumption. In the distributed environment each node has the same probability of being involved in any authentication (in the case of K=49 it is $\frac{49}{37*37} = 1/27$, and at most 2 packets are sent, so that the average is approx. 0.074 frames per authentication).
- If the MAC layer of the network needs an average time t to successfully deliver a frame, with centralized authentication the total time needed grows linearly with the length of the path to the server, while in distributed authentication it grows linearly with the depth of the request. If a roaming node completes multiple authentications while moving, our technique outperforms the centralized scheme. Moreover in a centralized scheme if the MAC fails to deliver one single frame, the whole procedure must be repeated, while in our case this issue can be dealt by N_2 issuing redundant requests.
- In our evaluation we have considered an extremely optimistic centralized network, in reality we do not expect the gateway to be in the center of the network and the routes to be of optimal length, moreover if bidirectional routing is needed only for authentication (often it is not needed

in WSN, where the communication is mainly from the nodes to the gateway only) then there is a great waste of energy due to their construction and maintenance.
- Authentication can take place even if the authentication server is not reachable.

It must also be noted that the introduction of new nodes is costless, since new generation of nodes can coexist (this is a major issue in pre-distribution schemes).

IV. Implementation

There is an ongoing work to implement the proposed scheme on a real WSN testbed. The testbed is composed of MicaZ [9] nodes using TinyOS operative system, and the implementation has been part of a cooperation in the context of CRUISE European NoE. The goal of the implementation is to test the impact of the authentication algorithm in a mobile sink scenario, that is a scenario where a mobile node moves across the network and fetches data from static nodes. The data to be measured are mainly the quantity of memory, CPU usage and time needed for authentication. Right now the project is still under development and at the current stage the authentication has been implemented in a static environment, without the possibility of having multiple following authentications with distinct nodes and without delegation procedure.

We can summarize that memory occupation is not dependant on the K parameter, while it depends on the size of the integer used and the number of polynomials, as expected; the growth is linear with this values. In our tests we used 8 bit integers, even if this value may seem insufficient, we expect it not to vary up to 32 bits, which is the processor native register size. Over that size, instead of using larger integers, it is advisable to split the integers into 32 bits slices. For most WSN applications even 56/64 bit keys should suffice. All the code was contained in a 14KByte space, which we believe can be greatly reduced with code optimization. RAM usage never overcome 1.4 KBytes.

In fig. 2 the time needed to perform a single authentication is reported, with different degrees of the polynomial chosen, the measurements where performed using Tossim simulator that comes with TinyOS operative system, since the testbed at the moment doesn't include enough motes.

Changing the degree of the polynomial has the consequence of needing more neighbors nodes for the intermediary, that has to send a single broadcast packet and receive multiple unicast packets. Time increases also because of higher access to the media and interpolation of a higher degree polynomial. From the graphic we can see that the time needed, including delays introduced by a simple MAC scheme (the average impact of a simple backoff scheme is around 10%) are largely below 0.1 second. This speed seems extremely promising to support mobility, that produces several authentications one by the other.

V. Conclusions

The ongoing work over an efficient, distributed access control protocol for wireless sensor networks have shown that

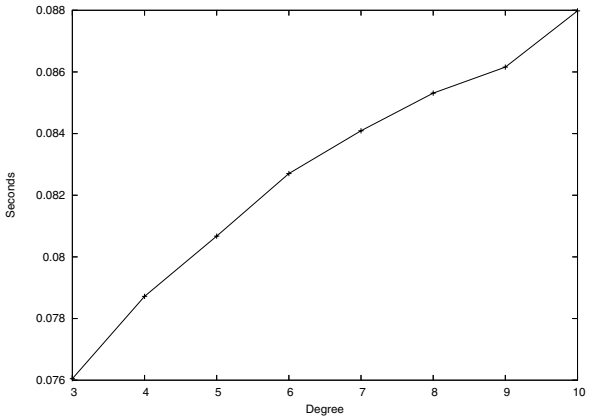

Fig. 2. Time in seconds needed to perform authentication, varying degree of the polynomial

polynomial functions can be used as a lightweight class of functions to achieve authentication using Shamir's secret sharing scheme. The protocol that has been designed overcomes some limits that are intrinsic to Shamir's algorithm and has been proved on real testbed, giving promising results. We outline some directions that will be analyzed in the following work:

- boot process techniques: during the boot phase of the network, an alternative authentication scheme must be detailed, up to when K nodes are authenticated. A simple scheme is to use a gateway that contains the whole polynomials definition, and can respond for K nodes.
- the space of the polynomials and the size of the keys must be carefully selected.

References

[1] J. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Chapter 17 Wireless Sensor Network Security: A Survey."
[2] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.
[3] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proceedings of 2003 Symposium on Security and Privacy*, pp. 197–213, 2003.
[4] F. Delgosha, E. Ayday, and F. Fekri, "MKPS: a multivariate polynomial scheme for symmetric key-establishment in distributed sensor networks," *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pp. 236–241, 2007.
[5] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-securing ad hoc wireless networks," *Proceedings of ISCC 2002. Seventh International Symposium on Computers and Communications*, pp. 567–574, 2002.
[6] A. Shamir, "How to share a secret," *ACM Communications*, 1979.
[7] L. Maccari, M. Paoli, and R. Fantacci, "Security analiysis of ieee 802.16," in *Communications, 2007 IEEE International Conference*, 2007.
[8] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung, "Systematic design of a family of attack-resistant authenticationprotocols," *Selected Areas in Communications, IEEE Journal on*, vol. 11, no. 5, pp. 679–693, 1993.
[9] [Online]. Available: http://www.xbow.com/Products/productdetails.aspx?sid=164