A Lightweight Distributed Group Authentication Mechanism

L. A. Martucci[†], T. C. M. B. Carvalho and W. V. Ruggiero
Department of Computer Engineering and Digital System - University of Sao Paulo – Brazil
email: {lmartucc, carvalho, wilson}@larc.usp.br

Abstract

Identifying trustable devices and establishing secure tunnels between them in ad hoc network environments is a difficult task because it has to be quick, inexpensive and secure. Certificate-based authentication mechanisms are too expensive for small devices. The use of such mechanisms must be controlled and reserved for special situations, (e.g. banking applications) but not for everyday transactions. In addition, indiscriminate use of asymmetric ciphering and certificate-based authentication is a shortcut to battery exhaustion attacks. This paper describes a lightweight distributed group authentication mechanism suitable for ad hoc network devices requirements. We introduce the concept of group authentication, the target of which is not the individual identification of devices, but to verify if a device belongs or does not belong to a trusted group. The proposed mechanism verifies if devices have a pre-shared secret and sets new cipher keys each time it runs. This mechanism requires loose synchronization among the devices' real time clocks to thwart replay attacks. It also mitigates the effects of battery exhaustion attacks due to its lightness.

Keywords

Security, Group Authentication, Ad Hoc Networks.

1. Introduction

Securing ad hoc mobile environments is not easy to be achieved in a quick, inexpensive and secure way. Security cannot rely on central servers, as there are no guarantees that they will be in radio range all times - devices' availability and motion are quite unpredictable in mobile ad hoc networks. Besides, complex configurations must be discarded, as target users of mobile ad hoc applications are the common audience and not security specialists.

In this paper, we propose a simple, but efficient, lightweight distributed group authentication mechanism that can be applied to the following scenario: a set of devices that belong to a single administration authority, such as a single user, a group (e.g. a family) or an enterprise that needs to exchange or synchronize data among devices, with or without the users' concern. In this paper, each set of devices is called *security cluster*. Each *security cluster* is composed of trusted devices that can recognize participants of known clusters through a mechanism called *group authentication*. While an individual authentication mechanism tries to identify devices and/or users univocally, group authentication only checks if two devices belong to a same (i.e. trusted) group. It is based on pre-shared secrets, which are distributed among devices of a security cluster (how this is achieved exactly is out of the scope of this paper).

Group authentication may be the only authentication mechanism available, but if a pre-shared secret is exposed in a single device, the whole group is compromised, as the secret is common to the entire

[†] Now with Karlstads Universitet - Institution för Informationsteknologi - Datavetenskap - Sweden

group. Group authentication can also precede individual authentication, in order to increase security, save power and even protect users' identities, as shown in this paper. The idea behind the proposed mechanism is straightforward and intuitive enough even for those not familiar with network security. It is extremely powerful as it can set strong short-term symmetric keys, which are never transmitted over-the-air, between devices. In addition, it is also completely transparent to the end-user. The lightweight distributed group authentication can be applied to ad hoc and non ad hoc networks, but its advantages are noticed on low-resource distributed computer environments.

The proposed distributed authentication mechanism can be applied at any OSI layer (from data link to application) and be bound to other authentication mechanisms, such as certificate based ones. Nevertheless, for specific cases group authentication may be enough (e.g. devices hosting non-critical services or low processing power and/or scarce battery resources). However, individual authentication may be necessary for devices hosting sensitive services and/or data. In these cases, the proposed mechanism can drastically reduce the number of unsuccessful authentication attempts using digital certificates and asymmetric keys, saving precious battery power.

This paper is organized as follows. In section 2, we present the architecture of the distributed group authentication mechanism preceded by a detailed description of how it works and sets new secret keys between devices. Section 3 evaluates the mechanism's security and its lightness compared to other mechanisms. Section 4 gives a survey to the related work, presenting some security mechanisms based on the same assumptions as this. Finally, conclusions are presented in section 5.

2. Lightweight Distributed Group Authentication Mechanism

Ad hoc networks' future environments (e.g. pervasive computing, sensor networks, etc.) rely on devices with major constraints regarding battery resources, processing power and available bandwidth. Thereupon, security mechanisms suitable for those devices are utterly important, as the establishment of a secure communication channel with a resource-expensive mechanism can lead to a successful battery exhaustion attack (Stajano and Anderson, 1999).

Before describing the proposed mechanism itself, we need to share our foresight of how ad hoc networks will be deployed in short and medium terms, as it will clarify the understanding and the meaning of this mechanism. From our point of view, the great majority of ad hoc networks will be of networks whose devices have something in common, such as their ownerships (e.g. an enterprise, a family, etc.) or placement (e.g. a meeting room, a house or even the streets). This presumption is reasonable, as several security mechanisms designed for ad hoc networks share this same foresight. In addition, we assume that it is possible to divide the whole ad hoc universe in small clusters of trusted devices (a similar approach can be found in (Capkun et al., 2003)). Furthermore, we assume that it is possible to set a pre-shared key among participant of a *security cluster*. Naturally, these clusters will overlap, as one device may belong to one or more groups. Figure 1 illustrates a mobile network, composed by several mobile devices divided in three security clusters (I, II and III).

The conclusion seems to be simple: if it is possible to set a pre-shared secret among devices that belong to a same *security cluster*, it is also possible to set secure sessions among them. However, important issues are concealed and have no easy answers: How are secure sessions going to be established? Is the lightweight distributed group authentication mechanism suitable? And why? This section attempts to provide answers to these questions.

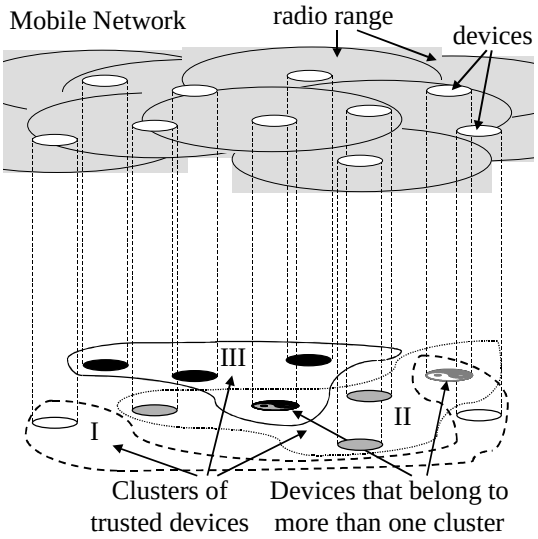

Figure 1: The ad hoc network universe divided in several clusters of trusted devices.

2.1 Authenticating Devices

In a few words, our distributed group authentication mechanism could be described as follows: first, the pre-shared secret *k*, which was previously set among devices belonging to one security cluster, is used as key input of a secure hash function (HMAC) with the current local time value, t_1, as data input. The output of the HMAC function, called 1^{st} *nonce*, is divided in three equal parts (*a*, *b*, *c*). After that, the timestamp t_1 and first part *a* are transmitted in a *challenge* message. A *nonce* can be set from one or more runs of the HMAC function. This initial step is illustrated below:

- $H_k(t_1) = 1^{st}\ nonce = (a, b, c)$, *challenge* message $= (a, t_1)$

How exactly and to which devices in the ad hoc network this information will be transmitted depends on which OSI layer the mechanism was implemented. On layers 2 and 3, for instance, broadcasting is the best option, but if implemented in layers 4 to 7, a TCP connection should be established before any data can be exchanged (see also section 3). After having receiving the challenge message, any other device from the same *security cluster* (shared-secret key *k* is known) is able to reproduce the 1^{st} *nonce* using timestamp t_1, and to recognize the slice *a* of the received challenge message as valid. Every device that receives and recognizes a *challenge* message generates a 2^{nd} *nonce*, using the shared-secret *k* as key input of the HMAC function and a second timestamp t_2 as data input. The 2^{nd} *nonce* is divided in three parts (*x*, *y*, *z*). The slice *z* is set as symmetric cipher key and is used to establish a cryptographic tunnel between the devices. *Response* message is then assembled with *x* and *b* and ciphered with *z*. Notice that the symmetric cipher key *z* was generated in run-time. The timestamp t_2 is also added to the message. This *response* message is sent back to the first device. Notice that only peer-to-peer (and no multiparty) authentication exists, as different t_2 are expected from different devices. This sequence is presented below:

- $H_k(t_2) = 2^{nd}\ nonce = (x, y, z)$, *response* message $= (E_z[x, b], t_2)$

The first device can reproduce the 2^{nd} *nonce* from the received timestamp t_2 and generate the symmetric key *z*. After that, it must decipher the message payload and check its contents, *x* and *b*. It then assembles the last message of our authentication mechanism. The *last* message contents are *c* and *y*, both ciphered using *z* as symmetric key.

- *response* message recognized, *last* message $= (E_z[y, c])$

The *Last* message is used for confirmation purposes. After receiving the last message, the second device is sure that the first device really knows the symmetric key *z* and, hence, *y* and *k*. When this protocol ends, two devices from a same *security cluster* can securely exchange data using *z* as temporary symmetric cipher key. The protocol described above can also be restarted at any moment in order to re-authenticate both ends and set a new, fresh temporary cipher key *z*', which is also never transmitted over-the-air and with only two HMAC additional runs. A deeper evaluation of the mechanism from the security point of view is provided in section 3.

2.2 The System Architecture

The proposed mechanism should be placed between the internal system (e.g. software application) and the external communication (e.g. wireless interfaces), as a security middleware (see subsection 2.4). Figure 2 shows the architecture of the proposed mechanism and its internal building blocks.

HMAC is a suitable sequence generator for our mechanism, as its inputs are: a secret key and a data input (timestamps in our case). Moreover, it has a 160-bit long hash value output (with SHA-1 as embedded hash function). The two basic requirements for sequence generator candidates were:

- It must be cryptographic secure, or polynomial-time unpredictable.
- It must accept an arbitrary value as input parameter (the timestamp).

The *Seed Box* is a storage unit responsible to hold all known pre-shared keys k_n, where each k_n corresponds to one different *secure cluster*. Each k_n receives a mnemonic name, assigned by the device's owner, to be easily associated to a *secure cluster*. The *I/O* is the building block responsible for the communication of the mechanism with the network communication interfaces (the layer just below the mechanism). The *Control Unit* has four functions: the first is to be an interface between the mechanism itself and applications from upper levels. The second regards re-keying and the last two are directly related with security: source address and timestamp verification and message ciphering. Messages are only considered valid if their timestamps values are within the lower and upper bounds of a *time window* set around the device's current time given by its real time clock.

The *Active Messages* is a storage unit responsible to store all valid sent messages, the timestamps and *nonce*s. This block is particularly useful to prevent message fabrication attacks (see in section 3). Every message sent is considered valid if it is not expired. Expiration is determined by messages' timestamps. *Active Messages* storage blocks also checks if timestamp information is used more than once and discards messages with repeated timestamps, in order to increase security.

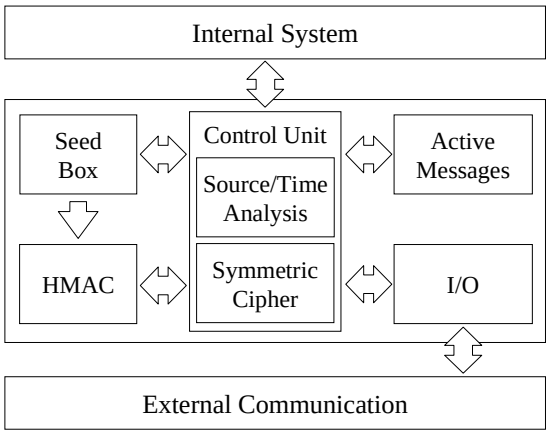

Figure 2: The Architecture of the Lightweight Distributed Authentication Mechanism

2.3 Loose Synchronization among Devices and Modular Security

Devices belonging to the same security cluster must have their clocks loose synchronized; otherwise their peers may discard authentic challenge messages if the message's timestamp is out of the bounds defined by the time window. Therefore, time windows should not be set too narrow if no time synchronization service is available in the network (e.g. a local NTP - Network Time Protocol - running for members of a security cluster only). The design of our distributed group authentication mechanism is completely modular. Therefore, it can be associated with other security mechanisms. If individual authentication is mandatory, a certificate-based authentication can happen just after the secure tunnel to be set between the devices that had already gone through the lightweight group authentication. This fact is particularly important when dealing with mobile devices, because their resources are often scarce and the indiscriminate use of expensive functions, as certificate-based authentication, must be reserved to very special situations or critical applications only.

2.4 Re-Authentication, Re-Keying and Implementation Layer

Any peer can request at anytime a renewal of the group authentication procedure to set a new symmetric cipher key *z* between the devices. The re-keying is transparent to upper layer applications and is made inside the secure tunnel already set, thus the re-keying procedure is concealed from outsiders (that are not aware of the re-keying procedure). In addition, re-authentication frequency is not fixed, and shall be agreed between communicating devices. Group authentication mechanism can be implemented at any layer of the OSI protocol (from data link to application), but security aspects change regarding to the chosen OSI layer. For instance: a data link layer implementation can only offer data link security, what can be used to conceal the device's hardware address and, thus, prevent tracking. On the other hand, upper-TCP implementations offer end-to-end security and can be used to tie applications to security clusters, in order to increase control over applications network access.

3. Security Evaluation

In this section, we provide a security evaluation of the proposed distributed group authentication scheme using an attack-oriented approach. We also emphasize its lightness and estimate how much power can be saved by its deployment along with a certificate-based solution, instead of relying on certificate-based solution only. Theoretical attacks against our mechanism are performed to evaluate its efficiency to thwart them, protect devices and transmitted data. The most relevant security attacks against our mechanism are: fabrication (including replay attack), man-in-the-middle (MitM) and brute-force attacks. All them proved infeasible against our mechanism. We had also implemented a prototype of the proposed mechanism using Java and 64-bit UTC (Universal Coordinated Time) timestamps. Our prototype runs over TCP and precedes a TLS procedure, allowing the certificates to be exchanged by the devices communicating inside a secure tunnel, protecting users' identities from potential attackers. The implementation is a peer-to-peer application, suitable for ad hoc networks.

3.1 Man-in-the-Middle (MitM) and Replay Attacks

A *man-in-the-middle* attack, or just MitM, happens when an attacker device *E* places itself in the middle of two legitimate devices *A* and *B* and masquerade as *B* to *A* and as *A* to *B*. Our proposed scheme thwarts MitM attacks as symmetric key *k* is set on both ends *A* and *B* and the rest of the

communication between both ends is done inside a secure tunnel. Therefore, intermediary nodes only forward ciphered packets (with the very exception of *challenge* messages).

Replay attacks are a combination of two different network attacks: a passive attack (interception) and a fabrication attack. An attacker *E* can easily capture valid messages being exchanged between two devices, *A* and *B*, that belong to the same security cluster *S*, without being noticed. After that, *E* may try to reuse this information by sending it to a fourth device, *C*, that also belong to *S*. The protection against replay attacks is provided by the use of timestamps as HMAC data input. The *Control Unit* block ignores *challenge* messages with timestamps that were already used and the *Active Messages* block, in association with *Control Unit* block, prevents *response* or *final* messages to be received without being related with a *challenge* message. In addition, only messages with a valid timestamp (within device's time window) are accepted. Therefore, a captured *challenge* message have to be retransmitted before its validity expires, otherwise it is useless.

3.2 Brute-Force Attack

A brute force attack consists of trying every possible key until the right one is found. If an attacker captures a *challenge* message, it can produce and send multiple *response* messages back to the originator. The lifetime of the *challenge* message is used to protect the device against this attack, as no response shall be expected to an expired message. Lifetime defines the period susceptible to brute-force attacks. The *Control Unit*'s source analysis tracks the source address of the incoming *response* messages and checks if multiple answers are coming from a single device (and, eventually, blocking messages arriving from it). Intercepted messages may also be submitted to a brute-force attack in order to obtain the shared-secret *k*. However, finding out a shared-secret key of 160 or 512-bit long is extremely expensive. If we assume that generating one *nonce* and comparing its first part with another part captured from a *challenge* message takes around 1000 cycles (950 cycles for SHA-1 (Bosselaers et al. 1996) and 50 more for other digital operations needed), a state-of-art 3GHz microprocessor fully committed on finding a 160-bit long key *k* would take approximately 7.7×10^{33} years to find it out (assuming that the attacker discover *k* in *n*/2 attempt, where *n* is the maximum number of attempts). On the other hand, the effectiveness of a brute-force attack over the symmetric cipher key *z* depends on the length of *z*, and on the output length of HMAC.

3.3 Lightweight Power-Saving Mechanism

Lightness may sound a bit strange for a security evaluation section, although it is a fundamental security matter when we aim ad hoc network devices with low battery resources. Power saving is a need and the proposed mechanism spends it wisely. Small devices running applications that don't need individual authentication can establish secure tunnels without the need of asymmetric ciphers as new shared-key are set between devices with only two HMAC runs. However, if a service truly demands individual authentication, certificates are exchanged as soon as the secure tunnel is set. This procedure helps devices to save battery power, as a high percentage of arriving certificates are expected to be valid, as they already had gone through group authentication, mitigating battery exhaustion attacks effects.

4. Related Work

Our lightweight power-saving distributed group authentication mechanism is based on the association of shared-secret and a secure sequence generator that takes as input parameter public

information (timestamp) and a secret. Authentication mechanisms that rely on the same assumptions and authentication systems for mobile communications are reviewed in this section.

SecurID authentication (RSA Security, 2001) is also based in a pseudo-random number generator and time information, although it is not a distributed solution, since it relies on a centralized authentication server. Moreover, SecurID uses tokens working as number generators, and passwords. SecurID and our mechanism goals are not the same, as SecurID pursues user authentication, while ours seeks group device authentication. Furthermore, our group mechanism is transparent to end-users, and also sets a secure tunnel between devices. The SIM (*Subscriber Identity Module*) is another authentication system used in GSM mobile systems based on a one-way hash function module (Schmidt et al., 2002). SIM relies on challenge-response procedures, with 128 bits keys, but only 32 bits of response. SIM is transparent to end-users and also lightweight, suitable for mobile devices, but it relies on centralized servers to verify incoming responses message.

IEEE 802.11 WEP is based on a PRNG that generate sequences to be used to cipher messages. However, the RC4 PRNG using a secret key of 40 bits is considered weak, and several attacks over WEP were published in the last few years (Borisov et al., 2001), and even open-source tools to break it are freely available. The lighter IEEE 802.11i next-generation security protocol for wireless networks being evaluated is the TKIP (*Temporary Key Integrity Protocol*) (Walker et al., 2002). TKIP masks WEP weaknesses, encrypting data with secret keys of 128 bits, periodically renewing the symmetric cipher key and preventing IV (*Initialization Vector*) to be repeated with the same cipher key. However, the re-keying relies on an EAP-based server, a centralized device. Moreover, IEEE 802.11i CCMP (*Counter Mode with CBC-MAC Protocol*) proposal also has its key management relying on an EAP-based server. Therefore, none of these two IEEE 802.11i security proposals are suitable for an ad hoc network unless every device on the ad hoc network runs an EAP-based authentication server. Furthermore, the only EAP that meets all IEEE 802.11i requirements is the EAP-TLS, which works with digital certificates and asymmetric cryptography.

IKE with a pre-shared key (Harkins and Carrel, 1998) can also be used as pre-authentication mechanism. Its advantage is that no loose synchronization among devices real-time clock is needed. In addition, only three messages are needed with IKE authentication with pre-shared keys in aggressive mode (the same amount needed by the pre-authentication mechanism described previously). However, IKE with pre-shared keys has a major drawback that impacts power-consumption: messages can be replayed. Even though a replay attack will not succeed, as attackers do not have the pre-shared-key, replayed challenge messages are always replied, as there is no time information in the message payload. Therefore, a replayed challenge message will, initially, pass as authentic for the IKE Responder and will be replied, causing battery power to be spent (data transmission mode is the most expensive mode in terms of energy consumption (Feeney and Nilsson, 2001). In conclusion, it is not difficult to foresee that IKE authentication with pre-shared keys spends more battery power than the pre-authentication mechanism presented in the previous subsection when submitted to a battery-driven attack.

SKEME with a pre-shared-key (Krawczyk et al. 1997) is also a candidate for pre-authentication mechanism. SKEME with a pre-shared key advantages are basically the same of IKE with a pre-shared key: real time synchronization among device's real time clock is not needed and only four messages are exchanged between mobile devices. However, its disadvantages are also the same as IKE with a pre-shared key: replayed messages will be answered, and battery power spent. The conclusion is exactly the same of IKE with pre-shared keys: it will spend more battery power than

the pre-authentication mechanism presented in the previous subsection when submitted to a battery-driven attack.

5. Summary & Conclusions

In this paper, we have proposed an efficient lightweight distributed group authentication mechanism as a feasible solution to secure mobile ad hoc networks. We have shown how group authentication works and how it is implemented, assuming that it is possible to distribute a secret among trustable devices. We also have explained how a secure tunnel is set between each pair of mobile devices and how a symmetric cipher key is derived from an initial pre-shared secret. We also illustrated how we renewed the symmetric cipher key automatically in a distributed environment. We associated a name to each *security cluster* to make it intuitive and straightforward even for the common audience.

Group authentication provided by the proposed mechanism can be sufficient for almost every wireless device. Moreover, we believe that individual authentication is restricted to few applications, and the lightness provided by our mechanism when compared with certificate-based mechanisms, justifies its use as everyday solution for security. We have also illustrated how the proposed mechanism thwarts security attacks, such as MitM and replay attacks. On purpose, we have not selected a specific symmetric key cipher for the proposed mechanism, as we were aiming an open solution that works with any kind of mobile devices, even with legacy and simple devices with very few resources and computational power. We emphasize that the proposed mechanism is not only an ad hoc networks secure solution, and can be set on any kind of computer networks, offering a light and distributed security solution.

6. REFERENCES

Borisov, N. Goldberg, I. and Wagner, D. (2001), Intercepting Mobile Communications: the insecurity of 802.11. In: International Conference on Mobile Computing and Networking - MobiCom'01. *Proceedings*. Roma, Italia.

Bosselaers, A., Govaerts, R. and Vandewalle, J. (1996), Fast Hash on the Pentium. In: Advances in Cryptology - CRYPTO'96, Lecture Notes in Computer Science, Springer-Verlag, *Proceedings*. Santa Barbara, CA, USA.

Capkun, S., Hubaux, J.P. and Buttyan, L. (2003), Mobility Helps Security in Ad Hoc Networks - MOBIHOC'03, 4. *Proceedings*. Annapolis, MD, USA.

Feeney, L. and Nilsson, M. (2001), Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In: IEEE Infocom'01, 20. *Proceedings*. Anchorage, AL, USA.

Harkins, D. and Carrel, D. (1998), RFC 2409. The Internet Key Exchange (IKE). *IETF Network Working Group Request for Comments*.

Krawczyk, H., Bellare, M. and Canetti, R. (1997), RFC 2104. HMAC: Keyed-Hashing for Message Authentication. *IETF Network Working Group Request for Comments*.

RSA Security Inc. (2001), RSA SecurID Authentication: a better value for a better ROI. RSA *Whitepaper*. Available at: <http://www.rsasecurity. com/ products/securid/>.

Schmidt, B. Schimmler, M. and Adi, W. (2002), Area Efficient Modular Arithmetic for Mobile Security. In: International Conference on Wireless Networks - ICWN'02. Las Vegas: CSREA Press. *Proceedings*. Las Vegas, NV, USA. p.208-214.

Stajano, F. and Anderson, R. (1999), The resurrecting duckling: security issues for ad hoc wireless networks. In: AT&T Software Symposium, 3., Middletown. *Proceedings*. NJ, USA.

Walker, J. et al. (2002), IEEE 802.11i Overview. In: NIST 802.11 Wireless LAN Security Workshop. Falls Church. *Slides*. Falls Church, VA, USA, 2002.