Journal of Systems Architecture 61 (2015) 256–266

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Distributed storage protection in wireless sensor networks

Gianluca Dini, Lanfranco Lopriore [*]

Dipartimento di Ingegneria dell'Informazione, Università di Pisa, via G. Caruso 16, 56126 Pisa, Italy

ARTICLE INFO

Article history:
Received 10 June 2014
Received in revised form 2 February 2015
Accepted 24 March 2015
Available online 30 March 2015

Keywords:
Access right
Protection
Sensor node
Symmetric-key cryptography

ABSTRACT

With reference to a distributed architecture consisting of sensor nodes connected in a wireless network, we present a model of a protection system based on segments and applications. An application is the result of the joint activities of a set of cooperating nodes. A given node can access a segment stored in the primary memory of a different node only by presenting a gate for that segment. A gate is a form of pointer protected cryptographically, which references a segment and specifies a set of access rights for this segment. Gates can be freely transmitted between nodes, thereby granting the corresponding access permissions. Two special node functionalities are considered, segment servers and application servers. Segment servers are used for inter-application communication and information gathering. An application server is used in each application to support key management and rekeying. The rekey mechanism takes advantage of key naming to cope with losses of rekey messages. The total memory requirements for key and gate storage result to be a negligible fraction of the overall memory resources of the generic network node.

1. Introduction

We shall refer to a distributed architecture consisting of sensor nodes connected in a wireless network. In an architecture of this type, stringent limitations exist in terms of the hardware resources available in each node [15]. These limitations include the lack of hardware support for the two usual processor modes, a kernel (privileged) mode and a user (non-privileged) mode with restricted memory access, a limited memory space, and the absence of a memory management device for virtual to physical address translation [12,20,21]. It follows that, within the node boundaries, no separation exists between the kernel space and the user space, for instance.

In an environment of this type, we shall refer to a protection system featuring applications and segments. A *segment* is a contiguous memory area entirely contained within the boundaries of the primary memory of a single node. Segments are the basic unit of information gathering and transmission between the nodes. An *application* is the result of the joint activities of a set of cooperating nodes (the application *members*). We make no hypothesis on the activity model of each member, which can be a scheduled computation [3] or, in an event driven environment, a routine activated by a hardware interrupt [10,24].

A classical approach to access right representation in memory is based on the concept of a *password capability* [1,5,16,27]. In a segment-oriented, password-capability architecture, the protection system associates a set of passwords with each memory segment. Each password corresponds to an access permission. A password capability is a pair (S, w) where S is a segment identifier and w is a password. If a match exists between w and one of the passwords associated with segment S, then the password capability grants its holder the corresponding access permission on S. If passwords are large and sparse, password capabilities can be freely mixed in memory with ordinary data items; an illegal attempt to modify an existing password capability (e.g. in view of an undue amplification of access permissions), or even to forge a password capability from scratch, is destined to fail, as the probability of guessing a valid password is vanishingly low.

A salient feature of password capability protection is simplicity in access right distribution. A process that holds a valid password capability can grant the corresponding access rights to another process by a simple action of password capability copy, from its own address space to the address space of the recipient process. In turn, the recipient process may well transmit the password capability to a third process. In a situation of this type, it is hard, if not impossible, to keep track of all copies of a given password capability that exist in the system at the same given time. This exacerbates the problem of access right revocation: the original owner of a given password capability should be in a position to retract the password capability from each subsequent recipient,

* Corresponding author.
E-mail addresses: g.dini@iet.unipi.it (G. Dini), l.lopriore@iet.unipi.it (L. Lopriore).

http://dx.doi.org/10.1016/j.sysarc.2015.03.002
1383-7621/© 2015 Elsevier B.V. All rights reserved.

selectively. Of course, if we modify the passwords of a given segment, we revoke all the password capabilities referencing that segment. This revocation mechanism cannot be used for selective revocation of a subset of all the password capabilities for the same given segment.

In this paper, we shall refer to a variant of the password capability model that has been designed to comply with the resource limitations, outlined above, which characterize the sensor nodes in a wireless sensor networks. In our approach, within a node, every software routine has unlimited access to the whole primary memory of that node, irrespective of segment boundaries; whereas a routine running in a given node can access a remote segment stored in the primary memory of a different node only by presenting a *gate* for that segment. A gate is a form of password capability protected cryptographically, which references a segment and specifies a set of access rights for this segment. Possible access rights are read, write, or both read and write. Gates are protected from tampering by a form of symmetric-key cryptography [13,33], superior to public key cryptography in both terms of low computation requirements and low energy costs [4,17].

In a sensor node, the high memory cost of a set of passwords for each memory segment is not acceptable. Our gate implementation uses a single set of password for each node. We have obtained this result by taking advantage of cryptography to incorporate the name of the segment referenced by a given gate into the protection field of this gate. A small set of system primitives, the *protection primitives*, makes it possible to define segments, to generate gates for existing segments, and to use gates in remote segment accesses. A node that generates a gate is free to transmit this gate to another node, thereby granting the corresponding access permissions to the recipient node. Two or more segments can be defined for the same memory area. By deleting one of these segments, we revoke the gates referencing this segment; revocation does not affect validity of the gates for the remaining segments.

The rest of this paper is organized as follows. Section 2 introduces our protection environment with special reference to segments and gates. The protection primitives are presented, and the actions involved in the execution of each primitive are illustrated with special reference to interactions between nodes. Section 3 presents our application model. Two special node functionalities are introduced, the *application server*, used within the application boundaries to support information gathering, key management and rekeying, and the *segment server*, used for inter-application communication. Section 4 discusses the motivations for the proposed organization from a number of salient viewpoints, including the hardware limitations existing in sensor nodes, gate manipulation and revocation, security, and the memory requirements for key and gate storage. We consider two different network topologies in special depth, a configuration featuring a form of full pairwise connectivity at the application level, and a hierarchical topology featuring a *general server* that gathers data from all the application servers. Relations of our work to previous works are outlined. Section 5 gives concluding remarks.

2. The protection model

2.1. Segments

In the previous section, we have defined a segment as a contiguous memory area that is entirely contained within the boundaries of the primary memory a single node. A segment S is identified by pair $S=(M, C)$ where M is the node storing S, and C is the *local identifier* of S in M. In node M, a table, the *segment table* ST_M, contains the associations of local segment identifiers with the corresponding areas in the primary memory of that node. The table entry for local segment C, i.e. segment $S=(M, C)$, contains the starting address B of this segment in the primary memory of M (the segment *base*) and the segment *length* L.

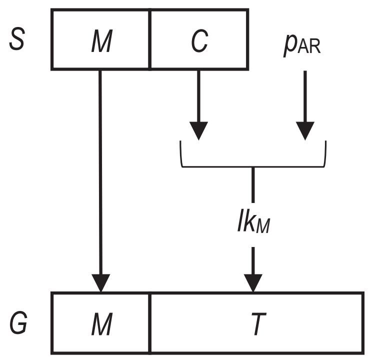

Fig. 1. Generation of gate $G=(M, T)$ that specifies access right AR for segment $S=(M, C)$. p_{AR} is the password that corresponds to AR in the password set P_M associated with node M.

2.2. Gates

A set P_M of three *passwords*, $P_M=\{p_{\mathrm{R}}, p_{\mathrm{W}}, p_{\mathrm{RW}}\}$, is associated with each given node M and is stored in the primary memory of this node. Each password corresponds to an access right for the segments in M. Password p_{R} corresponds to access right R, which makes it possible to access the segments for read. This is similar to password p_{W} for access right W, which makes it possible to access the segments for write, and to password p_{RW} for access right RW, which makes it possible to access the segments for both read and write.

A gate G referencing segment $S=(M, C)$ in node M is a pair $G=(M, T)$, where T is a *protection field* that includes the specification of the segment local identifier C and a password p. If a match exists between p and one of the passwords in P_M, then the gate grants the corresponding access right for segment S. Quantity M is in plaintext, whereas quantity T is encrypted by using a symmetric-key cipher and a cryptographic key, called the *local key* lk_M, which is associated with node M. lk_M is stored in node M; it is never transmitted or revealed by M to any other node, and it is exclusively aimed at encrypting the gates for the segments in M.

Fig. 1 shows the generation of gate $G=(M, T)$ granting access right AR for segment $S=(M, C)$, AR being one of R, W or RW. Let p_{AR} denote the password in P_M that corresponds to this access right. Quantity T is the result of encrypting pair (C, p_{AR}) by using a symmetric key cipher and local key lk_M. Thus, a gate referencing a segment in node M can only be assembled in this node, as gate generation requires knowledge of local key lk_M. Throughout this paper, we assume that ciphers comply with an encryption mode supporting both authentication and confidentiality, e.g. the Counter with CBC-MAC (CCM) mode [11].[1]

Fig. 2 shows the reverse transformation of gate G into plaintext. Local key lk_M is used to decrypt the protection field T and obtain quantities C and p. Quantity p is compared with the passwords in P_M to validate the result of the transformation. If a match is found and p_{AR} is the matching password, validation is successful, gate G references segment $S=(M, C)$ and specifies the access right

[1] Intuitively, a single encryption key can be used for both authentication and confidentiality. The sender authenticates the header and the payload, it appends the resulting Message Identification Code (MIC) to the payload and, finally, it encrypts the bundle. The receiver decrypts the ciphertext into a payload and a MIC, and verifies the MIC against the received header and payload.