Journal of Systems Architecture 60 (2014) 11–21

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Hardware security platform for multicast communications

José M. Granado-Criado *, Miguel A. Vega-Rodríguez, Juan M. Sánchez-Pérez, Juan A. Gómez-Pulido

Department of Computers and Communications Technologies, University of Extremadura, Escuela Politécnica, Campus Universitario s/n, C.P. 10003 Cáceres, Spain

ARTICLE INFO

Article history:
Received 4 December 2012
Received in revised form 23 September 2013
Accepted 17 November 2013
Available online 27 November 2013

Keywords:
Security platform
Multicast communications
Group key management
System on chip
Logical Key Hierarchy
Advanced Encryption Standard

ABSTRACT

Secure multicast applications of multimedia contents, such as Internet TV, pay per view, satellite TV, etc., need to maintain a high number of keys. In these applications, a user contracts a group of channels or even specific content (films, sports, etc.) which do not have to coincide with the services contracted by other users, so different keys are needed to encrypt the contents. These keys must be recalculated, encrypted and redistributed when a user joins or unjoins a specific group in order to prevent users who do not belong to a group from being able to access the contents. Original algorithms generate only one group key for all users, so this key must be recalculated and resent when a user joins or unjoins in the user group. This is an important problem, because a group key could be changed even when one content is performing. This paper presents a high performance implementation of one of the most employed algorithms of group key maintenance, the LKH algorithm, using reconfigurable hardware and a very high and realistic number of users (8,388,609). The performance obtained by this study improves a lot other results found in the literature in terms of both performance and number of users.

1. Introduction

Currently, services like pay-per-view, IP television, and satellite television are at their height. These applications encrypt their content to prevent users who have not contracted the content from being able to access it. Other applications like Digital Video Broadcasting (DVB) encrypt their content to restrict the broadcasting area. So, a user key and a group key (the key used to send the encrypted content to all users) must be sent to each user. However, the group key is recalculated when users join or unjoin the group. Moreover, when the group key is recalculated, it is also encrypted with every user key before these are sent, in order to ensure that if an unauthorized person accesses the key he will not be able to access the content. The encryption of a new group key could be a high computational task if the number of users was very high. In Fig. 1, where every user has their own key plus the group key, the number of encryptions when the group key is changed is equal to the number of users. Obviously, a scalability problem arises.

The main contribution of this paper is to create a hardware platform to perform the group key management. In this way, several improvements of cryptographic modules have been implemented (a complete description of these improvements can be seen in the following sections):

- AES module: several operations have been parallelized. Moreover, some phases of AES have been combined to reduce the number of clock cycles needed to perform an encryption.
- Hash module: this module has been modified to combine the operations of Matyas–Meyer–Oseas Hash Function [16] with the own cryptographic module used, in this case AES. Therefore, the hash module can perform a hash operation so fast as the AES module.
- MAC module: this module has been changed in the same way as the hash module, that is, combining the own MAC operation with the AES module with the same result in terms of performance.
- RSA Module: This module implements a highly improved version of RSA public cryptographic algorithm. Particularly, this module implements a bit parallel version of Montgomery Modular Multiplication and 8-bit carry out adders, reducing the complete number of clock cycles needed for this very expensive operation.

Finally, a dual core implementation has been employed. The architecture has two MicroBlaze processors in order to combine the new keys generation and the LKH tree modification.

The problem of group key management has been addressed in recent years as a means of reducing the rekeying costs and achieving a scalable group key management. Paper [1] shows an LKH implementation using FPGAs, which uses a PowerPC-based System-on-Chip architecture to implement a security platform.

In [2–4] an n-ary tree implementation of a group key tree is presented; Refs. [3,4] show theoretical results, hence comparison with

* Corresponding author. Tel.: +34 927 25 70 00; fax: +34 927 25 72 02.
E-mail addresses: granado@unex.es (J.M. Granado-Criado), mavega@unex.es (M.A. Vega-Rodríguez), sanperez@unex.es (J.M. Sánchez-Pérez), jangomez@unex.es (J.A. Gómez-Pulido).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.11.007

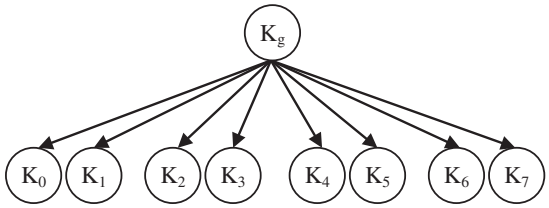

Fig. 1. Basic group key management technique.

these two references will not possible in the comparison section. The problem of an n-ary tree to implement the key tree is that the search of a free leaf is more expensive than a binary tree. So, in this paper a binary group tree has been chosen in order to reduce the complexity of the search functions and of the Key State Memory.

Papers [5] and [6] present a balanced LKH implementation. Both present simulated results and hence it is not possible to make fair comparison with these studies. These studies' main aim is to reduce the number of keys stored in the system. The problem of these works is that keeping balanced the tree continuously entails a performance penalization and do not reduce the employed memory. However, in the present work the main aim is to increase the performance in the key management, so keeping the Logical Key Tree balanced all the time is not necessary.

In paper [7] a complete key management protocol is proposed, and finally in researches [8], [9], [10] other different key management algorithms are implemented, specifically SDR, NSBHO and WPMKT. The present paper implements the LKH key management method because it reduces the number of encryptions needed when a user joins or disjoins a group. Moreover, due to it is a binary tree based algorithm, it is very fast to find a free leaf when it is necessary.

Summarizing the above paragraphs, some works solve the presented problem by different ways, but they bring some performance penalizations due to either their search functions (n-ary trees) or in their tree management (keeping balanced the tree all time). Moreover, a very preliminary version of this paper was published in the proceedings of a national conference [11]. This paper has been greatly extended and improved. The earlier version does not include the MAC, Hash and RSA modules. Accordingly, that implementation was an incomplete security platform, which only includes the key management; no other security methods were implemented. These security modules guarantee security at all steps in the multicast communication process; in particular, they prevent unauthorized data modification and sender identification.

This paper presents a hardware implementation of one of the best-known techniques in the key management problem, specifically the Logical Key Hierarchy (LKH) algorithm, which reduces the number of encryptions needed to resend a group key by dividing the users into subgroups, each with one subgroup key. Some of these new subgroup keys will also be recalculated when a new group key is needed, so the number of keys generated increases. However, the number of encryptions needed to resend a group key to all users is reduced. Table 1 shows the number of encryptions needed in the LKH and the basic techniques. For example, if a number of users of 8,388,608 is employed (which is the number of users assumed in this paper), the number of encryptions for one joining and one unjoining is reduced from 8,388,609 (2+8,388,607) in the basic technique to 90 (46+44) in LKH.

Table 1
Number of encryptions needed for n users in the basic technique and in the LKH technique.

	Basic technique	LKH
User join	2	$2 * \log_2 n$
User unjoin	$n - 1$	$2 * (\log_2 n - 1)$
Average	$O(n)$	$O(\log n)$

The hardware platform designed consists in an embedded implementation using a hardware reconfigurable device (FPGA). Specifically two MicroBlaze soft-processors have been used to implement a System-on-Chip architecture with several cryptographic hardware coprocessors.

This paper is structured as follows: Section 2 shows the key management algorithm employed in this study, including the LKH description, the key generation algorithm and the memory organization. In Section 3 all the cryptographic modules and their hardware implementations are described. Section 4 presents the complete hardware architecture implemented. Section 5 analyzes the final results, comparing them with results found in other researches, and finally conclusions are stated in the last section.

2. Description of the key management algorithm

2.1. Logical Key Hierarchy

The Logical Key Hierarchy (LKH) [12] consists in storing a complete group of users in a binary tree. In this tree, the root node is the group key, every leaf node is a user key and every inner node is a subgroup key (called help-keys). These help-keys allow a reduction in the number of encryptions needed when a user joins or unjoins the group. This means that one user must store several keys, specifically its identifier key (K_i), which is known only by that user and the server, the group key (K_g), known by all group members, and every help-key (K_{i-j}) between the group key and that user in the key tree.

As indicated previously, the help-keys reduce the number of encryptions needed when, for example, a user unjoins the group. In that case, in order to prevent that user being able to access future content, a new group key must be generated. In the basic technique, when a new group key is generated, it must be encrypted by every user key; hence there is a very high computational cost task if the number of users is substantial. However, if the LKH method is employed, only two encryptions of the group key are needed, specifically one for the left branch and one for the right branch. On the other hand, all the help-keys stored by the former user must be recalculated again, encrypted and sent to their corresponding users. However, the number of encryptions needed to send the new help-keys is much lower than the number of encryptions in the basic techniques.

For example, Fig. 2(a) shows a complete LKH with eight users. If user 4 unjoins the group, all his keys are obsolete, that is, K_g, K_{4-7} and K_{4-5}, so they must be recalculated (K'_g, K'_{4-7} and K'_{4-5}). In this case, the following encryptions are made (where $E_k(X)$ indicates that X is encrypted using key K): $E_{K5}(K'_{4-5})$, $E_{K4-5}(K'_{4-7})$, $E_{K6-7}(K'_{4-7})$, $E_{K0-3}(K'_g)$, $E_{K'4-7}(K'_g)$. Because this is a very simple example, only 2 encryptions are saved as compared with the basic technique where seven encryptions are needed. However, if the number of users is higher, the encryption decreases exponentially. Moreover, another encryption can be saved if the tree does not store help-keys in nodes which do not have users in both branches (for example node K_{4-5} after user 4 disjoins the group). This makes operation $E_{K5}(K'_{4-5})$ unnecessary and changes operation $E_{K4-5}(K'_{4-7})$ to $E_{K5}(K'_{4-7})$. Fig. 2(b) shows this situation.

In order to know if a specific node has both left and right users, another tree structure must be employed: the Key State Memory (KSM). The KSM stores, only for the root and the help-key nodes, a four-state (two-bit) value which indicates whether there are users in the branches of the corresponding node. The KSM does