Access Control in Networks Hierarchy: Implementation of Key Management Protocol

Nicolas Sklavos and Odysseas Koufopavlou

(Corresponding author: Nicolas Sklavos)

Electrical & Computer Engineering Dept., University of Patras
Patras, Greece (Email: nsklavos@ieee.org)

(Received June 25, 2005; revised and accepted July 6, 2005)

Abstract

The special needs for cryptography, of both wired and wireless networks, have attracted the researchers' major interest in the design of new security schemes. This work deals with the access control in network hierarchy. More analytically, an efficient architecture and the implementation of a key management protocol are proposed in this paper. This protocol main philosophy is centered in the usage of hash functions. Alternative hash functions have been implemented and studied, in order to select between the most efficient proposed architecture, concerning both performance and allocated resources. Finally the dynamic access of the system is presented. The proposed system could be applied efficiently in networks with multi-nodes and multi-users authentication demands, providing high speed performance and high level security strength.

Keywords: Access control, hardware integration, hash functions, key management, network authentication

1 Introduction

Access control plays an important role in the area of information security, which ensures that any access to available data and offered services are authorized [13, 15]. In a network hierarchy, a Central Authority (CA) assigns a key for each node. The problems that are rising upon to this issue is how the users of higher level nodes, can derive the keys of their direct of indirect child nodes.

The first possible solution that someone can apply to this problem is a user to keep all the keys of its direct and indirect child nodes. This solution has hard allocated resources cost, in terms of memory cells or registers, especially in the cases of networks with high complexity hierarchy. This also causes problems concerning the key management function and the supported security level also.

Akl and Taylor [1, 2] proposed another solution to this problem, based on a cryptographic approach. The advantage of this security scheme is that the key generation and derivation algorithms are simple enough. The main disadvantage of this scheme is that it is centered in the use of public parameters. The last ones have to be selected properly in order to guarantee security and at the same time a lot of resources are needed for storage purposes.

Nowadays there are many proposed solutions for multilevel security hierarchical key management scheme and secure group communications [6, 7, 8, 10, 17, 18]. Almost all of them have major disadvantages. They can not support efficiently the dynamic access control problems. Furthermore, in some cases they have hard area resources requirements, for the public parameters storage.

In 2004 Cungang Yang et. al. [19] proposed a flexible and efficient solution to access control in hierarchy is based on the one-way hash functions. Today hash functions are widely spread and are applied to many different cryptographic applications [14]. In this work, a cryptographic key management solution in the hierarchy is proposed. Each key of a node is calculated through one-way hash functions [3] locally. Comparisons with previous works shows that this scheme provides a more efficient method to deal with the dynamic access control problems such as adding/deleting nodes, or modifying relationships between nodes in the network hierarchy. Furthermore, this key management scheme uses less storage resources compared with previous introduced works.

In this paper, an efficient architecture and a VLSI implementation are presented for the key management protocol. According to our knowledge this is the first work on hardware integration, concerning this certain key management protocol regarding to network hierarchy. Different hash functions has been implemented and examined in the terms of performance and allocated area resources. An FPGA device has been used as a hardware integration platform for the proposed system architecture. Finally, results concerning performance are presented for the dynamic access control.

This paper is organized as follows: in Section 2, the philosophy behind key management scheme is described. In Section 3, the proposed system architecture for the hardware implementation is presented. In the next Section 4, the dynamic access control of the system are presented and analyzed concerning performance terms. Finally, conclusions and outlook are given in Section 5.

2 Access Control in Hierarchy: Key Management Method

The solution of Cungang Yang et. al. [19] for access control in hierarchy is based on the one-way hash functions. According to this work, for a given role of hierarchy a set of one-way hash functions are chosen: $H_i = \{H_1, H_2, \cdots, H_n\}$. With n is denoted the maximum number of the hierarchy's child nodes.

As the one-way hash function well-known hash functions could be selected such as MD5, SHA-1, SHA-2, and RIPEMD [4, 5, 11, 12, 16].

In such a hierarchy, a node is defined as a dead-end node if there are no direct parent nodes, while the other nodes are called non-dead-end nodes. In Figure 1 an example of such a hierarchy is illustrated. In the following example, node A is a dead-end node, while the nodes $B, C, \cdots, I$ are non-dead-end nodes.

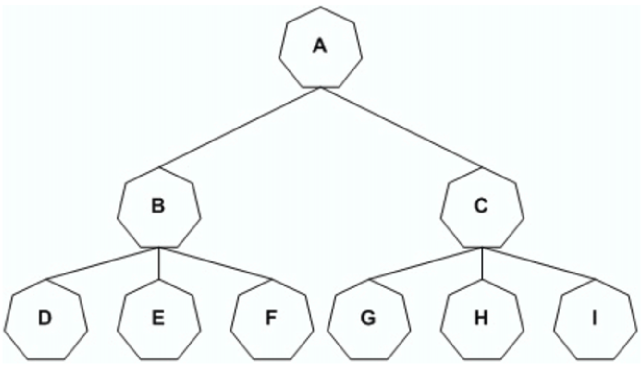


Figure 1: Hierarchy nodes

A central authority (CA) is responsible for the key generation, of nodes hierarchy. Figure 2 presents the derived keys for each node of the hierarchy. $Key(1)$ is an arbitrary key assigned by the CA to the dead-end node A. If a node r_j only has one direct parent node whose key is Key and if node r_j is the i^{th} direct child node of its direct parent node (from left to right), then the key of role r_j will be $H_i\{Key\}$.

In the case that the a node r_j has more than one direct parent nodes (r_j^1, r_j^2, $\cdots$, r_j^m) and assuming that r_j, is the j^{th} direct child node of its left most direct parent node r_j^1. In addition, if r_j is the k^{th} direct child node of r_j^2, r_j is the n^{th} direct child node of r_j^m, and the keys of the direct parent nodes of r_j (r_j^1, r_j^2, $\cdots$, r_j^m) are $Key_1, Key_2, \cdots, Key_m$, then the key of node r_j will be $H_i\{H_i(Key_1), H_k(Key_2), \cdots, H_n(Key_m)\}$.

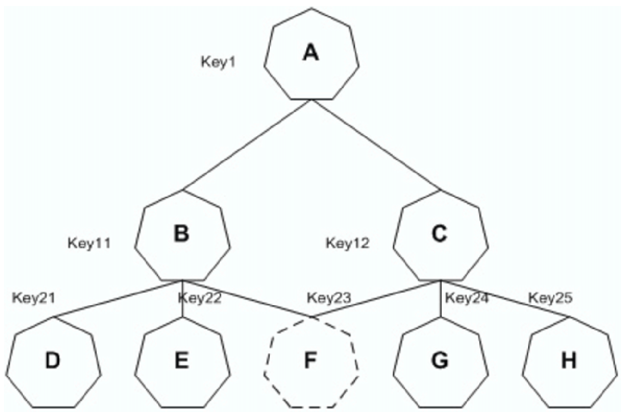


Figure 2: Keys derivation for the hierarchy node

In the above example, three different hash functions H_1, H_2, H_3 must be selected, since the maximum number of child nodes for this hierarchy is three.

As it has also be mentioned before for the dead-end node A the key Key_1 is assigned directly form the CA. Thus, the derived keys of the other nodes are generated based on Key_1 and the three selected hash functions with the following way:

- For the node that only has one direct parent node, such as B, they derived key Key_{11}, will be $H_1(Key_{11})$.

- Nodes with more than one direct parents nodes, (F), the key Key_{23} will be $H_3(H_3(Key_{11}), H_1(Key_{12}))$.

In order to derive the key of node F from node B or C, public parameters must be specified. The users of node C must know the value of $H_3(Key_{11})$, and users of node B must know the value of $H_1(Key_{12})$. This means in practical that the nodes that have more than one direct parent node should have public parameters.

The security of this management method is relied on the security of the hash functions set. In other words, this means that the users of a node cannot derive the keys of each parent nodes.

Furthermore, the user of a node A, which is one of the direct parent nodes of a node R, may have some public parameters, but the user cannot obtain the keys of other direct parent nodes of node B. The only known information is the hash values of those keys. In the example of Figure 2, the user of node B can retrieve the public parameter $H_1(Key_{12})$ from node C, but it can not derive the key of node C, Key_{12}, since hash functions values can not be reversed calculated.

3 Key Management System Implementation

3.1 Proposed System Architecture

As it is has been stated in the previous sections the key management protocol for the access control in hierarchy, is fundamentally based on one-way hash functions processing. In this section, we propose an efficient architecture and present also the hardware integration, for the management protocol. In Figure 3, the proposed architecture is illustrated.

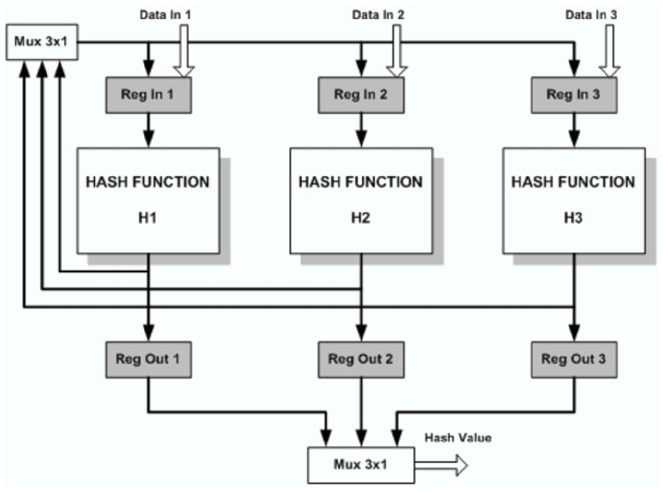


Figure 3: System architecture

We assume that in the studied hierarchy, the maximum number of direct child nodes for each node is three. This results to system architecture with three basic hash functions cores. It has to be mentioned that this is only an assumption in order to provide implementation comparisons results for a specified number of child nodes. If the cores chain is employed not different hash functions, there is no restriction for a specified number of cores (child nodes).

For a child node for which is dedicated the hash function core H_i, the input key from it's direct parent(s) is forced to the hash function core through the Data In i, with $i = 1, 2, 3$. After the appropriate number of transformation rounds, which are defined each time by the used hash function the hash value is stored to the Reg Out i. The multiplexer of the final stage Mux 3x1, defines each time which store hash value would be loaded to the Hash Value Output.

In the cases that previous hash values parent(s) keys are needed for the calculation of a child key, these could be loaded through the Reg Out (1 to 3) or even through the three data inputs.

It has to be mentioned that for the proposed system implementation for each node a n-bit register, or memory cells are needed for key storage reasons. With the variable n is defined the width of the used hash valued each time. Typical values of this variable are 160 for SHA-1, 256, 384, and 512 for SHA-2, 128 for MD5 etc.

For the proposed system it is efficient to implement the same hash function in each one of the three hash function cores. The other available solution is to select a combination of hash functions, a different for each core, in order to advance the system performance in terms of throughput, frequency, as well as and for the allocated area resources.

In this work different hash functions has been implemented in order to choose the best between the available efficient solutions each time, concerning throughput terms. The studied hash functions are, MD5, RIPEMD, SHA-1, and SHA-2. For the examined implementations a typical full rolling architecture was used, which is presented in the next Figure 4.

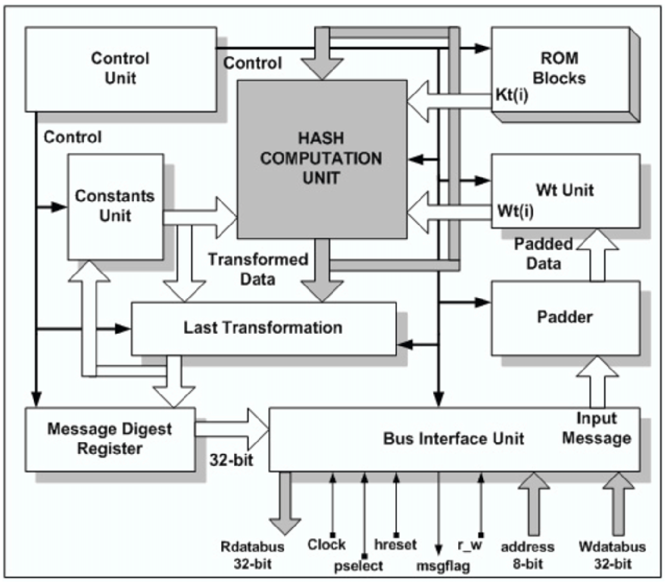


Figure 4: Hash function core architecture H_i

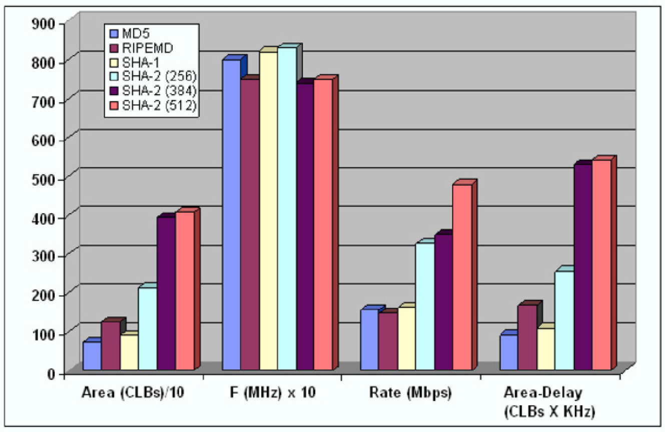


Figure 5: Implementations comparison graph

During initialization phase, the user with the appropriate write commands selects the operation mode. First,

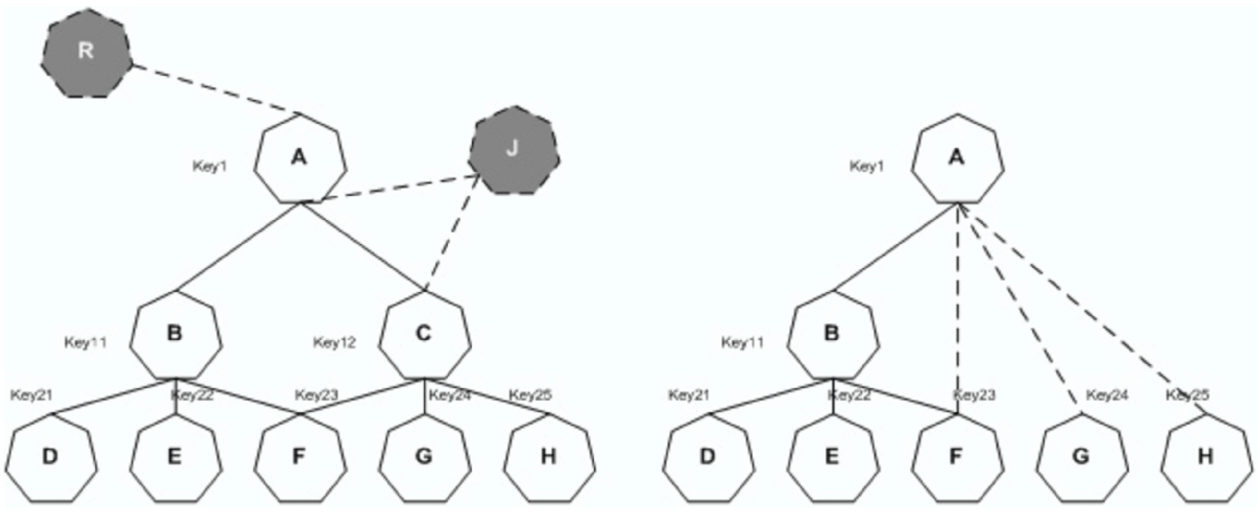


Figure 6: a) Adding a dead-end (R) and a non-dead-end (J), b) Deleting a non-dead-end (C)

the Padder pads the input message and after that the hash computation begins. The Control Unit coordinates all the system operations and processes. After the initialization phase, the control unit is totally responsible for the system operation. It defines the proper constants and operation word length, it manages the ROM blocks and it controls all the proper algebraic and digital logic functions for the hash function operation.

The Hash Computation Unit is the main datapath component of the system architecture. The specified number of the data transformation rounds, for each one of the hash functions, is performed in this component with the support of a rolling loop (feedback). In every data transformation round, based on the padded data, in the Wt Unit a new data block, $Wt(i)$, is produced. In the ROM Blocks the specified constants set, $Kt(i)$, of the hash function are stored, in order to support the Hash Computation Unit process.

The Transformed Data are finally modified in the Last Transformation, which operates in cooperation with the Constants Unit. In this way, the message digest is produced and is stored into the Message Digest Register. A Bus Interface Unit has also been integrated, in order for the proposed system to communicate efficiently with the external environment.

3.2 Implementation Synthesis Results

The proposed system architecture (Figures 3-4) was captured by using VHDL, with structural description logic. The code was synthesized, placed, and routed using an FPGA device of XILINX (Virtex) [9]. The system then was simulated again, and verified considering real time operating conditions. The tools that were used for the above procedures are: ModelSim SE/EE Plus 5.4e for simulation, Leonardo Spectrum 2003 for synthesis and XILINX Foundation 3.1i for placing and routing. The synthesis results of the proposed system are shown in Table 1.

Based on the above synthesis results the comparison graph of the studied hash functions MD5, SHA-1, SHA-2, and RIPEMD is illustrated in Figure 5.

Based on the above implementation synthesis results it is proven that SHA-1 is a flexible and efficient solution for the key agreement protocol implementation. This hash function needs less area resources. The operation frequency reaches quite high value, while using the Area-Delay model SHA-1 is one of the best implementations, compared with the others.

4 Dynamic Access Control

The proposed system operates efficiently to dynamic access control procedures, which deals with changes in the relationships between the nodes of the hierarchy. These procedures include adding, deleting and changing a node.

4.1 Adding Nodes

Adding a mode R is a simple procedure in the case of dead-end node. In this case, CA will assign a new key to R. The keys for the direct and indirect child nodes of node R must be regenerated (Figure 6a). In the case that the added node J is non-dead-end, its key will be derived from the direct parents' nodes. In addition to this, the keys for both direct and indirect child nodes of J have also to be regenerated (Figure 6a).

4.2 Deleting a Node

In the cases that a dead-end node R has to be deleted and a new dead-end node exist, the CA keeps the original key values of the last one. If the deleted node is not a dead-end node (as an example we use the node C of Figure 6b), the key values of the child nodes have to be regenerated (nodes F, G, & H) in our example.

Table 1: Implementations synthesis results

HASH FUNCTIONS	Area (CLBs)	Frequency (MHz)	Throughput (Mbps)
MD5	728	80	157
RIPEMD	1251	75	148
SHA-1	892	82	161
SHA-2 (256)	2123	83	326
SHA-2 (384)	3934	74	350
SHA-2 (512)	4075	75	480

4.3 Changing Nodes Relationship

Another basic process is related to the relationship change between the nodes. For a node R which is the direct parent of another node of the system, named S, if node S becomes a dead-end node after deleting the relationship between R and S the CA will change the key of S. In different case, the keys of both S and its direct and indirect child nodes will be regenerated.

Figure 5 presents such a scenario of changing nodes relationships. If the relationship between A and B stops to exist, they key of B will not be regenerated. If the relationship between C and F is deleted, the key of node F will be generated again. In addition, if a new relationship between two nodes is created, for instance between S and R, and S become a direct parent of the node R, both the keys of R and the child nodes have to be regenerated. In other words, a new relationship between B and H as it is shown in Figure 6, with B to become the parent of node H, results to the regeneration of node H.

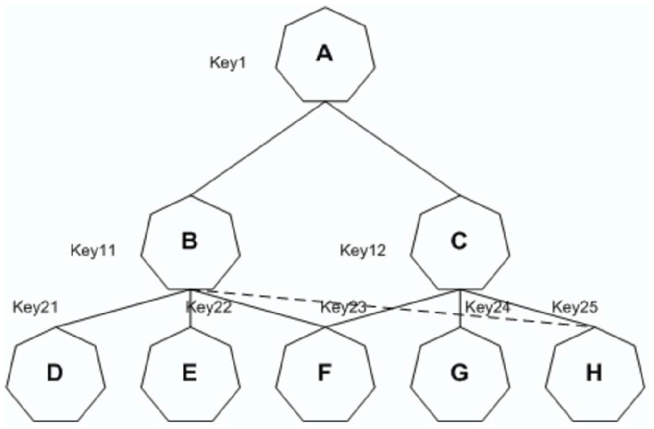


Figure 7: Changing nodes relationship example

By adding deleting or changing the relationships between the nodes of the network hierarchy it is resulted to additional hash function processing. This costs to an additional number of clock cycles, which are depended on the selected hash function each time. The additional cost for each one of the studied hash functions in the term of clock cycles per hash calculation, are given in Figure 8.

The above graph shows that SHA-2(256) and MD5 could be selected as the hash functions with less performance cost, measured as clock cycles per hash computation, in the case of changing nodes relationships. Although, the area cost of such an implementation is quite high for SHA-2, compared with the others (Figure 5). Based on both comparisons results of both Table 1 and Figure 8, one flexible solution is the selection of both SHA-2(256) and MD5 for the cores of the proposed system of Figure 3. Both of them ensures lowest additional performance cost, in the case of dynamic access control. The selection of SHA-1 for the hash functions cores supports a good balance concerning the area resources against the SHA-2(256) or MD5. At the same time, SHA-1 is a worthwhile implementation for the initial key generation of the hierarchy nodes.

5 Conclusions and Outlook

In this work, the implementation of a key management method is examined. This protocol is used basically for the access control in a network hierarchy. It is based on the usage of hash functions, which guarantees the security level of the system. This paper studies the alternative implementations of different hash for this key agreement protocol. More analytically, by using a full rolling architecture as an integration platform the implementation of the following hash functions are examined concerning both performance and area implementation cost terms: MD5, RIPEMD, SHA-1, and SHA-2. Finally, the dynamic access control of the system is presented. The performance additional cost is given for the operations of adding, deleting and changing the relationships of a node(s).

References

[1] S. G. Akl and P. D. Taylor, “Cryptographic solution to a multilevel security problem,” Chaum D, Rivest RL, Sherman AT, editors, in em Advances in Cryptology, 1982.

[2] S. G. Akl and P. D. Taylor, “Cryptographic solution to a problem of access control in a hierarchy,” *ACM Transaction on Computer Systems*, vol. 1, no. 3, pp. 239–248, July 1983.

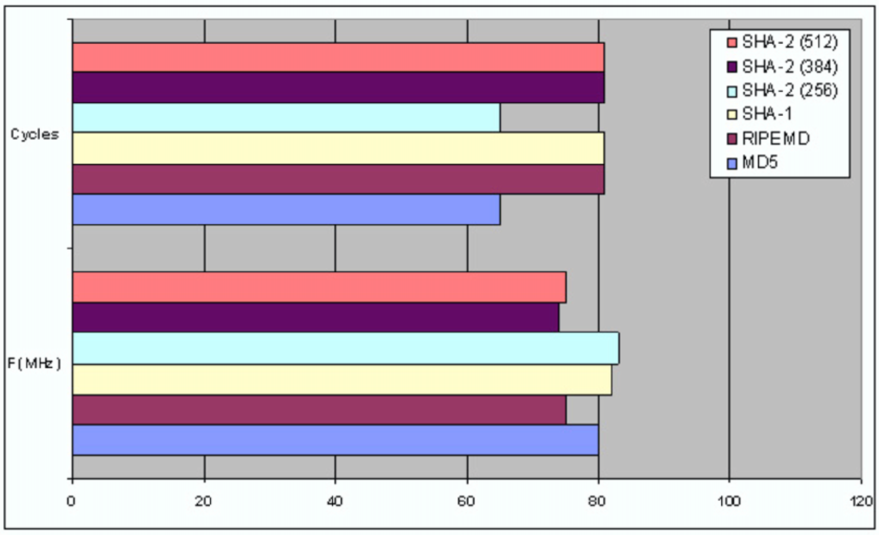


Figure 8: Additional cost, clock cycles per hash computation

[3] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk, *Cryptographic Hash Functions: A Survey*, Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.

[4] S. Bakhtiari, R.Safavi-Naini, J. Pieprzyk, *Cryptographic Hash Functions: A Survey*, Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.

[5] H. Dobbertin, A. Bosselaers, and B. Preenel, "RIPEMD-160, a strengthened version of RIPEMD," in em Fast Software Encryption, LNCS 1039, Springer-Verlag, pp. 71–82, 1996.

[6] H. (Nokia) Frederick, J. Mike, *XML Key Management (XKMS 2.0) Requirements*, W3C Note 05-May-2003.

[7] T. Hardjono, B. Cain B, I. Monga, *Intra-domain Group Key Management Protocol*, Internet-Draft, draft-ietf-ipsec-intragjm-00.txt, Nov. 1998.

[8] J. H. Huang, M. S. Mishra, "A highly scalable key distribution protocol for large group multicast," in *proceedings of IEEE Global Communications Conference (GLOBECOM 2003)*, pp. 232–235, 2003.

[9] S. Jose, *Virtex, 2.5 V Field Programmable Gate Arrays*, Xilinx, California, USA, www.xilinx.com, 2005.

[10] V. Manish, "XML security: the XML key management specification," XKMS helps make security manageable: IBM developer Works, Jan. 27, 2004.

[11] SHA-1 Standard, National Institute of Standards and Technology (NIST), *Secure Hash Standard*, FIPS PUB 180-1, www.itl.nist.gov/fipspubs/fip180-1.htm, 2005.

[12] SHA-2 Standard, National Institute of Standards and Technology (NIST), *Secure Hash Standard*, FIPS PUB 180-2, www.itl.nist.gov/fipspubs/fip180-2.htm, 2005.

[13] B. Schneier, *Applied Cryptography-Protocols, Algorithms and Source Code in C*, Second Edition, John Wiley and Sons, New York, 1996.

[14] N. Sklavos, P. Kitsos, K. Papadomanolakis and O. Koufopavlou, "Random number generator architecture and VLSI implementation," in *proceedings of IEEE International Symposium on Circuits & Systems (ISCAS'02)*, pp. 854–857, USA, 2002.

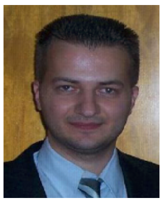
[15] N. Sklavos and O. Koufopavlou, "Mobile communications world: security implementations aspects - A state of the art," *CSJM Journal: Institute of Mathematics and Computer Science*, vol. 11, no. 2 (32), pp. 168–187, 2003.

[16] D. R. Stinson, *Cryptography: Theory and Practice*, CRC Press LLC, 1995.

[17] C. K. Wong, M. Gouda, S. S. Lam, "Secure group communications using key graphs," in *proceedings of ACM SIGCOMM'98*, pp. 69–72, 1998.

[18] T. C. Wu, T. S. Wu, W. H. He, "Dynamic access control scheme based on the Chinese remainder theorem," *Computer Systems: Science and Engineering*, vol. 2, pp. 92–99, 1995.

[19] C. Yang and C. Li, "Access control in a hierarchy using one-way hash functions," *Computers & Security*, vol. 23, no. 8, pp. 659–664, 2004.



Nicolas Sklavos received the Ph.D. Degree in Electrical & Computer Engineering, and the Diploma in Electrical & Computer Engineering, in 2004 and in 2000 respectively, both from the Electrical & Computer Engineering Dept., University of Patras, Greece. His research interests include Cryptography, Wireless Communications Security, Computer Networks and VLSI Design. He holds an award for his PhD thesis on "VLSI Designs of Wireless Communications Security Systems", from IFIP VLSI SOC 2003. He has participated to international journals and conferences organization, as Program Committee Member and Guest Editor. Dr. N. Sklavos is a member of the IEEE, the Technical Chamber of Greece, and the Greek Electrical Engineering Society. He has authored or co-authored more than 80 scientific articles, books chapters, tutorials and reports, in the areas of his research. Contact him at: nsklavos@ieee.gr.

Odysseas Koufopavlou received the Diploma of Electrical Engineering in 1983 and the Ph.D. degree in Electrical Engineering in 1990, both from University of Patras, Greece. From 1990 to 1994 he was at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Patras. His research interests include VLSI, low power design, VLSI crypto systems, and high performance communication subsystems architecture and implementation. Dr. Koufopavlou has published more than 100 technical papers and received patents and inventions in these areas.