14th IEEE International Conference on Electronics, Circuits and Systems (IEEE ICECS'07),
Marrakech, Morocco, December 11-14, 2007.

SecurID Authenticator: On the Hardware Implementation Efficiency

Nicolas Sklavos

University of Patras,
Patras, Greece
email: nsklavos@ieee.org

Costas Efstathiou

Informatics Dept.,
Technological Educational Institute of Athens,
Athens, Greece
email: cefsta@teiath.gr

***Abstract*—SecurID tokens have been developed by SDTI/RSA Security to authenticate users to a corporate network infrastructure. Although, thus authorization approach could be efficiently applied to modern networks such as wireless and mobile. In this work, two different architectures and their integrations on silicon, are proposed for SecurID Authenticator. Both proposed designs are compared with the conventional one, in terms of operating frequency, allocated area resources, and throughput values. More analytically, the paper presents two alternative design solutions and efficient implementations for SecurID Authenticator: the first is suitable for high speed communications, while the second supports minimized area applications. Both proposed architectures could be used alternatively to communication networks, in order to ensure high level of security, with special needs for high speed or small area resources, respectively.**

I. INTRODUCTION

Security and privacy [1] are two of the main topics, that researchers have centered their interest the last years. Secured communications prove to be a matter of great importance day by day, and affect the most aspects of electronic and networking data transfers, via communication devices [2].

Wireless networks security [3] is proven a very sensitive area, in the provision of electronic services. Before a subscriber or a user can access services in a administrative domain, first he registers with his ID (Identification Number) to take permission of a specific sector of provided services. In other words, the system or network processes first the authorization phase, in order the usage of certain applications to be allowed.

Authorization defines the process of verifying object's permission to perform a particular action or not. Two are the main alternative function mechanisms that are applied mainly for this purpose:

- *authentication-based schemes require, as a precognition, object authentication, which is utilized to check via Access Control Lists (ACL), whether this identified object is allowed to perform the requested actions.*
- *credential-based schemes, which apply credentials with trustworthy information, provided by the algorithm performing the authorization process.*

SecurID denotes an authentication scheme, developed by SDTI/RSA, which now is known as RSA Security [4]. It is basically used as a hand held authorization device, which serves mainly workers in companies. The main idea behind this security mechanism is the combination of a pseudorandom token code, with the pin of the owner, in order the subscriber to gain access in a workstation, and finally in the network infrastructure of the company. This process cyclically works every a specified short period of time: for example every minute or every second a new pseudorandom code is generated. John Brainard, the SecurID developer, uses a SecurID function in order to achieve the requested secure level of the scheme. Although, SDTI/RSA has never made this function public, the source code of an alleged SecurID was published in the web, as a product of reverse engineering [5].

This paper proposes two alternative architectures, for the hardware implementation of SecurID authentication scheme. Both of them integrate this specific authentication scheme, in all terms of specified operation. The first proposed system architecture, named MinSecurID, is a compact design and utilizes the applications with minimized available area resources. This is achieved due to the feedback loop, of transformation data. The second proposed system architecture, called HighSecurID, achieves high speed performance. It is based on a pipeline design technique, which supports parallel transformation of four data blocks at the same time. Both proposed designs could be integrated in FPGA and ASIC hardware devices. In order to have a fair and detailed comparison of the proposed architectures, a conventional design is also presented and could also be integrated with the usage of the same type hardware devices.

The proposed MINSecurID architecture is proven superior to the conventional architecture and to the HIGHSecurID proposed architecture at about 40-45%, and 45-50% respectively. HIGHSecurID on the other hand, performs 4 times better than conventional architecture and the MINSecurID. This high value of throughput is achieved due to the applied design methodology of MINSecurID.

This paper is organized as follows: In Section 1 an introduction to the focused topic is given. In Section 2 the Conventional Architecture of the SecurID is presented in detail. Sections 3 & 4, have been dedicated to the proposed system architectures. Detailed comparison results are given in the next Section 5, between the conventional architecture and the two proposed systems. The paper ends with the conclusions and outlook, given in Section 6.

II. CONVENTIONAL ARCHITECTURE

The conventional architecture for SecurID hardware implementation is presented in the following Figure 1.

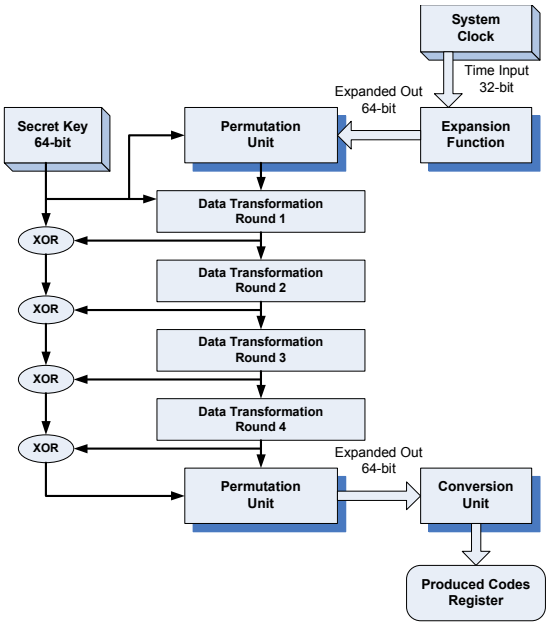

Figure 1. Conventional Architecture

The system operation starts with the time input. This input is a 4-byte (32-bit) vector which represents the time in minutes between the time of systems' startup and he time Jan, 01, 1986 0.00 GMT. The Expansion Function uses the three least significant bytes (B1, B2, B3) of time input (B0B1B2B3), and produces the output of the unit equal to B1B2B3B3B1B2B3B3, which is a 64-bit vector.

The Expansion Function could be designed as a 64-bit register and the appropriate data transformation is a matter of wiring between input and outputs, with no additional cost.

Before the major data transformation and a step previous to systems' final output, Permutation Unit roles as both initialization and final process. Permutation Unit produces a 64-bit output Y and works as it is described bellow. The secret key is split into 16 vectors named K0,…,K15, while Permutation Out defines an 64-bit array called u. A used flag indicates the u(K0). The four previous bits of u(K0) are the 4-bit output part $Y_{60}Y_{61}Y_{62}Y_{63}$. These 4 bits are removed from the array, while the pointer is increased, modulo the size of the array, by the next key vector K1. Once again the four previous bits of pointed bit are equal to the values of the next 4 bits of Permutation Unit output $Y_{56}Y_{57}Y_{58}Y_{59}$.

This process is continued in the same way up to the 16 vectors of secret key, until K15 and in this way the output of the expansion function is determined

As it is shown in the above Figure 1, the data transformation is centered in four Data Transformation Rounds (1 to 4). The main rounds use as inputs a 64-bit data value and a 64-bit key (Key1 to Key4). The KeyN for each round is given by XORing the key value of the previous round key Key(N-1), with the 64-bit data output of the previous round also.

Every round consists of 64 subrounds. Each subround transformation uses one bit of the round key, and transforms the data, based on two different functions called R and S. Each subround i, with i=1...64, transforms the input data vector B(i-1) to B(i). Each data function S, R is used alternatively in subrounds transformations, according to the comparison of the values between KeyN(i-1) and the round's 64-bit data vector input B(i-1):

- *In the case that these two values are equal, R function is the data transformation core, while S function stays idle.*
- *In case that the above values are different S is transforming data, while R does not operate.*

When the 64 subrounds operation have been completed in the above described way, the data output B64 of a round has been successfully produced. S and R functions operation could be described as follows:

Function R:

$$B_{OUT0}=((((B_{IN1}(i-1)>>>1)-1)>>>1)-1)\oplus B_{IN4}(i-1) \quad (1)$$
$$B_{OUTj}(i)=B_{INj}(i-1) \text{ for } j=1,2,3,4,5,6,7. \quad (2)$$

Function S:

$$B_{OUT0}(i)=B_{IN4}(i-1) \quad (1)$$
$$B_{OUT4}(i)=100-B_{IN0}(i-1) \bmod 256 \quad (2)$$
$$B_{OUTj}(i)=B_{INj}(i-1) \text{ for } j=1,2,3, ,5,6,7. \quad (3)$$

For the SecurID hardware integration the conventional architecture could be designed, based one two separated units for data transformation functions S, and R. These two separated units' designs are shown in the following Figure 2:

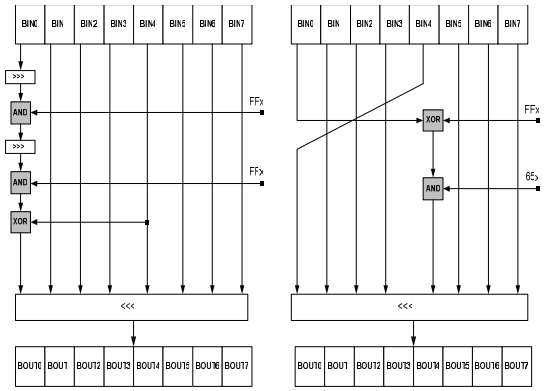

Figure 2. Architectures for R & S Subrounds Functions

Both functions R and S divide the 64-bit input to eight parts of 8-bit. These functions basically forward the inputs to the outputs, after certain simple steps of modification. Data transformation in both functions includes cyclical shift to the right, >>>, or to the left, <<<, by a constant value of positions, one in both functions. This process could be integrated with the use of an 8-bit multiplexer. Other processes that are used for data transformation is XORing denoted as $\oplus$ and subtraction with 01_{hex}, which could be implemented as addition with FF_{hex} .

III. PROPOSED ARCHITECTURE MINSECURID

Two different architectures are proposed for the SecurID authentication scheme. The first one, called MinSecurID minimizes the requested allocated area and it is suitable in applications with low available resources. The proposed MinSecurID architecture is shown in the next Figure 3.

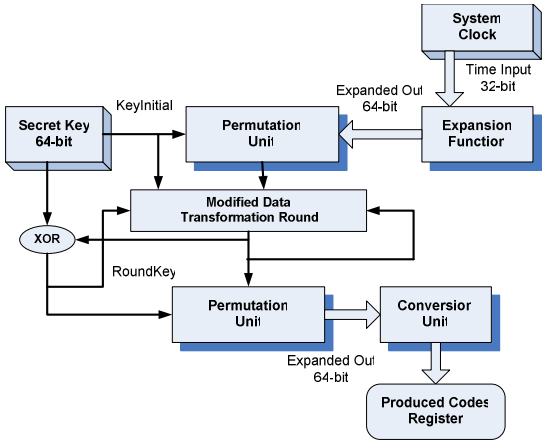

Figure 3. Proposed MinSecurID Architecture

Compared with the conventional architecture, MinSecurID is based on a Modified Data Transformation Round which operates as all the requested 4 Data Modifications Rounds.

The appropriate output of each Round(i) operation is used as an input for the next Round(i+1), by the use of feedback loop technique. For this reason a 64-bit bus is used for the appropriate data transportation.

Eternal to the Modified Data Transformation Round, a alternative approach of designing two different architectures for R, S is used. Instead, of the two different functions of S and R (Fig. 2), a MixSR Function is used. Figure 4 shows the architecture of MixSR.

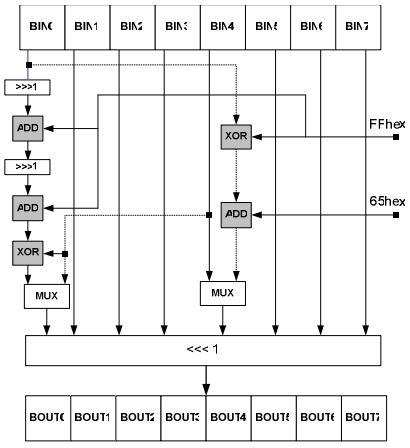

Figure 4. Proposed MixSR Architecture

MixSR operates as original S and R according to the control bits of key round values. The design of this architecture is based on the usage of a set of multiplexers, MUX, which define different data paths for the outputs BOUT0 and BOU1.

IV. PROPOSED ARCHITECTURE HIGHSECURID

For applications with special needs of speed, which are concluded to high values of throughput and operating frequency also another architecture is proposed. This is called HighSecurID and it is illustrated in Figure 5.

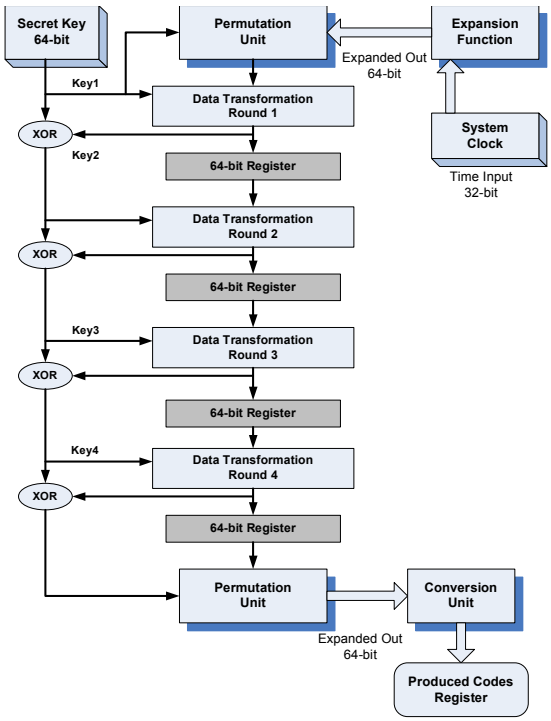

Figure 5. Proposed HighSecurID Architecture

HighSecurID uses a chain of Data Transformation Rounds and 64-bit registers. In this way the output data of each round are temporarily stored. In each clock cycle 4x64-bits could be transformed at the same time. A new set of produced codes is generated also every clock cycle.

Other units such as Expansion Function, Conversion Unit and Permutation Unit are the same with compatible architecture and they are operating in a similar way, according to SecurID specifications.

It has to be mentioned that for the 4 Data Transformation Rounds, HighSecurID uses the proposed architecture of MixSR Unit (Figure 4). In this way, an area reduction is also achieved for the HighSecurID architecture also.

V. COMPARISONS: AREA, TIME, SPEED

Regarding area resources conventional architecture uses almost the same resources with HighSecurID. The second one, uses more 4 x registers of 64-bit, as well as 4 x 2 multiplexers of 8-bit each one.

MinSecurID uses 3 less Data Transformation Rounds than conventional and HighSecurID also, due to the applied feedback loop design (Fig. 3).

All the three examined architectures use also a number of common units such as: Key Unit, Expansion Function, Permutation Unit and Conversion Unit.

Table I presents analytically the allocated area resources of each one architecture, in terms of used components.

TABLE I.
AREA COMPARISON

UNITS & COMPONENTS	ARCHITECTURES		
	Compatible	Proposed MinSecurID	Proposed HighSecurID
Secret Key	I	I	I
XOR Blocks	4	I	4
Permutation Unit	I	I	I
Expansion Function	I	I	I
Conversion Unit	I	I	I
Data Transformation Round	4	I	4
Extra	-	8 x 8-bit MUXs	4 x 64 Registers 8 x 8-bit MUXs

Both proposed architectures, as well as conventional one could be integrated with software and also hardware integration platforms, such as FPGAs of Xilinx [6]. Although, for the integration of encryption algorithms and privacy schemes, for a great number of reasons, regarding both security matters and performance issues, hardware implementations are mostly preferred.

Based on the above implementation results proposed MINSecurID architecture is proven superior to the compatible architecture and to the HIGHSecurID proposed architecture at about 40-45%, 45-50% respectively, due to the reduced number of used Data Transformation Rounds (one instead of 4). It has to be mentioned that Data Transformation Round is the most critical component regarding area cost. The following Figure 6 illustrates the comparisons results of the three examined architectures, concerning allocated resources terms of integration.

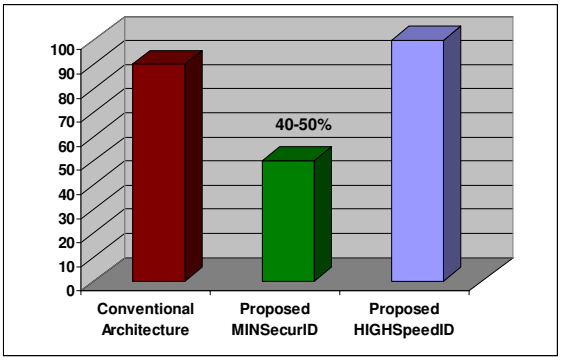

Figure 6. Comparisons Results – Area Terms

HIGHSecurID on the other hand, performs 4 times better than conventional architecture and MINSecurID one. This high value of throughput is achieved based on the applied design methodology of MINSecurID, which supports the data transformation of 4 data blocks at the clock cycle, instead of the one per clock cycle that both conventional and MINSecurID transform.

Regarding the operation frequency of the proposed designs, MINSecurID has almost the same time critical path with the HIGHSecurID, equal at about to one transformation round total delay. This consults to the almost the same operation frequency values for both proposed designs. Conventional architecture has time critical path four times worst than the two proposed designs. This is due to the fact that the time critical path of conventional designs covers the data transformation of four rounds at once. In HIGHSecurID design, such delay is avoided based on the cascaded registers chain, which has been applied. MINSecurID has a short time critical delay path, equal to one round data transformation, due to the used feedback logic technique. In Figure 7, detailed time comparison results are illustrated, regarding frequency and throughput values.

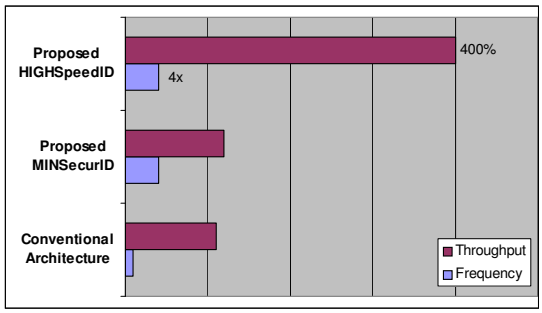

Figure 7. Comparisons Results – Time and Performance Terms

VI. CONCLUSIONS & OUTLOOK

Security is a primary issue of today's communication networks, in the provision of the subscribed services. Authentication is a very critical process, prior to the use of the services for each user. This work proposes two alternative architectures, efficient for hardware implementations for SecurID authentication schemes. In addition, the design of the conventional architecture is also presented. The first performs efficiently in cases of low available area resources, while the second is superior regarding the performance values. Each one of the proposed architectures could be used respectively, according to area or performance issues criteria.

ACKNOWLEDGMENT

This work is co-funded by 75% from the E.E., and 25% from the Greek Government under the framework of the Education and Initial Vocational Training Program-Archimedes.

REFERENCES

[1] Bruce Schneier, *Applied Cryptography – Protocols, Algorithms and Source Code in C*, Second Edition, John Wiley and Sons, New York, 1996.

[2] N. Sklavos, O. Koufopavlou, "Mobile Communications World: Security Implementations Aspects - A State of the Art", CSJM Journal, Institute of Mathematics and Computer Science, Vol. 11, Number 2 (32), pp. 168-187, 2003.

[3] N. Sklavos, X. Zhang, *Handbook of Wireless Security: From Specifications to Implementations*, CRC-Press, A Taylor & Francis Group, ISBN: 084938771X, 2007.

[4] RSA Security Website, SecurID, 2006.

[5] IC. Wiener, "Sample SecurID token emulator with token secret import," post to BugTraq, http://archives.neohapsis.com/archives/bugtraq/2000-12/0428.html, 2000.

[6] Xilinx 2007, www.xilinx.com.