WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2007; **7**:483–494
Published online 6 July 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.367

UMTS security: system architecture and hardware implementation

P. Kitsos*,†, N. Sklavos and O. Koufopavlou

VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras, Patras, Greece

Summary

Universal mobile telecommunication system (UMTS) has specified security mechanisms with extra features compared to the security mechanisms of previous mobile communication systems (GSM, DECT). A hardware implementation of the UMTS security mechanism is presented in this paper. The proposed VLSI system supports the Authentication and Key Agreement procedure (AKA), the data confidentiality procedure, and the integrity protection procedure. The AKA procedure is based on RIJNDAEL Block Cipher. An efficient RIJNDAEL architecture is proposed in order to minimize the usage of hardware resources. The proposed implementation performs the AKA procedure within 76 μs comparing with the 500 ms that UMTS specifies. The data confidentiality and the integrity protection is based on KASUMI Block Cipher. The proposed KASUMI architecture reduces the hardware resources and power consumption. It uses feedback logic and positive-negative edge-triggered pipeline in order to make the critical path shorter, without increasing the execution latency. The S-BOXes that are used from RIJNDAEL and KASUMI block ciphers have been implemented with combinational logic as well as with ROM blocks. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: UMTS security; 3GPP; RIJNDAEL; KASUMI; block cipher; negative edge-triggered pipeline; hardware implementation

1. Introduction

Universal mobile telecommunication system (*UMTS*) represents the next generation of wireless communication. UMTS has the possibility to deliver a plethora of high-value broadband information, and services to mobile users via fixed, wireless, and satellite networks in both outdoor and indoor environments.

In order to achieve efficient and secure roaming between the different networks, UMTS security supports mechanisms with additional features, as compared to the previous mobile systems (*GSM, DECT*). The encryption algorithms are stronger. The application of authentication algorithms is stricter and the subscriber confidentiality is tighter. In GSM systems, the network requires an authentication of mobile users, but the network does not authenticate itself to the user. UMTS introduces mutual authentication.

The UMTS security architecture is based on three procedures. At the beginning, the user authenticates the network, and vice versa. After, the integrity of the signaling information is required, and finally the user and signaling data must be confidentiality protected. A system architecture and hardware implementations

*Correspondence to: P. Kitsos, VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras, Patras, Greece.
†E-mail: pkitsos@ieee.org

providing these security procedures are described in this paper.

The authentication is performed by the *Authentication and Key Agreement* (*AKA*) procedure [1]. The AKA procedure is built on the *RIJNDAEL block cipher* [2]. In addition to authentication, AKA procedure also results in the *Cipher Key* (*CK*) and the *Integrity Key* (*IK*). In UMTS, only the encryption mode of the RIJNDAEL block cipher is used [3] as an iterated hash function [4]. The block and key length have been set to 128-bit.

The performance of the proposed RIJNDAEL block cipher implementation in terms of throughput is slightly slower than previous designs [5–9]. This is not a drawback since the proposed RIJNDAEL block cipher implementation produces the proper output in 12.6 μs, while UMTS specification requires 50 ms. The major advantage of the proposed implementation is the minimization of the covered area resources, which is important in applications with strict area limitations (e.g., portable systems).

The integrity of the signaling information is handled by *Message Authentication Code* [1] procedure implemented by using the *Integrity Algorithm f*9 and the Integrity Key (IK) [10]. The user and signaling data confidentiality protection is handled by the *Data Confidentiality* [1] procedure implemented by using the *Ciphering Algorithm f*8 and the Cipher Key (CK) [10].

Both *f*8 and *f*9 algorithms are based on the *KASUMI block cipher* [11,12]. An efficient implementation of the KASUMI block cipher using feedback logic and negative edge-triggered pipeline [13] is introduced. This implementation makes the critical path shorter, without increasing the latency of the cipher execution. If the clock frequency is determined by the system specifications, the usage of negative edge-triggered pipelining can reduce the clock frequency of its original value for the same data throughput. As a result, power consumption is reduced. This results in reduced power consumption and has better performance than previous designs [14–17].

S-BOXes are fundamental elements for both RIJNDAEL and KASUMI block ciphers. For each cipher, two system architectures are proposed in order to provide alternative implementations regarding the covered area and throughput. In the first architecture, the S-BOXes are implemented by using ROM blocks, while in the second one by using combinational logic.

The rest of the paper is organized as follows. In Section 2, the UMTS security architecture is briefly introduced and in Section 3, the proposed system architecture and the hardware implementations are described. In Section 4, the hardware implementation results are presented and evaluated. Finally, Section 5 concludes this paper.

2. UMTS Security Architecture

The security specifications for the UMTS have been standardized by the *3rd Generation Partnership Project* (*3GPP*). In mobile equipment, all the security tasks are integrated as shown in Figure 1.

In the *Universal Subscriber Identity Module* (*USIM*) the Authentication and Key Agreement (AKA) module is integrated. It is the most important security feature of the UMTS security system. The other two basic modules, Message Authentication Code and Data Confidentiality are integrated on the Mobile Device.

At the beginning of a new connection between two mobile users, the *Authentication Center* (*AuC*), sends to the users the appropriate authentication parameters, *Authentication Token* (*AUTN*) and *Random Challenge* (*RAND*). These parameters with the addition of the *Secret Key* (*K*) [18] are the only information that AKA module needs in order to perform the authentication procedure. The AUTN is a 176-bit value that contains three sub-values. The first is the XOR operation product (symbolized by '$\oplus$') of the *Sequence Number* (*SQN*) and the *Anonymity Key* (*AK*), the second is the *Authentication Management Field* (*AMF*) and the third is the *Message Authentication Code* (*MAC-A*).

The main task of the Authentication and Key Agreement module is the execution of the RIJNDAEL block cipher algorithm (symbolized by 'E_K'). The OPc variable is computed outside the USIM, and is stored in USIM. The two necessary parameters for this calculation are the *Operator Variant Algorithm Configuration Field* (*OP*) and the secret key K. The constants c_1, c_2, c_3, c_4, and c_5 and also the integers r_1, r_2, r_3, r_4, and r_5 that define the cyclically rotation are specified in Reference [4].

At the beginning, the received RAND and SQN $\oplus$ AK with the secret key K are used in order to produce SQN and *User Authentication Response* (*RES*) (Figure 1). Then the double concatenation (symbolized by '||') of SQN and AMF are used for the production of the *Expected Message Authentication Code* (*XMAC-A*) and the *Resynchronization Message Code* (*MAC-S*). After, XMAC-A is compared with the received MAC-A. If XMAC-A and MAC-A matches,

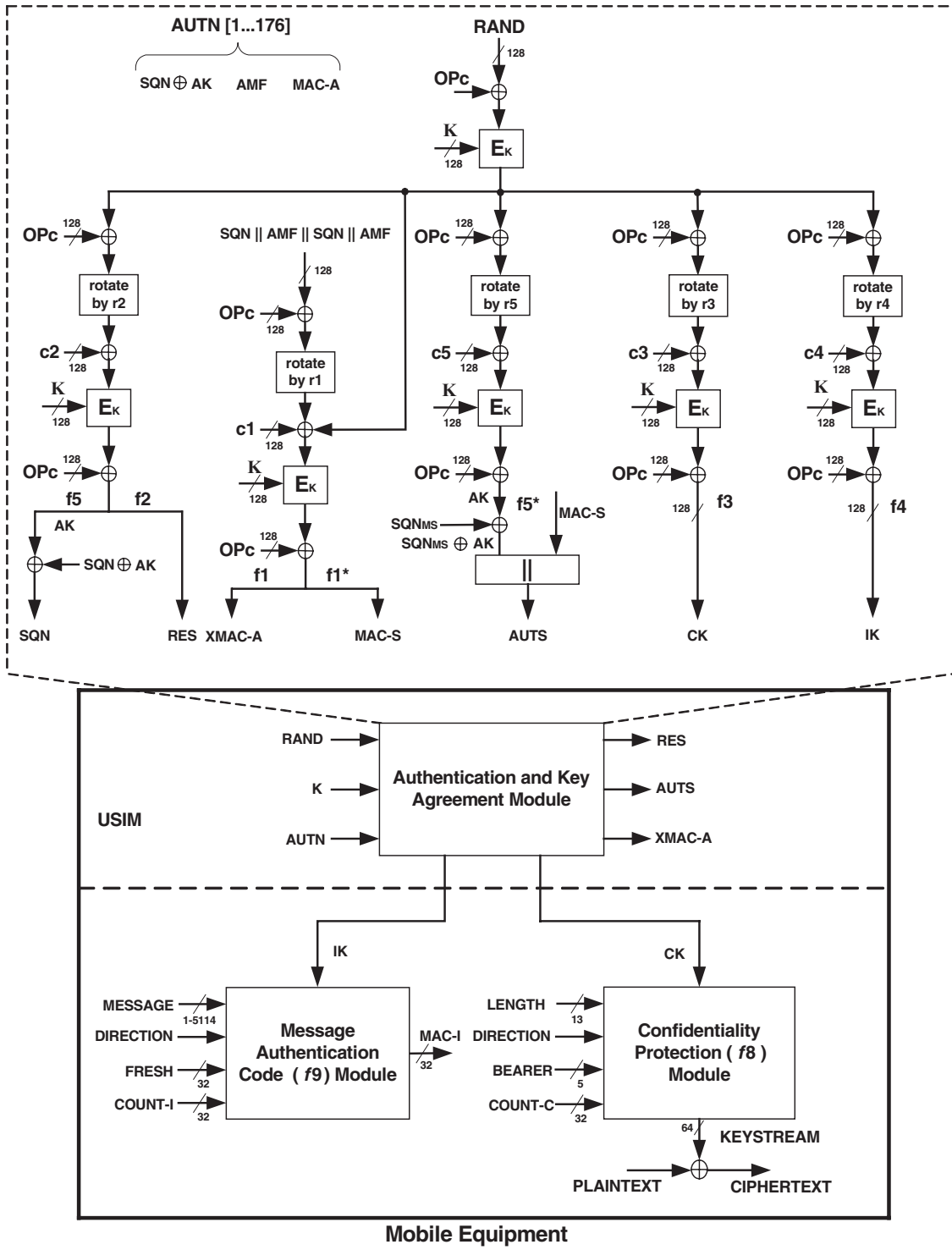

Fig. 1. UMTS security architecture.

the authentication procedure continues. If these parameters are different, a '*user authentication reject*' is sent back to the *Visitor Location Register/Serving GPRS Support Node* (*VLR/SGSN*).

In all cases, for the network authentication, USIM should verify if the produced SQN is within the correct range. If it is within the correct range, USIM continues to generate *RES*. RES is sent back to the

VLR/SGSN and it is compared with the *Expected Response XRES*. If they match, the user authentication is passed otherwise the authentication is failed. If SQN is out of the correct range, the authentication procedure is failed, and USIM generates a *Re-synchronization Token* (*AUTS*). AUTS is the concatenation of $SQN_{MS} \oplus AK$ and MAC-S. A '*synchronization failure*' message with AUTS as parameter is sent back to VLR/SGSN. Finally, the cipher key CK and the integrity key IK are generated by RAND and the secret key K.

The signaling information that is sent between the user and the network are integrity protected by the integrity algorithm $f9$. $f9$ adds a '*flag*' to messages to ensure that are generated at the claimed identity. Based on the input parameters MESSAGE, DIRECTION, FRESH, COUNT-I, and IK, the Message Authentication Code ($f9$) Module computes the *Message Authentication Code* (*MAC-I*). As mentioned above, IK is the 128-bit integrity key produced in USIM. MESSAGE is the input bit-stream and DIRECTION is a 1-bit input, which indicates the direction of transmission (uplink or downlink). FRESH is a 32-bit random number and COUNT-I is a 32-bit time variant. MAC-I is appended to the message and is sent over the radio access link. The receiver computes the corresponding *Expected Message Authentication Code* (*XMAC-I*) and verifies the data integrity of the message by comparing it to the received MAC-I.

The user data and some signaling information are considered sensitive and must be confidentiality protected. Based on the input parameters LENGTH, DIRECTION, BEARER, COUNT-C, and CK, the Confidentiality Protection ($f8$) Module computes the output key-stream block *KEYSTREAM*, which is used in order to encrypt the input *PLAINTEXT* and produce the output *CIPHERTEXT*. As mentioned above, CK is the 128-bit confidentiality key produced in USIM. LENGTH is the number of the PLAINTEXT bits and DIRECTION is a 1-bit input, which indicates the direction of transmission (uplink or downlink). BEARER is a 5-bit input and COUNT-C is a 32-bit time variant.

3. System Architecture and Hardware Implementations

For the UMTS security hardware implementation, we proposed the system architecture illustrated in Figure 2. This system is a part of a mobile equipment.

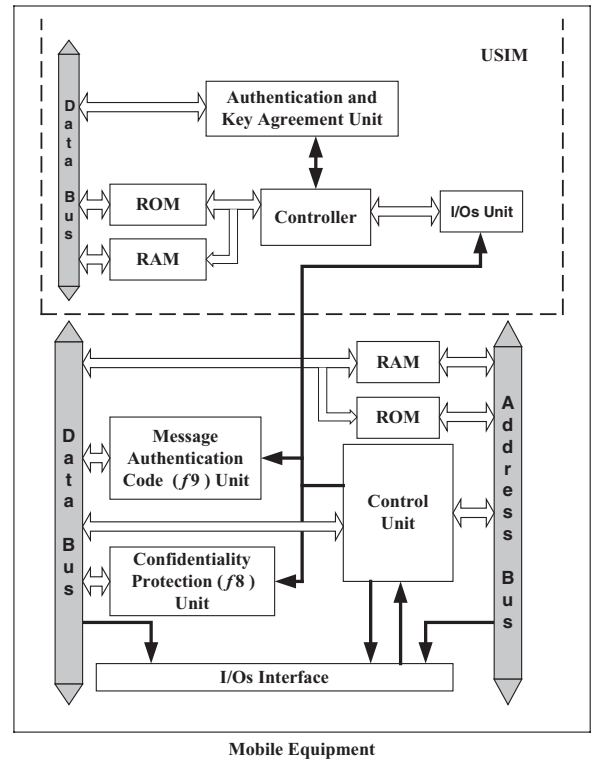

Fig. 2. UMTS security hardware system architecture.

It consists of two fragments. The first fragment is integrated in USIM and implements the Authentication and Key Agreement (AKA). The second fragment is integrated in the mobile device and implements the confidentiality of data protection ($f8$ algorithm), and the message authentication code ($f9$ algorithm).

In the USIM fragment, a data bus of 8-bit is used for all the internal data transfers. The appropriate parameters are stored in RAM, while the fixed constants *ri* and *ci* are stored in ROM. The I/Os unit is used for the USIM communication with the rest of the system. The *Controller* synchronizes the USIM operation.

The proposed system architecture is supported by a *Control Unit*, which coordinates all system operations and processes. In the mobile device fragment, a common 64-bit data bus and a 32-bit address bus are used for the internal data transfers. The algorithms appropriate keys are stored in the RAM. The required parameters of the $f8$ and $f9$ algorithms are stored in the ROM. Through the *I/Os Interface*, the proposed system transfers data from/to the external environment.

Below the three basic units of the proposed system architecture (i.e., Authentication and Key Agreement, Message Authentication Code ($f9$), and Confidentiality Protection ($f8$)) are described in details.

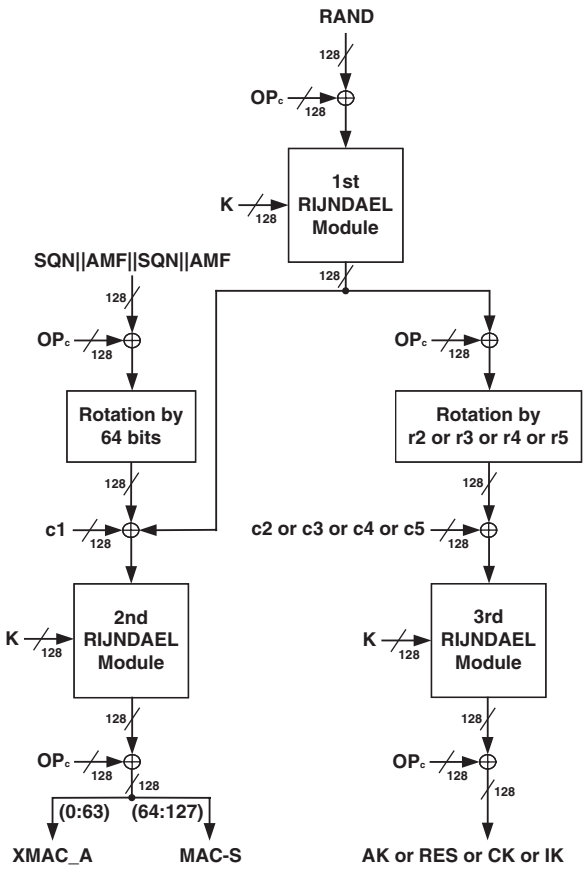

Fig. 3. The authentication and key agreement implementation.

3.1. The Authentication and Key Agreement Unit

A typical implementation of the Authentication and Key Agreement procedure needs six RIJNDAEL-modules (Figure 1). In the proposed system design (Figure 3), only three RIJNDAEL-modules are used. In this way, a significant hardware resources reduction is achieved which is very important in applications with strict area limitations. In addition, from the performance measurements' following it is obvious that the proposed architecture performs better that the UMTS specifications demands.

In many devices the requirement of portability (e.g., mobile phones) places several restrictions on the consumed power dissipation. In order to meet these restrictions, special attention has been taken in the design of the proposed architecture (Figure 2). It is well known that the hardware resources reduction results in a total active capacitance reduction. So, the overall system power consumption is reduced.

The data transfer to/from external components, through long interconnection lines, results in an enormous increase of the active capacitance. In the proposed architecture, the use of on-chip storage resources reduces significantly the active capacitance. This is achieved by using internal memory blocks.

Furthermore, wasteful circuitry activity has been avoided by adopting sleep/idle modes on the proposed architecture internal components.

The constants *ci*, and the integers *ri* are stored and accessed from the 20-byte ROM blocks. The OPc-value is stored and accessed from the RAM. The required authentication parameters are produced on the output of the 2nd RIJNDAEL and the 3rd RIJNDAEL modules.

3.2. RIJNDAEL Block Cipher Module

The proposed hardware implementation of the RIJNDAEL block cipher is shown in Figure 4. This is similar to the implementation in Reference [19], but uses less hardware resources.

The transformations of the algorithm architecture operate on the intermediate result, called State. The State can be pictured as a rectangular array of bytes. This array has four rows. The number of columns (N_b) is equal to the block length divided by 32. The Key is also considered as a rectangular array with the same number of rows as State. The number of columns (N_k) is equal to the key length divided by 32. The number of rounds (N_r), depends on the values N_b and N_k. For block and key length equal to 128 bits, both values of N_b and N_k are equal to 4 and the N_r is defined as 10.

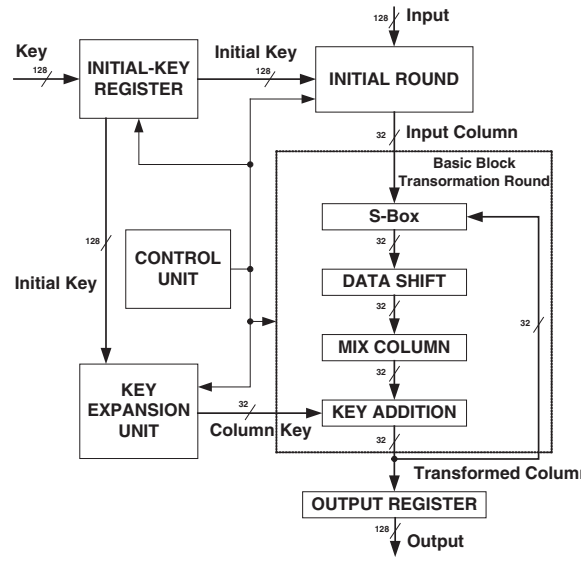

Fig. 4. RIJNDAEL block cipher hardware implementation.

The proposed architecture (Figure 4) consists of the key expansion unit, the basic block transformation round, the initial round, and the appropriate registers. Forty-one clock cycles are needed for the completion of a 128-bit plaintext transformation.

The basic block transformation round is composed of four building blocks: S-BOXes, Data Shift, Mix Column, and Key Addition. The most challenging design is the structure of S-BOXes. In order to reduce the required hardware resources and achieve high-speed performance, two alternative designs are proposed for the S-BOXes implementations. The ROM-based design and the combinational logic-based design.

The ROM-based design offers high-speed performance, while the combinational logic-based one minimizes the covered area with a performance delay penalty.

In general, FPGAs devices have internal available ROM (RAM) blocks. In the proposed implementation, four [256x8]-bit ROM blocks were used. The S-BOX delay time, implemented by ROM blocks, is 12.8 ns.

The S-BOXes require the implementation of two different mathematical functions: (1) the multiplicative inverse of each byte of the State in the finite field GF(2^8) and (2) an affine mapping transformation over GF(2). The most known VLSI architecture for the multiplicative inverse in GF(2^m), uses arrays of basic inversion block cells [20–22]. This design has time and area requirements with complexity varying from O(m^2) to O(m^4) [20–22]. The execution of the multiplicative inverse in GF(2^m) needs a number of cycles per inversion in the range between m and $3m+2$ [21,22]. These values are unacceptable for a high-speed implementation of a cryptographic algorithm. The multiplicative inverse function produces a byte, which is the input of the affine mapping transformation function. This function is defined as:

$$\begin{aligned}\mathrm{Out}[i] = {} & \mathrm{In}[i]\mathrm{XOR}\ \mathrm{In}[(i+4)\mathrm{mod}\ 8] \\ & \mathrm{XOR}\ \mathrm{In}[(i+5)\mathrm{mod}\ 8]\mathrm{XOR} \\ & \times \mathrm{In}[(i+6)\mathrm{mod}\ 8]\mathrm{XOR} \\ & \times \mathrm{In}[(\mathrm{i}+7)\mathrm{mod}\ 8]\mathrm{XOR}\ \mathrm{C}(i)\end{aligned}$$

where In[i] is the i-th bit of the input byte, and C(i) is the i-th bit of a byte constant C ($=\{01100011\}$), as the algorithm specifications defines.

The round keys are calculated on the fly by the key expansion unit. So, the keys production procedure has no additional time delay cost on the RIJNDAEL critical path.

3.3. Message Authentication Code and Confidentiality Protection Units

The *Integrity Algorithm f*9 [10] with the use of the integrity key computes the *Message Authentication Code* (*MAC-I*) of a message. The length of the message may be between 1 and 20 000 bits. The *f*9 algorithm is based on a block cipher KASUMI, in a variant version of the *Cipher Block Chaining-Message Authentication Code Mode* (*CBC-MAC*) standard as defined in ISO 9797 [23].

The *f*9 Data Mapping subunit (Figure 5(a)) produces the initial values for the Message Authentication Code (*f*9) unit according to the input parameters. It computes the Padding String (PSi), which is one input of the basic feedback loop. The maximum message length is 20 000 bits (LENGTH $\leq$ 20 000) and the maximum number of the PSi is 313 ($\approx$20 000/64). This value denotes the number of the iteration loops.

The integrity algorithm implementation uses two registers to hold variables A and B. After each loop operation, the registers A and B are updated and the new values are the new input of the KASUMI module. Finally, one last execution of the KASUMI algorithm is performed using a modified IK and the content of the register B. After this operation, in the leftmost 32 bits of register B, the MAC-I value is stored.

The Confidentiality Algorithm *f*8 [10] is a stream cipher that encrypts/decrypts block of data between 1 and 20 000 bits in length by using a confidentiality key. The *f*8 algorithm is based on a KASUMI cipher in a form of *Output Feedback Mode* (*OFB*) [24]. It generates the key stream in multiples of 64 bits. The confidentiality algorithm (*f*8) is implemented in the proposed confidentiality protection (*f*8) unit (Figure 5(b)).

The *f*8 data-mapping subunit, pads the initial KASUMI values to make a 64-bit input, and computes the input block number (BLKCNT). During the initialization process (first loop execution), MUX subunit selects the IN1 (initial input) and the KASUMI module produces the initial KS by using the modified CK.

This initial KS is stored in a register and is used for the next iterations. After the initial iteration, in all the iterations, MUX select the second input (IN2) and the CK is used by the KASUMI module. The block count (BLKCNT) counter is set initially to 0 and is increased by one after each iteration. The maximum value of the counter is (LENGTH/64) rounded up to the nearest integer, which is the number of iterations.

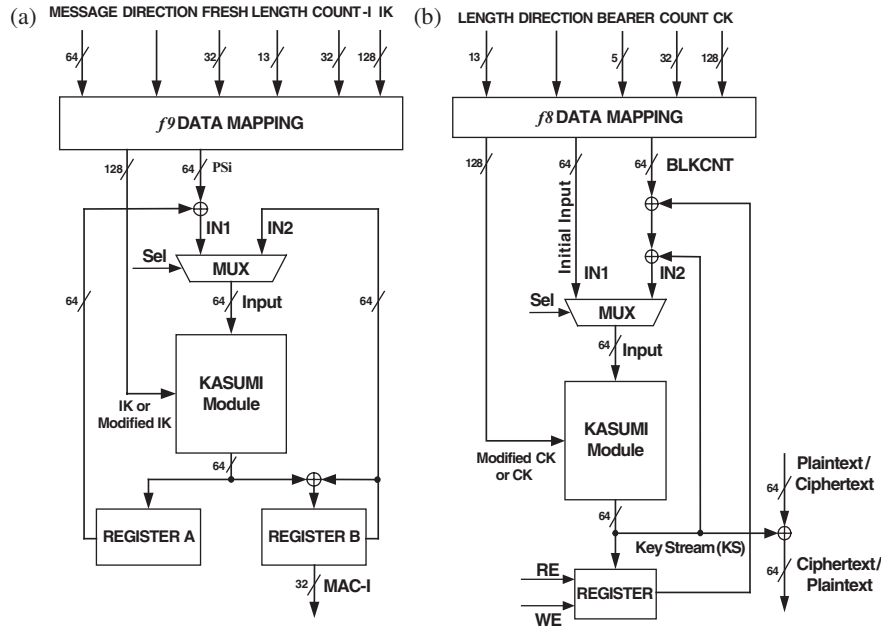

Fig. 5. (a) Message authentication code ($f9$) unit; (b) confidentiality protection ($f8$) unit.

The input LENGTH defines the plaintext/ciphertext length (# of bits).

3.4. KASUMI Block Cipher Module

The KASUMI block cipher [10,11] is used by the confidentiality algorithm ($f8$), and the integrity algorithm ($f9$). KASUMI is an enhanced version of *MISTY* [25]. Comprises with *Feistel* networks [26] operation with eight rounds, operates on a 64-bit input. It produces a 64-bit output according to a 128-bit ciphering key. Lets denote as f_i each round. The f_i has two different forms depending on whether it is an even round or an odd round. On the even rounds the computation procedures are executed in reverse order comparing with the odd rounds.

Lets denote as *odd round cell* (*ORC*) the odd rounds and as *even round cell* (*ERC*) the even rounds. For the implementation of the complete algorithm the data are applied in a repeated manner to the two basic cells, one ORC and one ERC (Figure 6(a)).

The round keys are computed by the *key expansion unit*. The total key schedule constitutes by hardwired right shifters that produce many sub-keys while the remaining are generated by bit-wise XOR operation with predefined constants. Forty 16-bit sub-keys are generated in total. Many of them are used more than once in order to generate the round keys with the appropriate concatenations. The round keys are precomputed and store in a 64×16 bit RAM memory. So, the system does not need to generate the keys for the decryption mode.

In order to increase the cipher performance the inner-round pipeline technique is used. A negative edge-triggered register is used for the pipeline inside the rounds. In a conventional pipeline with positive edge-triggered register design between the rounds, the data are transferred between two successive registers in one clock period. The use of this technique (negative edge-triggered pipeline) results in a significant reduction of the round critical path delays. The negative edge pipeline register is inserted in the *FO* function (Figure 6b), which is roughly in the middle of the round data path. This technique has been recently proposed for low-power and high-speed designs [13]. The execution time of each round is one system clock cycle. In order to synchronize the processing data paths, similar registers are inserted in the left and right branches of each round (Figure 6b). The result of this insertion is the reduction to roughly the half of the clock period. So, for a given throughput, the clock frequency can be reduced with a subsequent reduction in power dissipation.

As in the RIJNDAEL block cipher, for the KASUMI block cipher implementation the S-BOXes were designed by using two approaches, the one based on ROM and the other on combinational logic. So, the proposed KASUMI block cipher implementations performance is very high, as the UMTS standard demands.

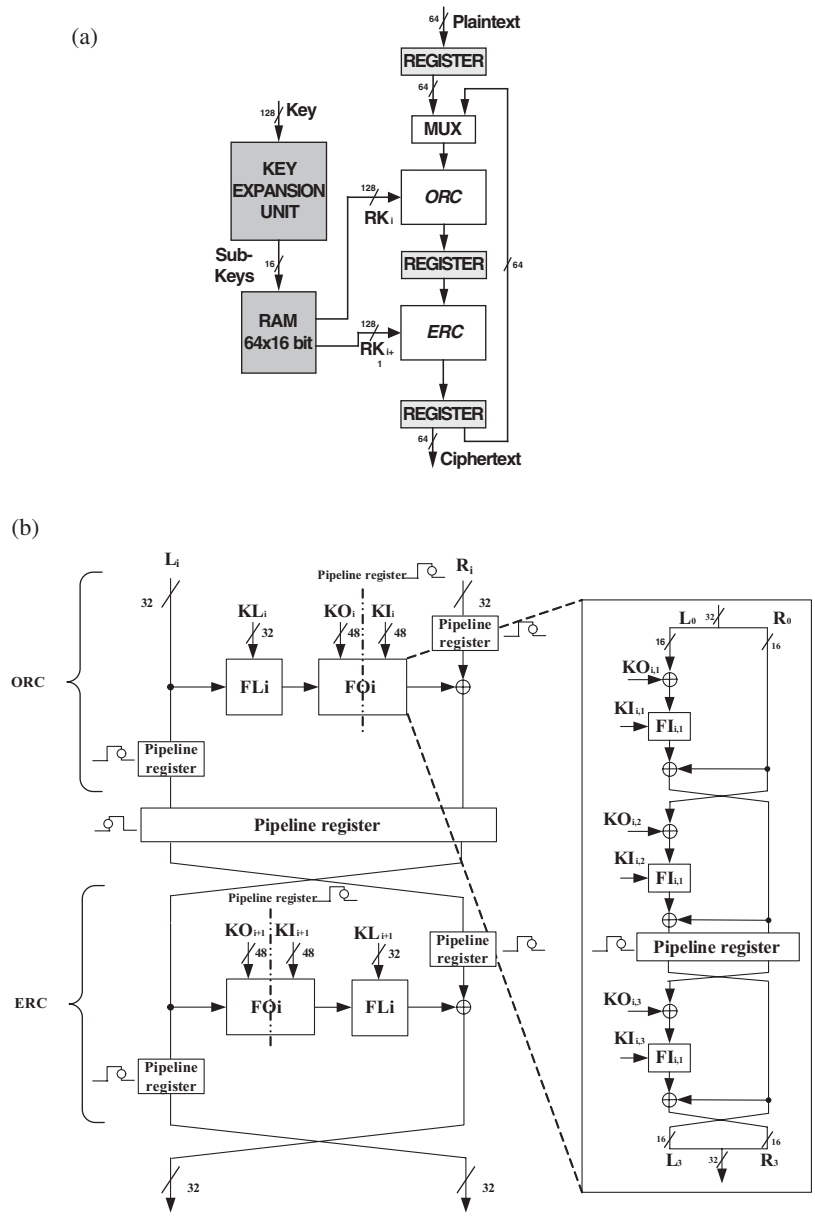

Fig. 6. (a) KASUMI block cipher hardware implementation; (b) the rounds of KASUMI hardware implementation.

4. Results and Evaluation

The VLSI synthesis results of the proposed UMTS security hardware implementation are presented in this section. The whole system (Figure 2) was captured by using VHDL with structural description logic. The VHDL code was synthesized, placed, and routed by using Xilinx FPGA devices [27]. The VHDL code was simulated and verified by using the 'implementors' test data [28]. The synthesized system was simulated and verified considering real time operation condition by using the design conformance test data, provided by the 3GPP standard [29,30]. The two system fragments are implemented in two separate FPGA devices.

The synthesis results for the RIJNDAEL module and AKA unit implementations are illustrated in Table I. The FPGA device XILINX V400E-FG676 was used.

The performance measurements of the two RIJNDAEL block cipher implementations are shown in Table II. Measurements from other designs are also

Table I. RIJNDAEL module and AKA unit implementation synthesis results.

	RIJNDAEL module	AKA unit	RIJNDAEL module	AKA unit
S-Boxes implementation	Based on combinational logic		Based on ROM	
Function generators	2035	6308	2387	7390
Configurable logic blocks	1153	3574	1194	3820
D flip-flops	715	2008	715	2212
Frequency: F(MHz)	67	62	78	70
Throughput (Mbps)	209	193	243	218

Table II. RIJNDAEL block cipher implementations performance measurements.

Architecture	Device	CLBs	Frequency (MHz)	Throughput (MBPS)
[5]	XCV1000 BG560	5302/10992	14,1/31,8	300/1940
[6]	XILINX (no specified)	5673	—	353
[7]	Xilinx Virtex	2902	25,9	331
[8]	ASIC approach	3.96 mm^2	100	910
[9]	Altera APEX1K4001	845 LE	—	750 (best)
Based on ROM	XCV200 EFG456	1603	78.3	244
Based on combinational logic	XCV200 EFG456	2725	67.1	209

included in this table for comparison with previous published works.

For the completion of a 128-bit block transformation, both implementations need 41 clock cycles. With operating frequency 78 MHz, the ROM-based implementation throughput is 244 Mbps. In order to build four S-BOXes, 4x2048 ROM bits storage are needed. The covered area is 40% less compared with the combinational logic implementation. The combinational logic implementation with a 67 MHz system clock provides 209 Mbps throughput.

The full 3GPP AKA algorithm set, can be implemented on an *Integrated Circuit Card*, equipped with an 8-bit microprocessor, ROM, and RAM modules. From the available hardware resources, 6 kbytes ROM and about 200 bytes RAM are needed for the RIJNDAEL function. Running with a 3.25 MHz clock, the implementation must produce the AK, XMAC-A, RES, CK, and IK in less than 500 ms. The RIJNDAEL should produce the 128-bit output values in less than 50 ms [4]. From the results, Tables I and II, it is obvious that both proposed architectures meet these requirements.

The proposed RIJNDAEL implementations use 128-bit data and key blocks. These implementations are slight slower (about 10%) in terms of throughput than other previous works [5–9]. A high-speed reprogrammable RIJNDAEL design is shown in Reference [31]. This design supports any combination of block length (128-, 192-, and 256-bit) and key length (128-, 192-, and 256-bit). But, this is not a drawback. Both implementations perform efficiently according to the UMTS specifications. The ROM-based one produces, with 78 MHz clock frequency, the output in 0.52 μs, while the combinational logic produces, with 67 MHz clock frequency, the output in 0.61 μs. With clock frequency 3.25 MHz both RIJNDAEL implementations produce the output in 12.6 μs comparing to the 50 ms that the UMTS demands.

The major advantage of the proposed implementations is the minimized covered area resources. This is very important in applications with strict area limitations (e.g., portable systems). Only 1 kbyte ROM is used in contrary to 6 kbyte that 3GPP specifies. There is also no need for RAM blocks.

In the proposed AKA procedure implementation only 1044 bytes ROM and 128 bits RAM are needed to store the OPc values. So, the required memory locations are less than the locations which 3GPP specifies (8 kbytes ROM and 300 byte RAM). In the proposed design, the parameters AK, XMAC-A, RES, CK, and IK are calculated in 76 μs, whilst UMTS specifies 500 ms at a clock frequency of 3.25 MHz. This reduction is very important because the AKA unit can process more messages in the same time.

The synthesis results for the KASUMI module are presented in Table III. The FPGA device XCV300E-8BG432 was used. This device has 16 kbyte internal

Table III. KASUMI module, $f8$ unit, and $f9$ unit synthesis results.

	KASUMI module	$f8$ unit	$f9$ unit	Total
S-Boxes implementation		Based on combinational logic		
Function generators	3452	3697	3911	7608
Configurable logic blocks	1726	1863	1974	3589
D flip-flops	1571	1789	1974	3360
Frequency: F (MHz)	54	53.5	53.5	53.5
S-Boxes implementation		Based on ROM		
Function generators	1819	2064	2278	4342
Configurable logic blocks	910	1042	1158	2200
D flip-flops	1570	1788	1973	3761
Frequency: F(MHz)	105	104	104	104

ROM divided in 32 blocks. In the ROM-based design, 688 bytes of ROM are used. For the combinational logic-based design, 432 Mbps throughput at 54 MHz was achieved, while for the ROM-based, 840 Mbps at 105 MHz.

In References [14,15], a two-round architecture with a conventional approach was proposed. Combinational logic and lookup table (LUT) used in order to implement the KASUMI S-BOXes. In addition, in Reference [15] an area efficient architecture for the $f8$ was proposed. In Reference [16], two implementation versions are proposed. The first is the low power version (Type1) and the second is the high performance version (Type2) with the usage of four-stage pipeline. Finally, in Reference [17] a hardware implementation that reduces the hardware resources is presented. Two synthesis results are given. The first (Synth1) was made with speed grade -6, and the second (Synth2) was made with speed grade -8. The proposed KASUMI architectures outperform all of the above (two rounds) implementations in term of time performance as shown in Table IV.

Table IV. KASUMI, $f8$ and $f9$ time performance comparisons.

Architecture	Device	Frequency (MHz)	Throughput (Mbps)
KASUMI_Comb in [14,15]	XCV300E-6BG432	20.88	167.04
KASUMI_LUT in [14,15]	XCV200E-6FG456	35.35	70.70
KASUMI_Type1 in [16]	—	20	110
KASUMI_Type2 in [16]	—	60	410
KASUMI_Synth1 in [17]	XCV300E-8BG432	33.14	265.12
KASUMI_Synth2 In [17]	XCV300E-6BG432	28.38	227.04
Proposed Based on ROM	XCV300E-8BG432	105	840
Proposed based on combinational logic	XCV300E-8BG432	54	432
$f8$_Comb in [14]	XCV300E-6BG432	20.52	162.1
$f8$_LUT in [14]	XCV200E-6FG432	33.14	261.8
$f8$_Comb in [15]	XCV300E-6BG432	16.93	135
$f8$_LUT in [15]	XCV600E-6FG432	46.56	372
$f8$_type1 in [16]	—	19.5	154
$f8$_type2 in [16]	—	52	411
$f8$_synth1 in [17]	XCV300E-8BG432	30.12	240.96
$f8$_synth2 in [17]	XCV300E-8BG432	25.80	206.40
Proposed based on ROM	XCV300E-8BG432	104	822
Proposed based on combination logic	XCV300E-8BG432	53.5	423
$f9$_Comb in [14]	XCV300E-6BG432	20.68	165.44
$f9$_LUT in [14]	XCV200E-6FG432	20.19	161.52
$f9$_Comb in [15]	XCV300E-6BG432	16.70	134
$f9$_LUT in [15]	XCV600E-6FG432	35.34	340
$f9$_synth1 in [16]	XCV300E-8BG432	30.12	240.96
$f9$_synth2 in [16]	XCV300E-8BG432	25.80	206.40
Proposed based on ROM	XCV300E-8BG432	104	822
Proposed based on combination logic	XCV300E-8BG432	53.5	423

The synthesis results for the message authentication code ($f9$) unit and the confidentiality protection ($f8$) unit are also illustrated in Table III. Both units were integrated in the same FPGA device.

Table IV also shows the comparisons between the proposed $f8$ and $f9$ units and the previous designs. It is obvious that the proposed implementations perform better in term of performance.

The proposed $f8$ implementation achieves 75.4% (combinational-based S-BOXes) and 760% (ROM-based S-BOXes) higher throughput than the implementation in Reference [14]. The MAC computation time in the proposed $f9$ combinational-based S-BOXes implementation is about 43% faster than the implementation in Reference [14]. The same measurement for the ROM-based S-BOXes implementation gives about 88% less average time. In addition, the $f8$ and $f9$ implementation covered area is half of the covered area in Reference [14]. In Reference [15], the Type 1 $f8$ implementation operates with 19.5 MHz, while the Type 2 one operates with 52 MHz clock frequency. As Table IV shows the proposed implementations provide much higher throughput comparing to Type 1 and Type 2 implementations.

5. Conclusion

A hardware implementation of the UMTS security system was presented in this paper. The introduced system supports the AKA, the user data and signaling information confidentiality protection, and the signaling information integrity. With the proposed designs a major hardware resources reduction is achieved. When the hardware resources are reduced the power consumption of the overall system is reduced too. The proposed AKA implementation executes the procedure within 76 μs comparing with the 500 ms that the UMTS specify. The main architectural units of the system are based on the RIJNDAEL and KASUMI block ciphers. Both algorithms use S-BOXes as fundamental elements. The S-BOXes have been implemented in combinational logic as well as with the use of ROM blocks. An efficient RIJNDAEL architecture is proposed in order to reduce the required hardware resources. The proposed KASUMI architecture reduces the hardware resources and power consumption. It uses feedback logic and positive-negative edge-triggered pipeline in order to make the critical path shorter without increasing the latency of cipher execution. The system was synthesized, placed, and routed by using FPGA devices. The results of the implementation synthesis were presented and compared with previous designs. The comparison shows that the proposed design may be used efficiently in future UMTS devices.

References

1. 3GPP TS 33.102 V 4.2.0, Technical Specification Group Services and System Aspects, 3G Security. Security Architecture, September 2001.
2. Daemen J, Rijmen V. AES Proposal: Rijndael, available at http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Rijndael/Rijndael.pdf.
3. 3GPP TS 35.206 V4.0.0, Technical Specification Group Services and System Aspects, 3G Security, Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*, Document 2: Algorithm Specification, April 2001.
4. 3GPP TR 35.909 V4.0.0, Technical Specification Group Services and System Aspects, 3G Security, Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*, Document 5: Summary and results of design and evaluation, April 2001.
5. Elbirt AJ, Yip W, Chetwynd B, Paar C. An FPGA based performance evaluation of the AES block cipher candidate algorithm finalists. In *Proceedings of 3rd Advanced Encryption Standard* (*AES*) *Candidate Conference* (2000), New York, USA, April 13–14.
6. Dandalis A, Prasanna VK, Rolim JDP. A comparative study of performance of AES final candidates using FPGAs. In *Proceedings of 3rd Advanced Encryption Standard* (*AES*) *Candidate Conference*, New York, April 13–14, 2000.
7. Gaj K, Chodowiec P. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. In *Proceedings of 3rd Advanced Encryption Standard* (*AES*) *Candidate Conference* (2000), New York, USA, April 13–14.
8. Kuo H, Verbauwhede I. Architectural optimization for a 1.82 Gbits/s VLSI implementation of the AES Rijndael algorithm. In *Proceedings of CHES 2001*, France, May 14–16.
9. Fischer V, Drutarovsky M. Two methods of Rijndael implementation in reconfigurable hardware. In *Proceedings of CHES 2001*, France, May 14–16.
10. $f8$ and $f9$ Specification, Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 1, ETSI/SAGE, September 2000.
11. KASUMI specification, Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 2, ETSI/SAGE, December 1999.
12. 3GPP KASUMI Evaluation Report. Security Algorithms Group of Experts (SAGE), Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms, SAGE version 2.0.
13. Strollo AGM, Napoli E, Cimino C. Analysis of power dissipation in double edge-triggered flip-flops. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems* 2000; **8**(5): 624–629.
14. Marinis K, Moshopoulos NK, Karoubalis F, Pekmestzi KZ. On the hardware implementation of the 3GPP confidentiality and integrity algorithms. *Proceedings of the 4th International Conference for the Information Security* (2001), *ISC 2001* Malaga, Spain, October 1–3; pp. 248–265.

15. Marinis K, Moshopoulos NK, Karoubalis F, Pekmestzi KZ. An area optimized hardware implementation of the 3GPP confidentiality and integrity algorithms. Proceeding of the *8th Conference on Optimization of Electrical and Electronic Equipment* (2002), *OPTIM 2002*, Brasov, Romania, May 16–17.
16. Kim H, Choi Y, Kim M, Ryu H. Hardware Implementation of 3GPP KASUMI Crypto Algorithm. The *2002 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)* 2002, 1, July 16–19, Phuket, Thailand; pp. 317–320.
17. Satoh A, Morioka S. Small and high-speed hardware architectures for the 3GPP standard cipher KASUMI. In *Proceedings of the 5th International Conference Information Security, ISC 2002* (2002). Sao Paulo, Brazil, September 30–October 2, 2002, Lecture Notes in Computer Science 2433 Springer-Verlag.
18. Dohmen JR, Olaussen L. UMTS Authentication and Key Agreement, Graduate Thesis, Agder University College—2001, on line available at http://siving.hia.no/ikt01/ikt6400/jrdohm99/
19. Sklavos N, Koufopavlou O. Architectures and VLSI implementations of the AES-proposal Rijndael. *IEEE Transaction on Computers* 2002; **51**(12): 1454–1455.
20. Brunner H, Curiger A, Hofstetter M. On computing multiplicative inverses in GF(2^m). *IEEE Transactions on Computers* 1993; **42**(8): 1010–1015.
21. Wang CC, Truong TK, Shao HM, Deutsch LJ, Omura JK, Reed IS. VLSI architectures for computing multiplications and inverses in GF(2^m). *IEEE Transactions on Computers* 1985; **C-34**(8): 709–717.
22. Araki K, Fujita I, Morisue K. Fast inverters over finite field based on Euclid's algorithm. *Transaction IEICE* 1989; **E-72**(11): 1230–1234.
23. ISO/IEC 9797-1:1999 (E). Information technology—Security techniques—Message Authentication Codes (MACs)–Part 1.
24. Recommendation for Block Cipher Modes of Operation. Methods and Techniques. National Institute of Standards and Technology (NIST). Available at http://csrc.nist.gov/encryption/modes/Recommendation/Modes01.pdf.
25. Matsui M. New block encryption algorithm MISTY. In *Fast Software Encryption '97* (1997), Vol. 1267 of LNCS, Springer-Verlag; pp. 54–68.
26. Menezes AJ, Oorschot PC, Vanstone SA. *Handbook of Applied Cryptography*. Paris Kitsos, VLSI Design Lab.: Dept. of Electrical & Computer Engineering, University of Patras, Greece, 1997.
27. Xilinx, San Jose, California, USA, Virtex, www.xilinx.com
28. 3GPP TS 35.207 V4.0.0, Technical Specification Group Services and System Aspects, 3G Security, Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*, Document 3: 'Implementors' Test Data, April 2001.
29. 3GPP TS 35.208 V4.0.0, Technical Specification Group Services and System Aspects, 3G Security, Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*, Document 4: Design Conformance Test Data, April 2001.
30. Design Conformance Test Data. Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 4, ETSI/SAGE, December 1999.
31. Verbauwhede I, Schaumont P, Kuo H. Design and performance testing of a 2.29-GB/s Rijndael processor. *IEEE Journal of Solid-State Circuits* 2003; **38**(3): 569–572.

Authors' Biographies

Paris Kitsos received his B.Sc. degree in Physics in 1999 and a Ph.D. in 2004 from the Department of Electrical and Computer Engineering, both at the University of Patras. Currently, he is a research fellow with VLSI Design Laboratory, Electrical and Computer Engineering, University of Patras. His research interests include VLSI design, hardware implementations of cryptographic algorithms, and security protocols for wireless communication systems. Dr Kitsos has published more than 45 scientific articles, tutorials and technical reports, as well as is reviewing manuscripts for International Journals and Conferences/Workshops in the areas of his research. In addition, he is a member of the IEEE and the International Association for Cryptologic Research (IACR).

Nicolas Sklavos received his Ph.D. in Electrical & Computer Engineering, and the Diploma in Electrical & Computer Engineering, in 2004 and in 2000, respectively, both from the Electrical & Computer Engineering Department, University of Patras, Greece. His research interests include cryptography, wireless communications security, computer networks, and VLSI design. He holds an award for his Ph.D. 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 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 technical reports in the areas of his research.

Odysseas Koufopavlou received his Diploma of Electrical Engineering in 1983 and the Ph.D. 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, U.S.A. He is currently a 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 130 technical papers and received patents and inventions in these areas. He served as general chairman for the IEEE ICECS'1999. He is a IEEE member.