FPGA implementation of an enhanced chaotic-KASUMI block cipher

Mahdi Madani, Camel Tanougast

FPGA Implementation of an Enhanced Chaotic-KASUMI Block Cipher

Mahdi Madani

Laboratory IETR, University of Nantes, Nantes, France
Email :mmadani49@gmail.com

Camel Tanougast

Laboratory LCOMS, University of Lorraine, Metz, France
Email :Camel.Tanougast@univ-lorraine.fr

Abstract

The radio link is a broadcast channel used to transmit data over mobile networks. Because of the sensitivity of this network part, a security mechanism is used to ensure users' information. For example, the third generation of mobile network security is based on the KASUMI block cipher, which is standardized by the Third Generation Partnership Project (3GPP). This work proposes an optimized and enhanced implementation of the KASUMI block cipher based on a chaotic generator. The purpose is to develop an efficient ciphering algorithm with better performance and good security robustness while preserving the standardization. The proposed design was implemented on several Xilinx Virtex Field Programmable Gate Arrays (FPGA) technologies. The synthesis results and a comparison with previous works prove the performance improvement of the proposed cipher block in terms of throughput, used hardware logic resources, and resistance against most cryptanalysis attacks.

Keywords: KASUMI block cipher, Hight performance, Robustness, FPGA implementation, Finite state machine, Simplified functions, Chaotic generator, Architectural synthesis.

1. Introduction

Nowadays, wireless technologies are widely used in our lives for communication in general and for mobile networks in particular. Due to the weaknesses of the radio link used, the transmitted data is secured by using ciphering algorithms.

In this study, we are mainly focused on mobile networks such as Global System for Mobile communication (GSM) and Universal Mobile Telecommunication and System (UMTS). Therefore, the main security algorithm used by those networks is the KASUMI block cipher, which forms the kernel of GSM

and UMTS security mechanisms [1, 2] widely used in most countries. However, in the last decade KASAUMI security was the object of many cryptanalyzed attacks as mentioned in [3–9]. But, investigations are still proposed to improve the security and resistance of the KASUMI algorithm [10–12]. For this reason, we decided to try proposing a solution aiming to recover this problem.

In this paper, we propose an efficient implementation of the KASUMI block cipher while overcoming its weaknesses. The main purpose is to enhance its robustness against cryptanalysis attacks and to perform improvement in terms of throughput and used hardware logic resources suitable for embedded systems like mobile phones.

The main idea is to simplify the internal structure of the standardized algorithm while including a chaotic generator to ensure a high robustness level. Therefore, the simplification is accomplished on four architectural levels. The first and low level consists to combine the initial Substitution Boxes (S-Boxes) S9 and S7 (defined in the KASUMI standard) only in one Global S-box (GS) using the combinational logic technology. The second level consists to replace the initial FI function using the ameliorated GS to form a simplified FI' function. The third level consists to replace the initial FO function based on initial FI by a new simplified FO' function based on the simplified FI' function. The fourth level consists to combine two FL and two simplified FO' functions to create the new kernel of the proposed algorithm. This technique allows us to eliminate the well-know problem of odd and even rounds. Finally, we combined our simplified KASUMI with a chaotic generator to improve the randomness of the final output keystream. A control unit based on a Finite State Machines (FSM) manages the connection between the different architecture levels.

The proposed approach has been described using a VHSIC Hardware Description Language (VHDL) description and implemented on Field-Programmable Gate Array (FPGA) technology. The robustness has been tested by evaluating the generated keystream distribution, analyzing the key sensitivity, examining the key space complexity, and investigating the National Institute of Standards and Technology (NIST) statistical tests [13]. The comparison with previous works [14–23], and experimental results show the the originality of the proposed KASUMI architecture which requires low logic resources and low power consumption while keeping the standardized properties published in the 3GPP specification documents [1, 2, 24, 25]. In terms of security, the proposed chaotic-Kasumi cipher enhances the resistance against cryptanalysis attacks compared to the regular algorithm.

The rest of this paper is organized as follows. In Sections 2, the internal architecture of the KASUMI block cipher is briefly described. Section 3 details the proposed simplified functions and optimized KASUMI architecture. Section 4 presents the proposed Chaotic-KASUMI architecture and associated used techniques. The security evaluation and analysis form the object of Section 5 including the dynamic chaotic behaviors investigated through the Lyapunov exponents. FPGA synthesis implementation results and comparisons with previous works are given in Section 6. Finally, conclusion is given in Section 7.

2. Standardized KASUMI block cipher

KASUMI's specifications were standardized by the Third Generation Partnership Project (3GPP) [24]. KASUMI is a block cipher based on the previous MYSTY1 algorithm [26]. It is characterized by a Feistel structure iterating in eight rounds (see Figure 1(a)). The plaintext is the input parameter to the first round, and the ciphertext is the output of the last round. It operates using 64-bits blocks under the control of the 128-bits Ciphering Key (CK) [24]. The internal architecture is based on three main functions : FO (see Figure1(b)), FI (see Figure1(c)), and FL (see Figure1(d)), respectively. During the execution, CK is used to generate eight sub-keys for each round i, knowing that $i = 0\ to\ 8$ (two KLi, three KOi, and three KIi).

2.1. KASUMI processing mechanism

The KASUMI input (64-bits) is divided into two 32-bits strings, mentioned as left L0 and right R0. Then, FL and FO functions are executed in the corresponding order. The output of the first and all the odd rounds is produced according to Equation 1 ($for\ i = 0,\ 2,\ 4,\ and\ 6$). Similarly, the output of the second and all the even round is produced according to Equation 2. ($for\ i = 1,\ 3,\ 5,\ and\ 7$)

$$L(i+1) = FO\ (FL\ (L(i),\ KLi\ ),\ KOi,\ KIi) \bigoplus R(i) \quad ; \quad R(i+1) = L(i) \quad (1)$$

$$L(i+1) = FL\ (FO\ (L(i),\ KOi,\ KIi),\ KLi) \bigoplus R(i) \quad ; \quad R(i+1) = L(i) \quad (2)$$

Note that $\bigoplus$ designates the bitwise XOR operation.

More details of the KASUMI three main functions are given as follows :

— As it is shown in Figure 1(d), FL is a fast simple linear function. It is applied to 32-bits data by using $KLi1$ and $KLi2$ 16-bits sub-keys. This function presents two advantages. First, this function requires a little extra cost for its hardware implementation. Second, this function makes it very difficult to follow the individual bits through the rounds.

— Figure 1(b) shows the nonlinear FO function formed by three internal rounds. It is applied to 32-bits data.At each round, a non-linear FI function in which mixing 16-bits is applied. Each output bit of the FO function depends on all the input bits, which makes it difficult to guess its output if one input bit is changed.

— The FI function is formed by four rounds. At each round, it uses one of two S7 or S9 non-linear substitution boxes (see Figure 1(c)) which are detailed in [24]. The function mixes the input data with the used sub-key to generate output data [27].

3. Optimized KASUMI architecture

This section describes the proposed optimized KASUMI architecture based on the architectural synthesis technique. We simplified the internal architecture of the regular algorithm at four hierarchical levels. The principle is to use the first (low) level to develop the second level (hight), and so on, until the last one (final level). At each level, a control unit based on FSM is used to manage the architecture.

3.1. Level 1 : joining S9 and S7 S-boxes

In the first and low level, we join S7 and S9 S-boxes of the standardized algorithm to form only one Global S-box (GS). It takes a 16-bits bloc input and executes 7/9 combinational functions depending on the control unit (7 right bits when the control bit is set to '0' and 9 left bits when it is set to '1'). Note that functions 1 to 7 use the same registers in the architecture to generate the 7 Least Significant Bits (LSB) of S7 and S9, and functions 8 and 9 are designed only to generate the 2 Most Significant Bits (MSB) of S9. A descriptive scheme is illustrated in Figure 2. The proposed design is implemented using a specific combinational function of selection characterized by a good speediness and low logic cost in the FPGA hardware area. The purpose of this technique is mainly to simplify the implementation and accelerate the FI function using this proposed GS.

3.2. Level 2 : the simplified FI' function

The simplified FI' function is based on the proposed GS. It is executed on two rounds instead of four rounds in the original In the second step, we used the proposed GS to form a simplified FI' function which can be executed on two parallel rounds instead of four rounds in the original FI function. For this purpose, in the first round, the FI' input (16-bits) is divided into two parts, left L (9-bits) and right R (7-bits), to form the GS inputs. After this step execution, GS generates two outputs that are stored on Reg1 (9-bits) and Reg2 (7-bits) registers. In the second round, the content of Reg1 and Reg2 registers forms the novel GS inputs. After this execution step, GS generates two new outputs, which are saved on Reg3 (9-bits) and Reg4 (7-bits) registers. Thereby, the output of the simplified FI' function (16-bits) is set up by the concatenation of Reg3 and Reg4 registers. Consequently, the proposed simplification allows for twice-faster execution. The detailed architecture is given in Figure 3.

3.3. Level 3 : the simplified FO' function

In the third step, we considered our simplified FI' function to realize one simplified FO' function deeply difference with the original function based on the FI function. Indeed, the proposed FO' function is executed on three rounds. In the first round, the FO' input (32-bits) is divided into two parts, left L (16-bits) and right R (16-bits). L part is combined with subkey KO1 (16-bits) using bitwise XOR operation to form the first FI' input (16-bits). Then, the output is

combined with the R part using a bitwise XOR operation to generate the first output saved on the Reg1 (16-bits) register. In the second round, the L part is combined with subkey KO2 (16-bits) using bitwise XOR operation to produce the second FI' input. Next, the output is combined with Reg1 using bitwise XOR operation to generate the second output stored on the Reg2 (16-bits) register. In the third round, Reg1 is combined with subkey KO3 (16-bits) using bitwise XOR operation to form the third FI' input. At this step, the output is combined with Reg2 using bitwise XOR operation to generate the third output to update the Reg1 register value. Thus, the output of the simplified FO' function (32-bits) is set up by the concatenation of Reg1 and Reg2 registers. The detailed architecture is depicted in Figure 4.

3.4. Level 4 : the optimized KASUMI

The optimized design of the KASUMI block is based on two FL functions and two simplified FO' functions which are looped four times to generate the same result as the original algorithm. In the first round, the input data (64-bits) is divided into two equal parts, L1 left and R1 right parts. Then, L1 is processed by the FL function providing the intermediate result executed by the simplified FO'. The result is mixed with R1 and stored in the $Reg6$ register. After that, the $Reg6$ register is processed by FO', and the output is executed by FL. The result is mixed with L1 and stored in the $Reg7$ register. The concatenation of $Reg6$ and $Reg7$ registers forms the output of the first round (OUT1). In the second round, the output of the last round (OUT1) forms the new input. Then, the same process is executed until the last round. At each round, $Reg6$ and $Reg7$ registers are updated. The detailed architecture of the proposed KASUMI block is illustrated in Figure 5. Similarly to the original KASUMI block cipher, the optimized block is controlled with 128-bits CK. At each round, the key scheduling generator produces 8 sub-keys required for two processing rounds, as described in the subsection 4.2.

4. Proposed Chaotic-KASUMI algorithm

In this section, we present additional architectural optimizations and the robustness enhancement of security by considering the proposed Chaotic-KASUMI architecture.

4.1. The Finite State Machine (FSM)

The FSM is used as a control unit to manage the internal processing of the simplified functions and blocks. The operating principle is based on the following steps :

1. controlling the inputs/outputs.
2. updating the intermediary registers.
3. verifying the final output at each level.

4. ensuring that the output of the low-level function is correctly used by the high-level function.

To understand this process, we illustrate the following case of the optimized KASUMI controlled by the FSM at level 4 :

Process : *State*
Begin
if State = State_A **then**
round = 0 ;
iner = splain ;
else if State = State_B **then**
round = 2 ;
iner = outer ;
else if State = State_C **then**
round = 4 ;
iner = outer ;
else if State = State_D **then**
round = 6 ;
iner = outer ;
else if State = State_E **then**
round = 6 ;
results = outer ;
else
NULL
end if
end Process

The state machine performing this considering case is given in the Figure 6.

4.2. The Key Schedule Generator

KASUMI is controlled using a 128-bits CK. At each round of regular KASUMI, eight 16-bits sub-keys are derived from CK using a Key Schedule Generator (KSG) and corresponding to $KLi1$, $KLi2$ for FL function, $KOi1$, $KOi2$, *and* $KOi3$ for FO function, $KIi1$, $KIi2$, *and* $KIi3$ for FI function. The process of the KSG begins by calculating two 16-bits arrays mentioned designated (Kj *and* $K'j$) and using CK, and eight 16-bits constants Cj which are summarized in Table 1.

TABLE 1 – Key scheduling constants.

C1	C2	C3	C4	C5	C6	C7	C8
0x0123	0x4567	0x89AB	0xCDEF	0xFEDC	0xBA98	0x7654	0x3210

Therefore, the 128-bits CK is subdivided into eight 16-bits sub-keys as

$$CK = K = K1 \,||K2\, ||K3\, ||K4\, ||K5\, ||K6\, ||K7\, ||K8 \tag{3}$$

Then, calculate

$$K'j = Kj \bigoplus Cj \ (j = 1, 2, ..., 8) \tag{4}$$

In the proposed optimized KASUMI block, the KSG generates two 128-bits keys called $CK1$, $CK2$ as follows :

$$CK1 = KL11\ ||KL12\ ||KO11\ ||KO12\ ||KO13\ ||KI11\ ||KI12\ ||KI13$$
$$CK2 = KL21\ ||KL22\ ||KO21\ ||KO22\ ||KO23\ ||KI21\ ||KI22\ ||KI23$$

Note that the two keys are generated at the same time. However, $CK1$ is used in the first round while $CK2$ in the second one, as illustrated in Figure 7.

To understand the key scheduling process, Figure 7 referred in [22] illustrates an example of generating two rounds of sub-keys. More details of the sub-keys generation are given in [24].

4.3. Chaotic generator

A chaotic generator is connected with the optimized KASUMI block cipher to improve its randomness and enhance its security [28]. It is considered as an additional layer in the proposed architecture. The used technique combine the generated chaotic signals with the optimized KASUMI output at each round. The resulting signal forms the new input of the algorithm at the next round and corresponds to a perturbation of the KASUMI internal functions inputs.

In this work, we used Lorenz hyper-chaotic generator. It is a four-dimensional dynamical controlled system. It is very sensitive to initial conditions and presents high recurrences [29]. It is characterized by the following equation set (5) [30, 31] :

$$\begin{cases} \dot{x} = & \text{a(y-x)} \\ \dot{y} = & \text{cx-xz-y+w} \\ \dot{z} = & \text{xy-bz} \\ \dot{w} = & \text{-dx} \end{cases} \tag{5}$$

where the control parameters a, b, c, d are positive. Usually $a = 10$, $b = 8/3$, $d = 5$, and c is varied. The system exhibits one chaotic behaviour for $c = 28$. The initial conditions $x_0 = y_0 = z_0 = w_0$ are set to (-10). The system of equations (5) is solved using the Runge Kutta fourth order resolution method (RK-4 method) [32–34]. The chaotic signals generated are illustrated in Figures 8 and 9.

To obtain more randomness in the proposed algorithm, we used the decimal part of the generated chaotic signals in the perturbation mechanism. Figures 8 and 9 prove the enhanced randomness by considering only the decimal part (compared to real and decimal parts) of the chaotic signals.

4.4. Chaotic-KASUMI global architecture

After a description of the optimized blocks, this section presents the global architecture of the proposed Chaotic-KASUMI algorithm.

The process begin by initializing the initial conditions and control parameters in the chaotic generator. In parallel, CK and Cj are loaded into the KSG to generate two keys at each round. In the first round, the external text (64-bits) forms the input to the optimized KASUMI block cipher. Then, the output is combined with generated chaotic signals using a XOR operation. The

result forms the input to the next round, and so on, until the fourth and last round. The output signal forms the generated chaotic keystream (64-bits) used to cipher (decipher) the plaintext (ciphertext) in mobile networks. This process is executed n times, where n is the number of blocks (size of one block is 64-bits) to be encrypted. For further understanding, the architecture is depicted in Figure 10.

In this proposition, we improved three main properties compared to the original KASUMI algorithm. Firstly, we reduced the algorithm complexity by optimizing its internal functions. Secondly, we dynamically updated the intermediary inputs to the optimized KASUMI by using the perturbation technique. Thirdly, we improved the randomness and the robustness of the generated output keystream. Consequently, we obtained an enhanced block cipher, which is able to resist the most important cryptanalysis attacks. The detailed evaluation results are the object of the next section.

4.5. Generated outputs comparison

To prove the efficiency of the proposed implementation, we used the 3GPP test set vector [25] designed to help implementers in their realization. This document provides test data for the algorithms as well as details on the internal states of the algorithms when they process the input data. For example, the test set vectors 2 is defined as presented in Table 2.

To validate the outputs generated by the proposed algorithm, we used the 3GPP reference data presented in Table 2 and the ISE Simulator (ISim). The simulation results (VHDL test bench waveform) are shown in Figure 11.

By comparing the original algorithm output (presented in Table 2) with the optimized algorithm output (presented in Figure 11), it is clear that the optimized implementation generates the same and expected output (signal blue in Figure 11) as the original one. In parallel, the introduced random generator generates a chaotic signal (signal red in Figure 11). Then, the two signals are mixed to generate the output keystream (signal brown in Figure 11), as we discussed in Subsection 4.4 (see Figure 10). We note that one clock cycle of the represented signal concerns the execution of one internal function tour (FO' or FL). Therefore, we need 8 clock cycles to generate the expected keystream, as shown in Figure 11.

5. Dynamic behavior and security evaluation

In this section, we justify the existence of chaos by investigating the Lyapunov exponents and we give the security tests used to evaluate the robustness of the proposed architecture. The aim of this evaluation is to prove the enhanced security level of the designed Chaotic-KASUMI block cipher. The principal properties examined in this work are randomness, robustness, key sensitivity, key space, and statistical analysis [35].

5.1. Lyapunov exponents

The chaos behavior of the proposed system described in (5) is investigated by considering the Lyapunov Exponents (LE). The LE is the average exponential of the divergence of initially nearby orbits by considering two points in a space, $X(0)$ and $X(0)+\Delta X(0)$ (similarly, $Y(n, Y(0))$, $Z(n, Z(0))$ and $W(n, W(0))$) generating in the same space two orbits $L(X(0))$ and $L(X(0)+\Delta X(0))$ (similarly, $L(Y(0))$, $L(Z(0))$ and $L(W(0))$). The function $\Delta X(X(0), t)$ behaves randomly showing the chaotic behavior. The mean exponential rate divergence of two initially close orbits is obtained by considering the following expression [36] :

$$\lambda = \lim_{t\to\infty, \Delta X(0)\to 0} [(\frac{1}{t}) \times \ln((\frac{\Delta X(X(0), t)}{t}) \times \Delta X(0))]$$

Computing the LE values (λ_i) gives a possibility to detect the presence of chaos [37]. When the calculated value of (λ) corresponding to the requested LE is superior to zero ($\lambda > 0$), the system is chaotic. Usually, a general chaotic system has only one positive Lyapunov exponent and a periodic system's largest Lyapunov exponent is zero. Considering the system 5 with $a = 10$, $b = 8/3$, $c = 28$, $d = 5$ and according to the method presented in [38], the calculation of the obtained limit values of Lyapunov exponents are as follows [39] :

$$\begin{aligned} &\lambda_1 > 0.1997; \\ &\lambda_2 > 0.0271; \\ &\lambda_3 = 0; \\ &\lambda_4 > -15.4176; \end{aligned}$$

Therefore, the considered 4D Lorenz system has more than one positive Lyapunov exponent proving that it is a hyperchaotic system and its dynamic behavior means hyperchaos and more complex than general chaos. Consequently, the considered chaotic system to perform our proposed Chaotic-KASUMI Block Cipher provides a hyperchaotic behavior (LE¿0) achieving high complex chaotic trajectories more suitable for encryption purposes since that it offers larger key space, more non-linear parameters and has more positive number of LE.

5.2. Key sensitivity analysis

The key sensitivity analysis is an important property for evaluating a cryptosystem. In this test, we used 500 random keys to evaluate the proposed Chaotic-KASUMI block cipher. In each case, we compared two generated keystreams by changing a single bit in the secret key. The results are analyzed using the Hamming Distance (HD) presented in percent and calculated according to Equation 6.

$$HD = \frac{\sum_{k=1}^{N} C_i \oplus C'_i}{N} \times 100\% \tag{6}$$

Where N is the size in bit level of the keystream, C_i and C'_i are the corresponding keystreams generated using two secret keys different in one bit CK_i and CK'_i,

respectively. According to the final results shown in Figure 12, we observe that the average Hamming Distance percent is closer to the optimal value of 50% in bit level. Consequently, we conclude that a change of a single bit in the secret key leads to a thoroughly different keystream satisfying the avalanche effect [40]. Therefore, we conclude that the proposed algorithm is very sensitive to the key changes.

5.3. Key space analysis

In cryptography, the minimum key space is fixed at 2^{128}. According to this condition, we expected to increase the key space from 2^{128} with the original KASUMI algorithm to 2^{526} by considering the proposed chaotic KASUMI cipher.

The strength of the proposed algorithm lies in the additional chaotic generator associated with the regular algorithm. The principle property of this generator is its hight sensitivity to initial conditions $(x0, y0, z0, w0)$ and control parameters (a, b, c, d) defined in the equation set (5). Thereby, the new key space is formed by respecting those eight parameters in addition to the original CK. For example, we fixe the initial conditions and control parameters a, b, c, whereas we consider d the random number generator key. The obtained results show that the variance ratio of each bit is approximated to 99 %, even if the change of the key (d) is an extremely small value 10^{-15}, which means that the system is extremely sensitive to the key changes. Consequently, the keys pace corresponding to the considered key (d) is larger than 10^{15}. Similarly, the four initial condition values and the three remaining control parameters can be considered as input keys to our random number generator. Therefore, the random number generator key space is then equal to $(10^{15})^8 = 10^{120} \approx 2^{398}$. The final key space is defined by $2^{398} \times 2^{128} = 2^{526}$, which is large enough to make infeasible exhaustive or brute-force attacks.

5.4. Statistical analysis

To evaluate the statistical properties of the proposed architecture, we used the NIST statistical tests [13]. In this study, we analyzed 300 vectors of 10^9-bits in size generated by the proposed Chaotic-KASUMI block cipher. The obtained results are presented in Table 3.

According to these results, we remark that the generated keystreams are characterized by hight statical properties. Consequently, we conclude the resistance of the proposed algorithm against standards tests (linear, frequency, serial, randomness, etc.).

5.5. Synthesis and discussion

The avalanche effect [40], confusion and diffusion properties defined on Shannon's theory [41] are the main basic references used in cryptography to evaluate the security of the algorithms. Therefore, the obtained key sensitivity analysis results prove the satisfaction of the avalanche effect, Shannon's confusion, and diffusion properties. Besides, the achieved key space of 2^{526} is sufficiently large

to make infeasible the brute-force and exhaustive attacks. Finally, the good statistical test results prove the high resistance of the proposed algorithm against cryptanalysis attacks. Furthermore, the added nonreversible chaotic generator limits the ability of attackers attempting to break the encrypted text while improving the algorithm robustness.

6. FPGA implementation

The architecture proposed in this work has been described using VHDL (VHSIC Hardware Description Language) structural description and has been implemented on Xilinx Virtex FPGA [42–44]. Integrated Synthesis Environment (ISE) 13.2 of Xilinx tools have been used for this digital implementation allowing us to obtain the logic resource requirements and the associated real-time constraints. It has been designed in two steps.

We implemented the optimized KASUMI block cipher in the first step. The corresponding Xilinx Synthesis Technology (XST) results after place and route are shown in Table 4.

From these results, we conclude that the proposed architecture provides good performance in terms of throughput 5154.64 Mbps on Virtex-5, 2009.04 Mbps on Virtex-E, and 2811.92 Mbps in Virtex-II) due to reduced latency (8 cycles). We also observe that FPGA hardware logic resources are used efficiently (slices registers, LUTs and Flip-Flop used by the proposed simplified KASUMI occupies just 4% on Virtex-5, 32% on Virtex-E, and 2% on Virtex-II).

We implemented the proposed Chaotic-KASUMI block cipher in the second step. The corresponding XST results after place and route are shown in Table 5.

We remark that the chaotic generator connected to the optimized KASUMI requires some additional hardware resources (644 slices on Virtex-5) (see Table 5). It should also be known that even if the throughput is reduced to 475.60 Mbps, it remains compatible with the encryption of data exchanged between users of mobile networks in real-time. Therefore, by considering the tradeoff between the used resources, throughput, and security level, we conclude that the enhanced Chaotic-KASUMI architecture forms a good solution to protect data transmitted over mobile networks.

6.1. Comparison and discussion

To make a comparison with previous works, we implemented the proposed architecture on different Xilinx Virtex devices (Virtex-300E, Virtex-8000II) including the Virtex 5 technology to improve performance while proving the portability of the proposed architecture. The implementation results are presented in Table 6.

In this work, the first objective was the improvement of the algorithm throughput (by maximizing the clock frequency and minimizing the latency) defined in Equation 7, and which is proved considering the results presented in Table 6.

$$Throughput = \frac{block\ size \times clock\ frequency}{latency\ cycle} \tag{7}$$

Based on those results, we conclude that the proposed architecture achieves the highest throughput due to the high clock frequency (251.13 MHz on Virtex-300E and 351.49 on Virtex-8000II) and the low latency (8 cycles).

The second objective was the minimization of the used hardware resources. Therefore, by considering the compromise between the hardware logic area and throughput, we conclude that the proposed architecture is the best implementation on FPGA technologies. For example, it is faster than implementation [23] by a factor of 45.11, and its efficiency is better than implementation [23] by a factor of 15.41.

The third objective was the enhancement of the robustness of the KASUMI block cipher to improve its resistance against cryptanalysis attacks. This aim is obtained according to the used chaotic generator, which is combined with the original architecture. However, the occupied area on the FPGA device has slightly increased (see Table 5).

Finally, by considering the compromise between the high level of security provided by the proposed architecture, and the additional area occupied on the FPGA device, we conclude that this architecture forms a good solution for ensuring the security of data transmitted over mobile networks. Additionally, we note that this architecture can still be used by the embedded applications in real-time, such as mobile networks in our case study because it still offers a high throughput (see Table 6) and respects the standardized properties fixed in the 3GPP documents [1, 24, 25].

6.2. Potential uses

The proposed architecture (enhanced Chaotic-KASUMI) can be used either as the kernel of current mobile network security mechanisms based on the standard KASUMI block cipher (f8 confidentiality and f9 integrity functions used in UMTS, A5/3 and A5/4 algorithms used in GSM, GEA3 algorithm used in the GPRS and EDGE, etc.), or in the security mechanisms of future mobile networks. Finally, we specify that the proposed block cipher can be configured in different operating modes without risk of compromising the security provided, mainly in ECB (Electronic Codebook) and CBC (Cipher-Block Chaining) modes.

7. Conclusion

In this paper, we proposed two hardware implementations. In the first one, we optimized the KASUMI block cipher using four architectural levels. In the second one, we enhanced the proposed optimized KASUMI by using a chaotic generator. The used technique is based on simplifiying the internal functions used by the regular KASUMI block cipher (S-boxes S7, S9 and functions FI, FO). After that, we combined the generated keystream (output of optimized KASUMI) with the generated chaotic signals to form an output keystream with improved randomness properties. The proposed architectures have been implemented on FPGA Virtex technologies.

Table 2 – 3GPP Test set vector2.

Key	=	$8CE33E2CC3C0B5FC1F3DE8A6DC66B1F3$
$Input$	=	$D3C5D592327FB11C$
$Output$	=	$DE551988CEB2F9B7$

Table 3 – Comparison results of NIST tests.

The NIST test	Proposed Chaotic-KASUMI
Frequency (mono-bit) Test	success
Frequency Test with a Block	success
Runs Test	success
Long runs of ones Test	success
Binary Matrix Rank Test	success
Discrete Fourier Transform (Spectral) Test	success
Non-overlapping Template Matching Test	success
Overlapping Template Matching Test	success
Maurer's "Universal Statistical" Test	success
Linear complexity Test	success
Serial Test	success
Approximate Entropy Test	success
Cumulative sums (Cusum) Test	success
Random excursion Test	success
Random excursion variant Test	success

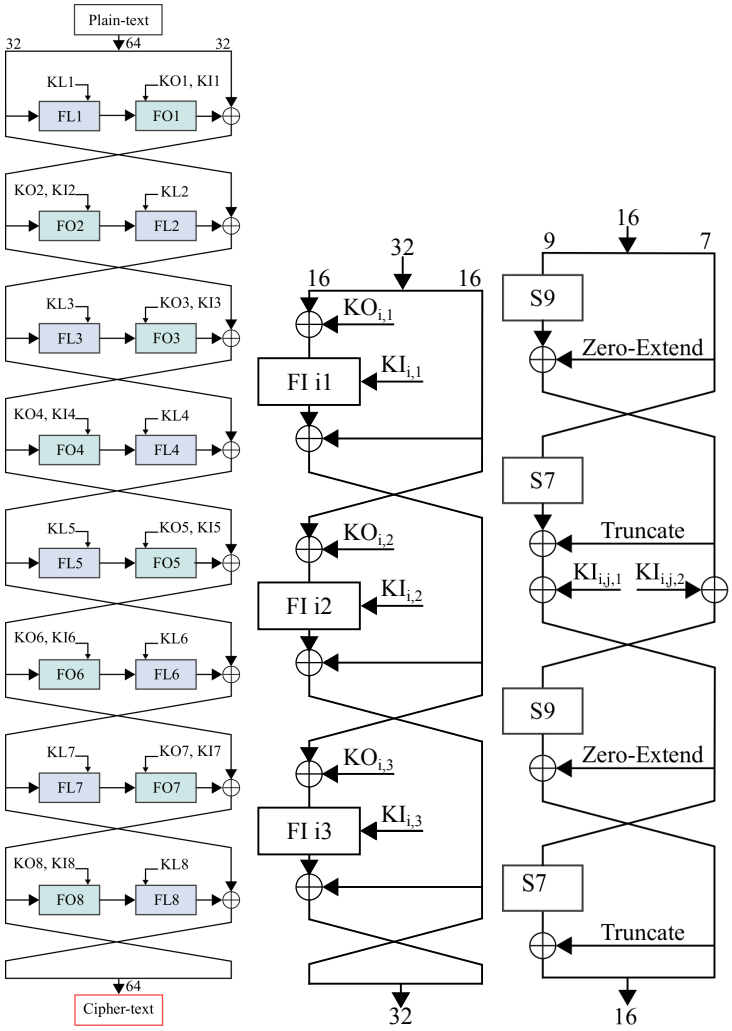
To confirm the performance of the proposed architecture, we compared it with previous works and evaluated it using many security tests. The obtained results prove the enhancement of the ciphering algorithm performance in terms of used hardware logic resources, throughput, security level, and robustness.

The development of new confidentiality and integrity functions based on the proposed Chaotic-KASUMI will be the object of our future works to improve the security mechanism of current and new mobile networks.

Table 4 – FPGA implementation results of the optimized KASUMI.

Device	Latency	Frequency	Throughput	Area(slice)		
	(cycle)	(Mhz)	(Mbps)	Amount	Total	Percentage
Virtex-5	8	644.33	5154.64	468	11200	4%
Virtex-II	8	351.49	2811.92	964	46592	2%
Virtex-E	8	251.13	2009.04	972	3072	32%

Table 5 – Chaotic-KASUMI FPGA implementation results.



(a) Architecture of KASUMI block cipher.

(b) The FO function.

(c) The FI function.

(d) The FL function.

FIGURE 1 – The Feistel structure of KASUMI [24].

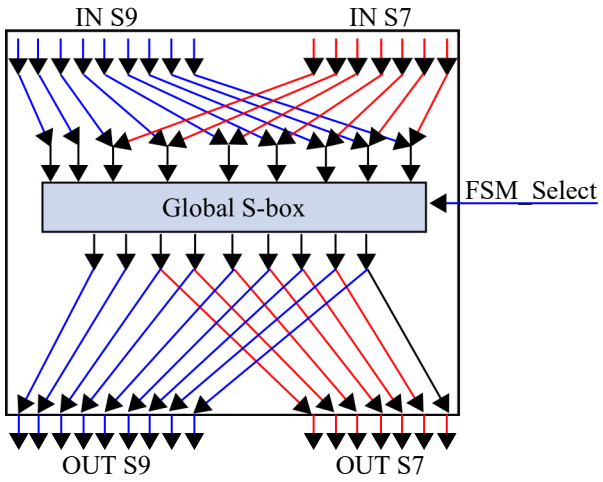


FIGURE 2 – The proposed Global S-box.

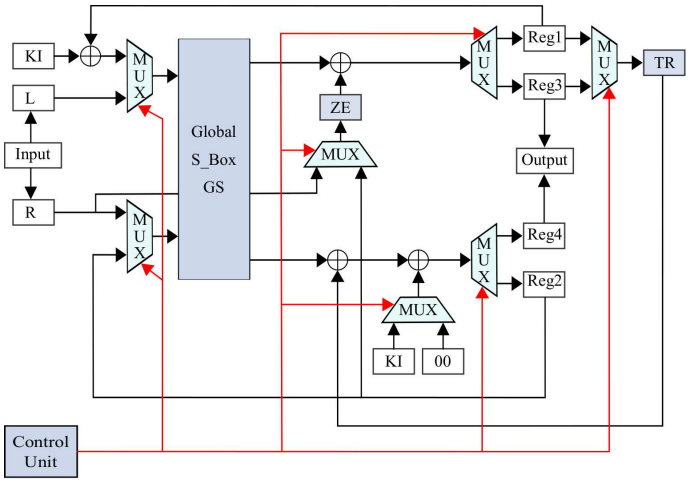


FIGURE 3 – The simplified FI' architecture.

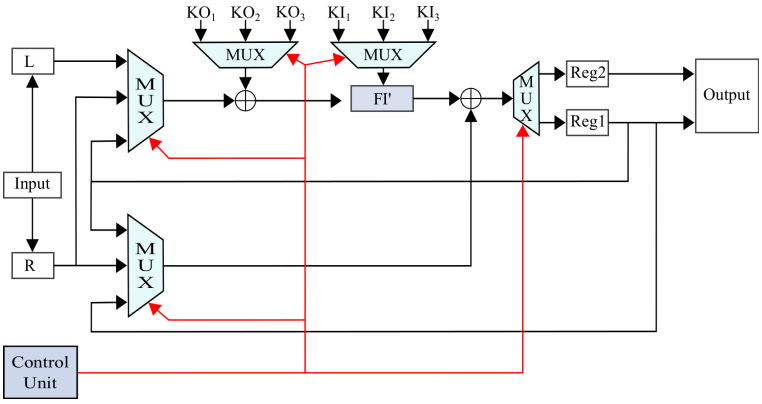


FIGURE 4 – The simplified FO' architecture.

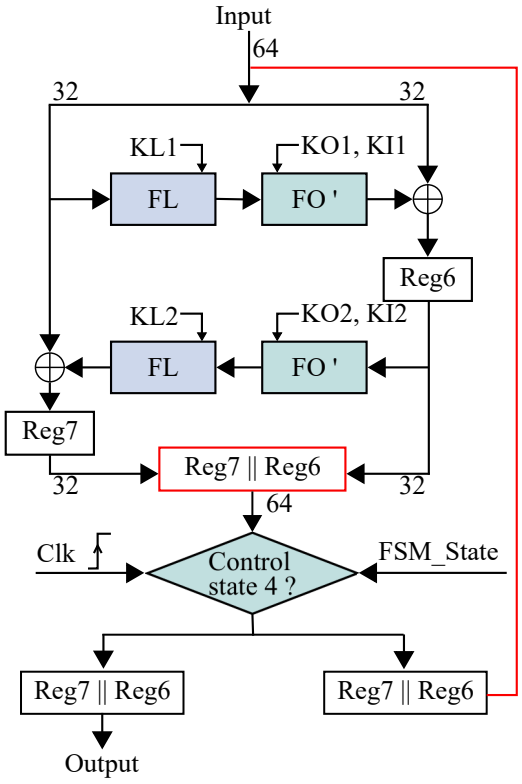


FIGURE 5 – The simplified KASUMI block.

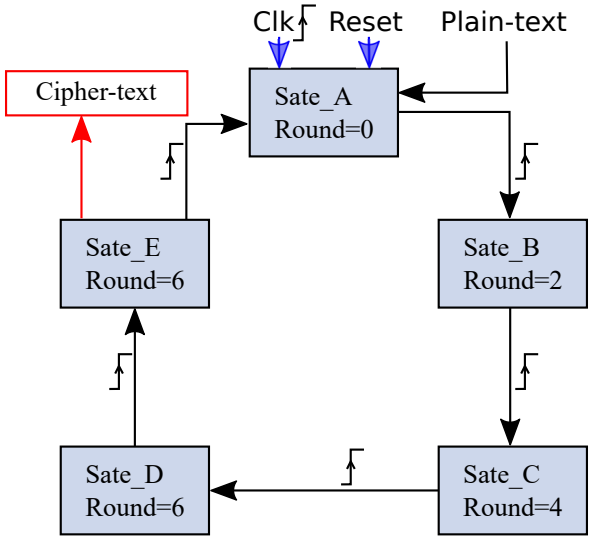


FIGURE 6 – FSM of the proposed optimized KASUMI at the level 4.

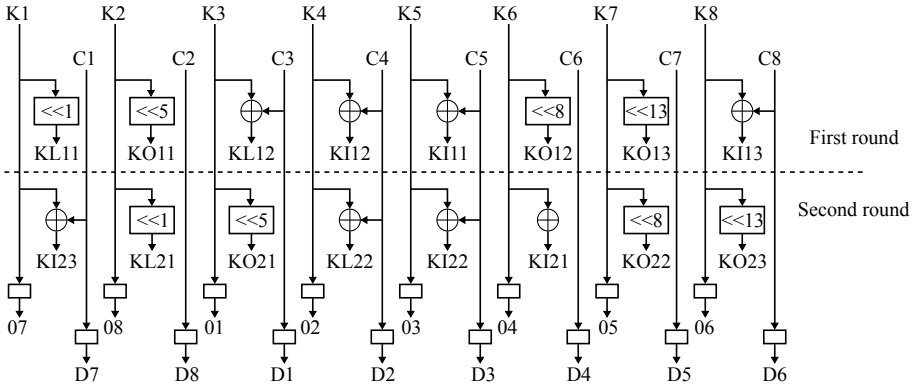
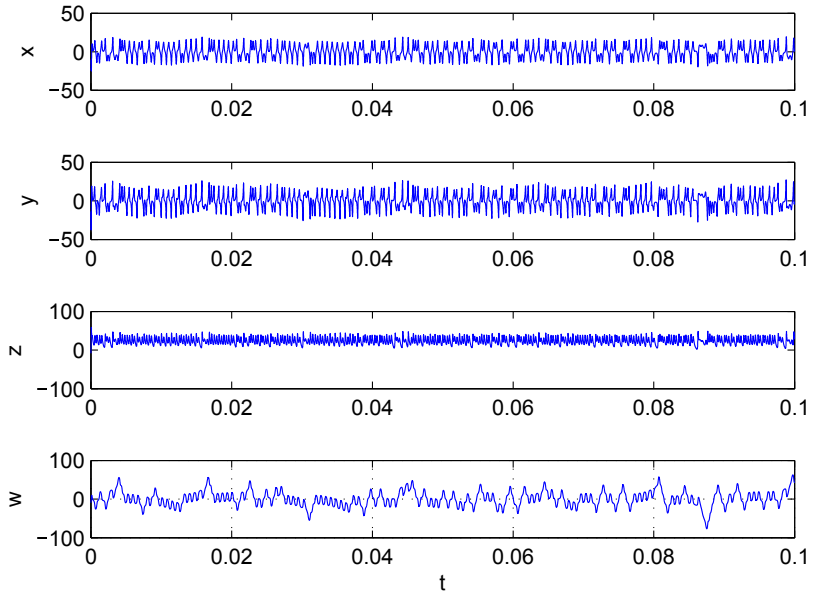
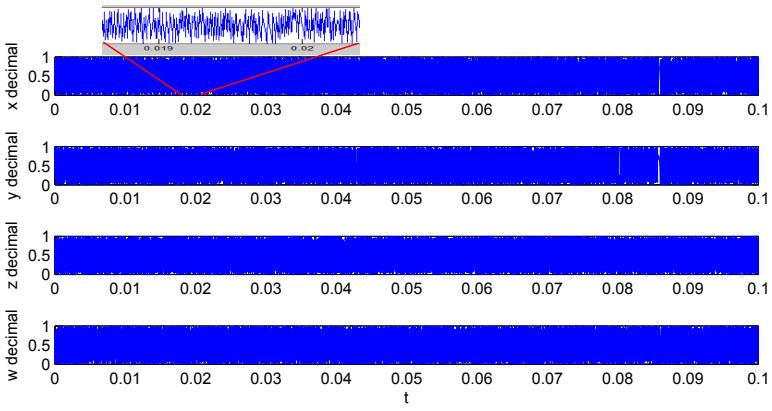


FIGURE 7 – Two round of the sub-keys scheduling.

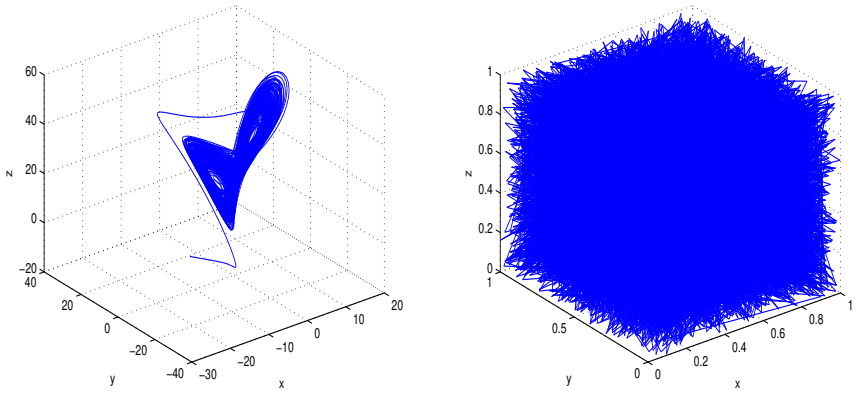


(a) Real and decimal parts of signals.



(b) Decimal part of signals.

FIGURE 8 – Chaotic signals of the used Lorenz's hyperchaotic system.



(a) Attractors (x,y,z) (real and decimal parts).

(b) Attractors (x,y,z) (decimal part).

FIGURE 9 – Attractors of the used Lorenz's hyperchaotic system.

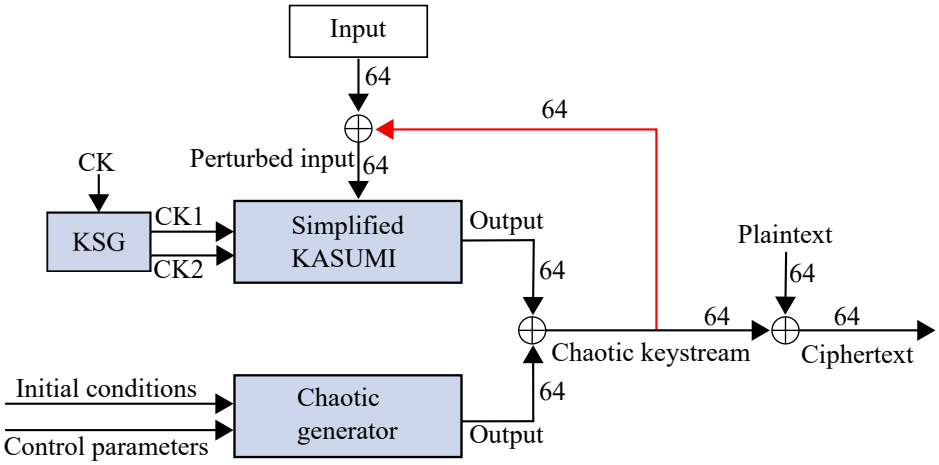


FIGURE 10 – Proposed Chaotic KASUMI stream cipher.

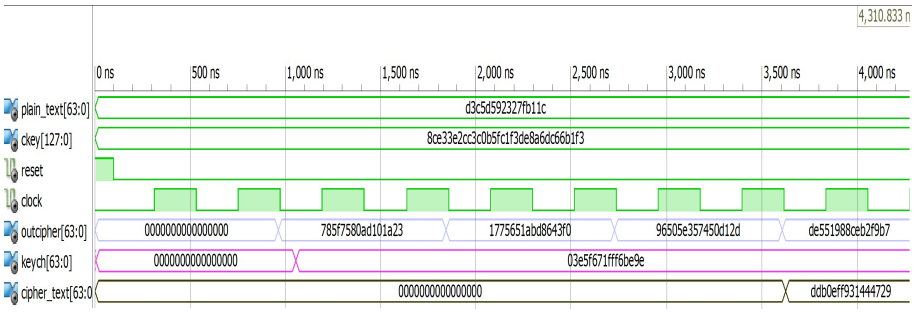


FIGURE 11 – The Chaos_KASAUMI proposed algorithm outputs.

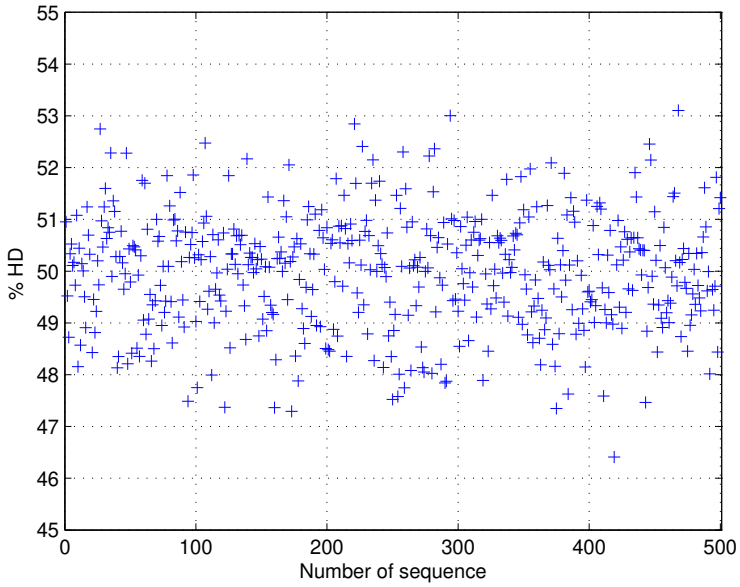


FIGURE 12 – The key sensitivity analysis results.

Device (Virtex-5)	Latency (cycle)	Frequency (Mhz)	Throughput (475.60)	Area(slice)		
				Amount	Total	Percentage
Optimized KASUMI	8	644.33	5 154.64	468	11200	4%
Chaotic-KASUMI	8	59.45	475.60	1112	11200	9%

TABLE 6 – Comparison of FPGA implementations.

Source	Latency (cycle)	Frequency (Mhz)	Throughput (Mbps)	Area (slices)	Efficiency (Kbps/slice)	Device
Work in [14]	8	20.88	167.04	1287	129.79	300E-6BG 432
Work in [15]	56	58.14	66.45	435	152.75	300E-6BG 432
Work in [15]	56	68.13	77.86	435	179	300E-8BG 432
Work in [16]	8	20	110	650	169.23	Virtex-E
Work in [17]	16	41.14	165	488	338.11	300E-8BG 432
Work in [18]	12	41.63	222	566	392.22	300E-8BG 432
Work in [19]	8	71	68	1174	483.81	N/A
Work in [20]	8	54	432	3452	125.14	300E-8BG 432
Work in [21]	16	79.45	318	625	508.8	300E-8BG 432
Work in [22]	16	96.33	385.32	448	860.08	Virtex-II
Work in [23]	54	31.93	36.09	332	108.7	300E-6BG 432
Work in [23]	54	39.4	44.54	332	134.15	300E-8BG 432
This work	8	201.86	1614.88	972	1661.14	300E-6BG 432
This work	8	251.13	2009.04	972	2066.91	300E-8BG 432
This work	8	351.49	2811.92	964	2916.93	Virtex-II
This work	8	644.33	5154.64	468	11014.19	Virtex-5

Références

[1] 3G Security ; Specification of the 3GPP confidentiality and integrity algorithms ; Document 1 : f8 and f9 specification, Technical Specification (TS) TS 35.201 V15.0.0, 3GPP (2018-06).

[2] 3G Security ; Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS ; Document 4 : Design and evaluation report, Technical Report (TR) TR 55.919 V12.0.0, 3GPP (Sep 2014-09).

[3] J. Hong, P. Sarkar, New Applications of Time Memory Data Tradeoffs, in : International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), Vol. 3788, LNCS, 2005, pp. 353–372.

[4] Y. Wentan, C. Shaozhen, Multidimensional zero-correlation linear cryptanalysis of the block cipher KASUMI, IET Information Security 10 (2016).

[5] Z. Wang, X. Dong, K. Jia, J. Zhao, Diffeential Fault Attack on KASUMI Cipher Used in GSM Telephony, Mathematical Problems in Engineering 2014 (2014) 7 pages. doi:http://dx.doi.org/10.1155/2014/251853.

[6] O. Dunkelman, N. Keller, A. Shamir, A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony, Journal of Cryptology 27 (2014).

[7] K. Jia, L. Li, C. Rechberger, J. Chen, X. Wang, Improved Cryptanalysis of the Block Cipher KASUMI, in : International Conference on Selected Areas in Cryptography SAC 2012, Lecture Notes in Computer Science, volume 7707, Windsor, Canada, 2012, pp. 222–233.

[8] E. Biham, O. Dunkelman, N. Keller, A related-key rectangle attack on the full KASUMI, in : International Conference on the Theory and Application of Cryptology and Information Security ASIACRYPT 2005, LNCS, volume 3788, 2005, pp. 443–461.

[9] M. Blunden, A. Escott, Related Key Attacks on Reduced Round KASUMI, in : International Workshop on Fast Software Encryption FSE 2001, LNCS, volume 2355, 2002, pp. 277–285.

[10] M. Salman, R. Yugitama, Amiruddin, R. F. Sari, KAMIES : Security Optimization of KASUMI Algorithm by Increasing Diffusion Level, International Journal of Security and Its Applications 12 (3) (2018) 29–46.

[11] R. Muthalagu, S. Jain, Modifying the structure KASUMI to improve its resistance towards attacks by inserting FSM and S-Box, Journal of Cyber Security Technology 2 (2018) 37–50.

[12] R. Muthalagu, S. Jain, Reducing the time required by KASUMI to generate output by modifying the FL and FI functions, Iran Journal of Computer Science 2 (2019) 33–40. doi:https://doi.org/10.1007/s42044-018-0017-2.

[13] A. Rukhin, et al, A Statistical Test Suite for the Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22, 2001 (Revised : April 2010)Http ://csrc.nist.gov/rng/SP800-22b.pdf (Revised : April 2010).

[14] K. Marinis, N. Moshopoulos, F. Karoubalis, K. Pekmestzi, On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms, in : L. N. in Computer Science 2200 (Ed.), Davida, G.I., Frankel, Y. (eds.) ISC 2001, Springer, Heidelberg, 2001, pp. 248—265.

[15] A. Satoh, S. Morioka, Small and High-Speed Hardware Architectures for the 3GPP Standard Cipher KASUMI, in : L. N. in Computer Science 2433 (Ed.), Chan.A.H, Gligor.V.D. (eds.) ISC 2002, Springer, Heidelberg, 2002, pp. 48—62.

[16] H. Kim, Y. Choi, M. Kim, H. Ryu, Hardware implementation of the 3GPP KASUMI crypto algorithm, in : ITC-CSCC 2002, 2002, pp. 317—-320.

[17] T. Balderas, R. Cumplido, An Efficient Hardware Implementation of the KASUMI Block Cipher for Third Generation Cellular Networks, in : GSPx, 2004.

[18] T. Balderas, R. Cumplido, An Efficient Reuse-Based Approach to Implement the 3GPP KASUMI Block Cipher, in : ICEEE 2004, 2004, pp. 113– –118.

[19] H. Kim, S. Lee, Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System, IEEE Transactions on Consumer Electronics 50 (1) (2004) 214—224.

[20] P. Kitsos, M. D. Galanis, O. Koufopavlou, High-speed hardware implementations of the KASUMI block cipher, in : ISCAS 2004, Vol. 2, 2004, pp. 549—552.

[21] T. Balderas, R. Cumplido, High Performance Encryption Cores for 3G Networks, in : DAC 2005, 2005, pp. 240—-243.

[22] T. Balderas, R. Cumplido, C. Feregrino-Uribe, On the design and implementation of a RISC processor extension for the KASUMI encryption algorithm, Computers and Electrical Engineering 34 (6) (2008) 531—-546.

[23] Y. Dai, I. Kouichi, Y. Jun, A Very Compact Hardware Implementation of the KASUMI Block cipher, in : LNCS 6033, 2010, pp. 293—307.

[24] 3G Security ; Specification of the 3GPP Confidentiality and Integrity Algorithms ; Document 2 : KASUMI specification, Technical Specification (TS) TS 35.202 V15.0.0, 3GPP (Jun 2018-06).

[25] 3G Security ; Specification of the 3GPP Confidentiality and Integrity Algorithms ; Document 3 : Implementors Test Data, Technical Specification (TS) TS 35.203 V15.0.0, 3GPP (2018-06).

[26] M. Matsui, New Block Encryption Algorithm MISTY, in : Proceedings of the 4th International Fast Software Encryption Workshop FSE'97, LNCS 1267, Springer-Verlag, 1997, pp. 54–68.

[27] P. Kitsos, N. Sklavos, O. Koufopavlou, An End-to-End Hardware Approach Security for the GPRS, in : Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference IEEE Cat No04CH37521, Dubrovnik, Croatia, Croatia, 2004.

[28] C. Li, Y. Zhang, E. Yong Xie, When an attacker meets a cipher-image in 2018 : A year in review, Journal of Information Security and Applications 48 (2019) 2214–2126. doi:https://doi.org/10.1016/j.jisa.2019.102361.

[29] C. Li, B. Feng, S. Li, J. Kurths, G. Chen, Dynamic analysis of digital chaotic maps via state-mapping networks, IEEE Transactions on Circuits and Systems I : Regular Papers 66 (6) (2019) 2322–2335. doi:10.1109/TCSI.2018.2888688.

[30] M. Madani, I. Benkhaddra, C. Taougast, S. Chitroub, L. Sieler, FPGA Implementation of an enhanced SNOW-3G Stream Cipher based on a Hyperchaotic System, in : The 4th international conference on Control, Decision and Information Technologies (CoDIT'17), IEEE Conference Publications, 2017.

[31] E. Lorenz, Deterministic nonperiodic flow, Journal of Atmospheric Sciences 1963 20 (1963) 130–141.

[32] M. Madani, I. Benkhaddra, C. Taougast, S. Chitroub, L. Sieler, Digital Implementation of an Improved LTE Stream Cipher SNOW-3G based on Hyperchaotic PRNG, Security and Communication Networks 2007 (2017). doi:https://doi.org/10.1155/2017/5746976.

[33] S. Sadoudi, C. Tanougast, M. S. Azzaz, et al., Design and FPGA implementation of a wireless hyperchaotic communication system for secure real-time image transmission, Eurasip Journal on Image and Video Processing, Springer (43) (2013) 1–18. doi:10.1186/1687-5281-2013-43.

[34] J.-P. Demailly, Analyse numérique et équations différentielles, EDP Sciences 4, Collection Grenoble Sciences, 2006.

[35] M. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, R. López-Gutiérrez, Implementation of an improved chaotic encryption algorithm for real-time embedded systems by using a 32-bit microcontroller, Microprocessors and Microsystems (45) (2016) 297–309. doi:10.1016/j.micpro.2016.06.004.

[36] M. Sano, Y. Sawada, Measurement of the lyapunov spectrum from a chaotic time series, Phys. Rev. Lett. 55 (1985) 1082–1085.
URL https://link.aps.org/doi/10.1103/PhysRevLett.55.1082

[37] Z. Sandor, B. Erdi, A. Szell, B. Funk, The relative lyapunov indicator: An efficient method of chaos detection, Celestial Mechanics and Dynamical Astronomy 90 (2004) 127–138. doi:10.1007/s10569-004-8129-4.
URL https://doi.org/10.1007/s10569-004-8129-4

[38] K. Ramasubramanian, M. S. Sriram, A comparative study of computation of lyapunov spectra with different algorithms, Physica D 139 (2000) 72–86.
URL https://doi.org/10.1016/S0167-2789(99)00234-1

[39] X. Wang, M. Wang, A hyperchaos generated from lorenz system, Physica A 387 (2008) 3751–3758.
URL https://doi.org/10.1016/j.physa.2008.02.020

[40] D. Han, L. Min, G. Chen, A Stream Encryption Scheme with Both Key and Plaintext Avalanche Effects for Designing Chaos-Based Pseudorandom Number Generator with Application to Image Encryption, International Journal of Bifurcation and Chaos 26 (5) (May 2016). doi:10.1142/S0218127416500917.

[41] C. Shannon, Communication Theory of Secrecy Systems, Bell Systems Technical Journal 28 (1949) 656–715.

[42] Xilinx, Virtex™-E 1.8 V Field Programmable Gate Arrays, Production Product Specification, DS022-1 (v3.0) March 21, 2014.

[43] Virtex 5 FPGA Configuration User Guide, ug702, Xilinx.

[44] Virtex devices specification, http://www.xilinx.com/.