Experimental and Timing Analysis Comparison of FPGA Trivium Implementations and their Vulnerability to Clock Fault Injection

F.E. Potestad-Ordóñez, C.J. Jiménez-Fernández, M. Valencia-Barrero
Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC/Universidad de Sevilla)
Email: {potestad,cjesus}@imse-cnm.csic.es,{manolov}@dte.us.es

***Abstract*—The security of cryptocircuits is today threatened not only by attacks on algorithms but also, and above all, by attacks on the circuit implementations themselves. These are known as side channel attacks. One variety is the Active Fault Analysis attack, that can make a circuit vulnerable by changing its behavior in a certain way. This article presents an experimental fault insertion attack on an FPGA implementation of the Trivium stream cipher. It also compares the faults introduced with the faults expected after a timing analysis. The results show that this implementation is vulnerable to such attacks, and also that it is not possible to estimate the position of the inserted faults by means of timing analysis.**

***Keywords*—Fault Attack, Stream Cipher, Trivium, FPGA implementation.**

I. Introduction

Improvements in cryptographic algorithms have led to the evolution of more sophisticated attacks aimed at breaking their security. Apart from classic attacks on algorithms it is now necessary also to consider a second type of attack which focusses on the physical hardware device that implements the cryptographic algorithm. With the information obtained from circuit operation, such as power consumption and electronic radiation, it is possible to threaten the safety of the whole cryptosystem. These kinds of attacks are known as Side Channel Attacks [1], and can take two forms. Firstly, there are Side Channel Analysis attacks, such as Correlation Power Analysis (CPA) and Differential Power Analysis (DPA) attacks, which attack the circuit by measuring power consumption during circuit operation. And secondly, there are Active Fault Analysis attacks, like Differential Fault Analysis (DFA) and Differential Fault Intensity Analysis (DFIA) attacks, which involve injecting faults during the circuit operation by modifying the operating conditions.

One of the earliest studies into Active Fault Analysis was presented by Boneh et al, who described a fault attack on the RSA cryptosystem [2]. Since then, the same technique has been applied successfully in many other cryptographic algorithms, including symmetric key systems like block ciphers and stream ciphers (including Trivium).

Some analyses of the Trivium stream ciphers vulnerability to Active Fault Analysis attacks can be found in literature [3]-[8], but none of them evaluate Triviums vulnerability on a specific hardware implementation. This paper therefore presents an experimental analysis of the behavior of FPGA Trivium cipher implementations when subjected to fault injection through variation of the clock signal. It also presents a comparative analysis of the experimental results obtained after the attack and the expected results obtained through simulation and timing analysis: that is to say, the fault positions of the Trivium inner state obtained experimentally and the fault positions predicted by the timing analysis. The results show the vulnerabilities of these implementations against this type of attack and demonstrate the impossibility of determining fault injections from simulation results.

The rest of the paper is organized as follows. Section II introduces the architecture of the Trivium stream cipher, reviews the different theoretical Fault Injection techniques applicable to the Trivium cipher and explains the experimental fault injection technique used. Section III describes the FPGA implementation of the Trivium and the fault injection system. Section IV presents the results obtained when our system was used against the Trivium stream cipher and Section V compares the experimental results and the results predicted by the timing analysis and simulation. Finally, Section VI presents some conclusions.

II. Fault Attacks on Trivium Stream Cipher

A. *Trivium Stream Cipher*

The Trivium stream cipher [9], one of the finalists in the eSTREAM project, is a synchronous cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initialization vector (IV). Its architecture is based on three shift registers, totaling 288 bits, and combinational logic to provide non lineal feedback. Like other synchronous stream ciphers, the algorithm needs to be initialized by loading the 288 bits of the shift registers (inner state) with one secret key, one initialization vector, zeros and ones. Before generating a valid key stream, the cipher needs to run for 1152 clock cycles. From then on, it starts generating a valid pseudorandom bit sequence (key stream).

The 288 bits of the inner state are distributed in three shift registers of different lengths. The first shift register has 93 bits, the second 83 bits and the third 111 bits. The feedback for each shift register is generated with AND and XOR operations. The key stream is the result of XOR operations on some of the bits in the shift register. Fig. 1 shows a schematic representation of the Trivium cipher.

B. Theoretical Fault Injection in Trivium

The Trivium stream cipher has been studied and analyzed theoretically to determine its vulnerability to fault injection side channel attacks. One of the first analyses was the work presented by Hojsík and Rudolf [3], who carried out a Differential Fault Analysis (DFA) attack. The DFA technique is a side channel attack in which an attacker is able to inject a fault into the encryption or decryption process. In other words, a fault is injected into the ciphers inner state. This technique assumes that an attacker is able to change only one bit of the inner state. In [3], two different techniques based on the same aim but using different mathematical formulations were presented. The second technique was the more efficient, successfully retrieving the secret key with 43 fault injections. The same authors later presented a new attack called the Floating Fault Analysis of Trivium [4], in which they reduced the number of fault injections to an average of 3.2. Yupu, Juntao, Qing and Yiwei [5] used those studies to improve the cryptographic analysis and retrieve the secret key and initialization vector with an average of only 3.7 fault injections.

In [6], an Improved Differential Fault Analysis of Trivium was introduced, capable of retrieving the secret key with two fault injections over the inner state. A new analysis, called the Mutant Differential Fault Analysis of Trivium (MDFA) based on [3], was presented in [7], where it was affirmed that it is possible to break the system with only one fault injection. Another reference was the work entitled Improved Multi-Bit Differential Fault Analysis of Trivium [8], which showed an improvement on the system constraints described in [5] and made it possible to attack the system using different fault models, injecting a fault in an unknown cycle. In this attack, the secret key was retrieved with four fault injections.

All these references share the same assumption: that in order to retrieve the secret key and initialization vector it is necessary to inject only one fault bit into any of the three registers of the stream cipher. According to this assumption, if an attacker is able to inject that single fault bit, then the cipher implementation will be vulnerable to fault injection attacks. However, none of the papers mentioned experimentally measured the possibility of such a fault injection.

C. Fault Injection Technique

As explained above, an experimental mechanism had to be developed to inject faults into Trivium implementations and measure the number of faults injected. Many practical techniques exist for injecting faults over different devices. As described in [10], faults can be injected through variations in supply voltage, temperature, white light, laser beams, X-rays, ion beams and variations in the external clock. The technique chosen in the tests was based on the insertion of short pulses in the clock signal. This technique increases the clock frequency in one specific, preselected clock cycle: thereby injecting a fault and making the device unable to operate correctly.

The technique we used was studied, among others, in [11], with theoretical analysis and practical experiments on an AES block cipher, and in [12], which showed a fault injection attack with short pulses in the clock signals of different block ciphers. In [13] the same technique was used to attack an FPGA implementation of the AES block cipher. Taking into account the theoretical models of the Trivium ciphers vulnerability to fault injections reported in literature and described in Section II-B, the mechanism we developed had to be able to insert only one fault in the inner state to be effective.

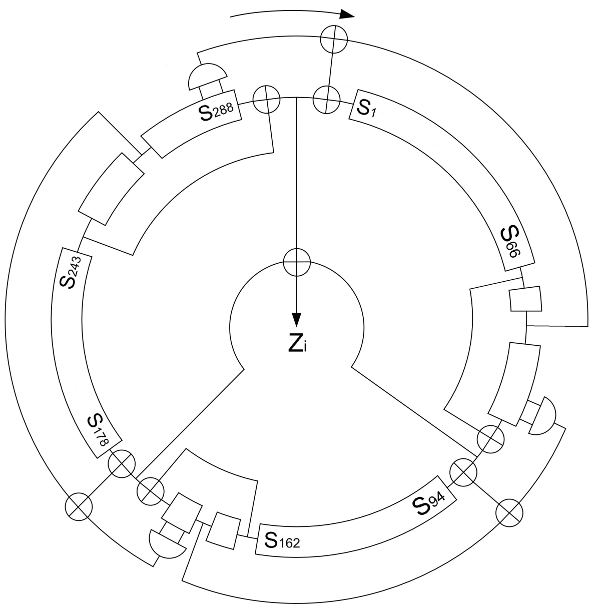

Fig. 1. Schematic representation of the Trivium stream cipher.

III. FPGA IMPLEMENTATION DESCRIPTION

A. Fault Injection System Design

To inject faults over the Trivium stream cipher, a system was designed that would generate a pulse in the clock signal. This system was presented in [14]. Since in order to generate a short pulse in the clock signal it was necessary to know the maximum frequency at which the cipher implementation was able to work, the specific characteristics of FPGA implementations had to be taken into account. On one hand, FPGA uses a dedicated line for clocks, with the aim of guaranteeing high efficiency in the implemented circuit. This produces very low skew and low delay in the clock signal. As explained previously, stream ciphers comprise flip-flops connected in serial mode with some combinational logic to generate the feedback signals and the key stream. This structure is very simple, so FPGA stream cipher implementations are very fast and are able to work at high frequencies. In our tests, Trivium stream ciphers could operate at the maximum frequency supported by the FPGA.

To inject a fault into the stream cipher, it is therefore necessary to induce a short pulse in the clock signal at a frequency higher than the maximum operating frequency of the FPGA (as explained above). This forces the system to operate with frequencies above the FPGAs maximum operating frequencies, but at the same time, the maximum frequency cannot be exceeded too much because, if it is, the FPGA could filter out the short pulse.

The system to be developed posed three problems: *a)* frequencies well above the maximum could not be used, because they would be filtered by the FPGA, *b)* although a

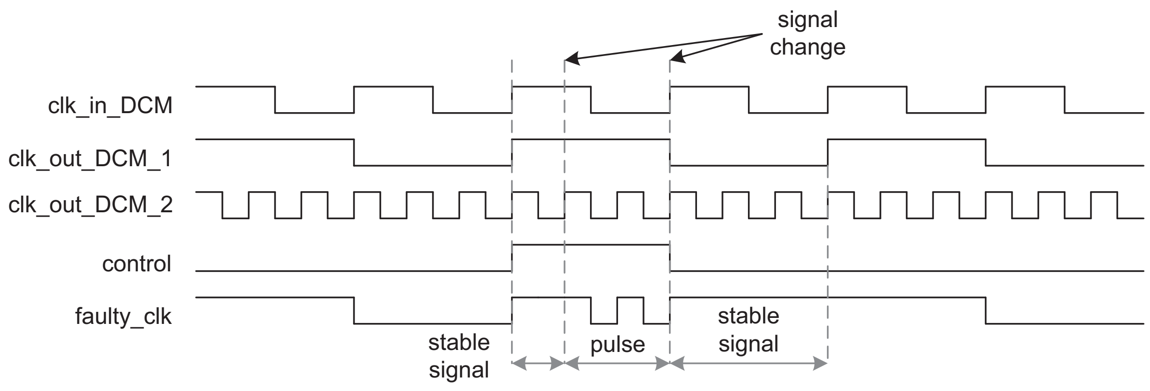

Fig. 2. Timing diagram to obtain the short pulse.

frequency slightly above the maximum was needed to insert a fault in the Trivium, the overall system of control and measurement had to operate at a frequency much lower than the maximum, and *c)* it had to be ensured that the faults were being introduced only into the Trivium and not into the rest of the system.

With these considerations in mind, different tests were carried out to determine the maximum operating frequency of the Trivium stream cipher implemented on FPGA. The tests involved increasing the operating frequency to the maximum level at which the cipher would work. The device used was a Spartan 3E XC3S500E with a maximum working frequency of 311 MHz. The test results showed that the Trivium stream cipher implemented on this FPGA could work at up to 316 MHz. At higher frequencies, the cipher began to work erroneously. The short pulse frequency chosen to inject faults over the cipher was therefore 316.66 MHz. This frequency satisfied all the pre-requisites, being slightly higher than the maximum frequency and thus suitable for fault injection, but without causing the whole system to malfunction.

Different possible fault frequencies were studied until the optimum frequency was found. The frequency finally selected as optimal was the frequency at which it was possible to enter more effective faults. If that frequency was slightly increased, the number of injected faults also increased, thus reducing effectiveness.

The system for injecting faults over the stream ciphers comprised a finite state machine that controlled the running status of the ciphers, like the load of secret key and initialization vector, normal running by generating *n* bits of the key stream and reset the cipher. The state machine controlled the cycle where the short pulse was added to inject the fault in the inner state, and made it possible to select the clock cycle where the inner state was sampled. To analyze the stream cipher inner states, the cipher design was modified to allow the inner state register to be sampled and analyzed in search of fault injections. The state machine also controlled the subsystem for generating the short pulse. To insert the short pulse in the clock line, the state machine chose between two clock signals: one of them was an optimum operating frequency for the whole system and data sampling process, and the other was the frequency used, only in one clock cycle, to inject faults into the inner state. The clock frequency of the state machine was lower than the maximum operating frequency of the FPGA, thus ensuring that the control system worked without any errors.

A short pulse can be generated in a signal in different ways. One way is by means of a logical operation between two shifted signals. Due to the particular structure of FPGA devices, however, it is very difficult to control delays in combinational, generated signals with the precision needed for this application. This method is therefore not suitable for FPGA. Another method is to generate a high frequency clock and switch it with a low clock frequency. This technique can be implemented in FPGA devices thanks to their specific resources, to generate high frequency clock signals and choose between two clock signals.

To generate the short pulse on the clock signal, a Digital Clock Manager (DCM) (available in Xilinx FPGA devices) was used. The DCM can generate clocks with different frequencies from an input clock, allowing generation not only of clocks with lower frequencies than the input clock, but also of clocks with higher frequencies. In the system we developed, the DCM was used to generate a clock with an optimal frequency for the whole system and for Trivium, and another clock with a frequency above the maximum frequency of the device. Considering that the maximum frequency of the Spartan 3E XC3S500E device is 311 MHz, a clock of 316.66 MHz was generated because in tests it was found that Trivium fails at that frequency.

Several tests were carried out to verify that the clock signal of 316.66 MHz was being generated correctly. With these two clock signals, one with optimal frequency and the other with the fault frequency, the entire system could be controlled and the short pulse generated that would allow the fault to be injected into the stream cipher. Switching between the two clock signals was done using a clock signal multiplexer, making it possible to switch between two clock signals without generating additional pulses. Fig. 2 shows the timing diagram for the clock signal generation with the short pulse. For only one pulse to be introduced into the fault clock when switching between the two clocks, the frequency of the optimal clock had to be a quarter of the fault frequency.

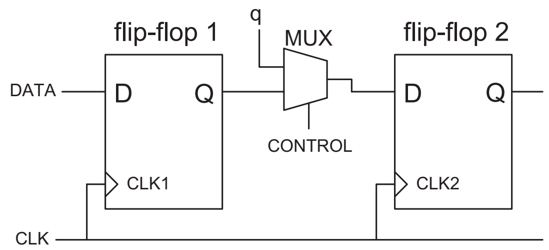

Fig. 3. A schematic image of the Trivium Stream Cipher inner state.

B. Trivium implementation on FPGA

After explaining the fault injection system for the Trivium Stream Cipher, it is necessary to take into account the characteristics of the Trivium implementation on FPGA. These characteristics mainly affect the way the cipher loads the secret key and initialization vector. In this case, loading is done in parallel. Subsequently, in the FPGA implementation, a multiplexer is placed between each flip-flop in the inner state. The shift register that makes up the inner state is not a serial flip-flop to flip-flop structure, but rather constitutes a flip-flop to multiplexer to flip-flop structure. Fig. 3 shows a schematic representation of two flip-flops in this kind of implementation, with the multiplexer visible between them. One result of this implementation is that the shift registers are slower than the flip-flop to flip-flop structure due to the fact that the output data is not immediately available at the input of the next flip-flop. The Trivium design was modified to provide access to the inner state in order to analyze the result of the fault injections, but this modification implied no change in the implementation.

Other factors taken into account were the influences of placement and routing on the fault injection. Differences in the placement and routing of a circuit cause different internal delay times, which may introduce faults in different positions.

IV. Results

The fault injection system was designed in VHDL and implemented with the ISE 14.7 Xilinx tool on a Spartan 3E XC3S500E FPGA. The data was sampled with ChipScope Pro Analyzer. As explained above, the design of the Trivium cipher was modified for the tests in order to access the ciphers inner state register. One additional register was introduced and used to copy the Trivium inner state for later analysis. To analyze routing dependency and behavior against the same attack, three copies of the Trivium cipher were implemented in the device in parallel mode. The short pulse was injected in two of them while the third remained fault-free. In other words, there were two ciphers with fault injection and another that worked properly. The fault-free inner state of the third Trivium was compared with the inner states of the other two Triviums in order to detect the fault injections.

As mentioned earlier, to test the systems vulnerability to fault attack only one fault needed to be injected into the inner state of the stream cipher. With this in mind, a fault injection would be considered effective or successful when a fault was injected into any of the ciphers three shift registers and this injection comprised only one wrong bit. In cases where the fault injections produced more than one wrong bit, the injection in question would be considered as an unsuccessful attack.

Table I shows the bit positions of the cipher shift registers where faults were injected during the tests. Results are shown for four random pairs of secret keys and IVs, and the pulse was injected in four different clock cycles. Trivium 1 and Trivium 2 represent the two implementations of the Trivium where the pulse was introduced in the clocks. When no number appears in a cell, this means that no error was injected in the cipher shift register for that key and IV pair in that clock cycle. When more than one number appears, it means that in this case the fault was injected into the shift register in the same cipher at different positions and at the same time. These cases represent attacks where the fault injection was deemed unsatisfactory due to the fact that more than one fault was injected. As can be seen, the bit positions that tended to fail were the same in the different tests, thus demonstrating that it is the positions in the inner state that make the stream cipher vulnerable. With this in mind, these components were analyzed in order to find the relationships between vulnerability and routing, signal delay and signal skew.

To analyze the repeatability of the fault injection, approximately 1600 tests were carried out. The results obtained show that the faulty bit positions tended to be the same for all tests under these conditions. One point of interest is that the positions that failed were those nearest to the bits used by the stream cipher to do logic operations or provide key stream generation feedback.

Tables II, III and IV show the different positions of the faults introduced for the same keys and IVs and for different insertion clock cycles. It can be seen that, under the same conditions, the ciphers display different types of vulnerability. There are evidently different types of fault injections in the same case, but those injections show it is always the same flip-flops that tend to fail.

The results show that placement and routing affect the positions where the faults are introduced, and therefore, by extension, also affect the vulnerability of the implementation. Another interesting point extracted from the results is that it was possible to introduce an effective fault for all four pairs of keys and IVs presented. In two of these cases, Key 2 IV 1 in Trivium 2 and Key 2 IV 2 in Trivium 2, the fault was injected only in insertion cycle 1500, but in others cases it was possible to inject the fault in several cycles.

V. Timing analysis and comparison with the results obtained experimentaly

With the aim of designing an implementation capable of resisting the attacks presented in this paper, an analysis was made of delays in the bits that failed with the post-route data.

The flip-flops that failed were expected to be those with the greatest delays in their input paths. Using timing restrictions on those paths would therefore reduce their vulnerability. The timing analysis with the post-route data was carried out using both static analysis tools and post-route timing simulation. In the static analysis, the delay in the path from the output of one flip-flop to the input of the next was studied, taking into account clock skew and flip-flop setup time. In the

TABLE I. FAULTY BITS POSITIONS ON THE CIPHERS FOR EACH KEY/IV PAIR AND INSERTION CYCLE.

	Key 1 IV 1		Key 1 IV 2		Key 2 IV 1		Key 2 IV 2	
Insertion cycles	Trivium 1	Trivium 2	Trivium 1	Trivium 2	Trivium 1	Trivium 2	Trivium 1	Trivium 2
1200	2	-	3	3	96	-	-	-
1300	3/2	3	3	3	3	-	-	-
1500	96/3/2	3/2	2	-	3	3	3	3
1750	2	2	3	-	2	-	2	-

TABLE II. POSITIONS OF THE INJECTED FAULTS TYPE 1, TYPE 2 AND TYPE 3 ON EACH CIPHER.

Key 1 IV 2	Fault Type 1		Fault Type 2		Fault Type 3	
Insertion cycles	Trivium 1	Trivium 2	Trivium 1	Trivium 2	Trivium 1	Trivium 2
1200	96/3/2	3	3/2	3	3	3
1300	3	3	-	-	-	-
1500	2	-	-	-	-	-
1750	3	-	-	-	-	-

TABLE III. POSITIONS OF THE INJECTED FAULTS TYPE 1 AND TYPE 2 ON EACH CIPHER.

Key 1 IV 1	Fault Type 1		Fault Type 2	
Insertion cycles	Trivium 1	Trivium 2	Trivium 1	Trivium 2
1200	2	-	-	-
1300	3/2	3	3/2	-
1500	96/3/2	-	96/3	3/2
1750	2	2	2	-

TABLE IV. POSITIONS OF THE INJECTED FAULTS TYPE 1 AND TYPE 2 ON EACH CIPHER.

Key 2 IV 1	Fault Type 1		Fault Type 2	
Insertion cycles	Trivium 1	Trivium 2	Trivium 1	Trivium 2
1200	96	-	-	-
1300	3	-	-	-
1500	3/2	3	3	3
1750	2	-	-	-

timing simulation analysis, it was possible to measure the time between the change at the output of one flip-flop and the change at the input of the next flip-flop.

The static timing analysis generated a report with the highest delay paths. According to this report, the path with the highest delay was the connection between the output of flip-flop 218 and the input of flip-flop 219 in Trivium 2. This test result suggested that the first candidate to fail when a short pulse was injected in the clock line would be flip-flop 219. However, as can be seen, this flip-flop was not affected by the attack.

The timing analysis results presented in Tables V, VI and VII are organized by bit pairs. The timing and post-route analysis needed to be done between the faulty bit (flip-flop X) and the previous bit (flip-flop $X-1$). That is to say, if a fault was injected into flip-flop X, that fault could be due to the routing or to skew and delay in the signals between the previous bit and the faulty bit. For example, if previous tests showed that position number 2 tended to fail, the timing relationship between the previous flip-flop (bit 1) and the faulty flip-flop (bit 2) was analyzed.

Table V shows the results of the timing analysis of the bit pairs that failed in Trivium 1. The results were obtained from static and dynamic analysis of the paths between the bits represented. The Max Delay column shows the maximum delay of the path between the output of one flip-flop and the input of the next one, as obtained by static analysis. The column marked O-I Delay shows the delay time between the output of a flip-flop and the input of the next, as measured by post-route simulation (or dynamic delay). The aim of this analysis was to check the dependency between the faults introduced and the delay in the paths. Table V shows the delays obtained for the input paths of the bits where errors were injected: bits 2, 3 and 96. But the table also shows the delays for two paths (between flip-flops 125 and 126 and flip-flops 96 and 97) that never had failures. The delays in the input signals of the bits that failed were expected to be greater than the delays of the bits that did not fail, but the data obtained by static timing analysis and timing simulations shows that in Trivium 1, the delay from bit 96 to bit 97 was greater than the delay in the faulty bits.

The same analysis was carried out for Trivium 2, and the results are shown in Table VI. The faulty bits for Trivium 2 were the same as those that failed in Trivium 1, although their input delays were slightly different. Like Table V, Table VI shows the delays between flip-flops 125 and 126 and flip-flops 96 and 97. The delay from bit 125 to bit 126 was greater in Trivium 2, but bit 126 did not fail in any of the tests. As can be seen, the two Triviums produced different results due to the different routing on the FPGA, but the behavior of the faulty bits was quite similar in both of them.

Table VII shows the delay time of the clock signal for each flip-flop shown in Tables V and VI, for Trivium 1 and Trivium 2. In all cases this delay was much smaller than the delays in input signals. The results show that, for both Triviums, the bit pairs that did not display any vulnerabilities had delays that were higher than or similar to those of pairs vulnerable to glitch insertion in the clock line.

TABLE V. ANALYSIS OF FAULTY BIT TRANSITION TIMES AND COMPARISON WITH NON FAULTY PAIR BITS FOR TRIVIUM 1.

		Static Analysis	Post-route Analysis
From Bit	To Bit	Max Delay (ns)	O-I Delay (ns)
1	2	2.215	1.490
2	3	2.253	1.529
95	96	1.968	1.243
96	97	3.047	2.322
125	126	2.154	1.434

TABLE VI. ANALYSIS OF FAULTY BIT TRANSITION TIMES AND COMPARISON WITH NON FAULTY PAIR BITS FOR TRIVIUM 2.

		Static Analysis	Post-route Analysis
From Bit	To Bit	Max Delay (ns)	O-I Delay (ns)
1	2	2.503	1.766
2	3	2.217	1.488
95	96	2.449	1.725
125	126	3.143	2.423
96	97	1.997	1.253

In the light of these analyses, it can be concluded that it is not possible to use timing analysis to estimate the bits that will fail after a fault attack based on the alteration of the clock signal. The behavior of the implemented circuit when subject to such an attack can only be shown by experimental measurements. The introduction of timing restrictions does nothing to reduce the implementations vulnerability.

VI. CONCLUSIONS

This paper presents an experimental fault attack, carried out by changing the clock signal on FPGA implementations of the Trivium stream cipher. This type of attack endangers the security of the Trivium cipher because it can modify only one bit of the inner state. Experimental results were compared with the results expected from the timing analysis of the implemented circuit. The comparison showed that the flip-flops in which faults were introduced experimentally did not correspond to the flip-flops with the greatest delays in their input signals and, therefore, did not correspond to the bits which would be expected to fail first. These results show that pre-implementation analyses are not valid when designing a robust implementation resistant to fault injections involving the insertion of a pulse in the clock line. The only valid analysis capable of determining the vulnerabilities of Trivium ciphers implemented on FPGA is therefore to subject the system to fault injection and identify its weakest points in experimental mode, prior to designing specific circuits to eliminate this vulnerability.

ACKNOWLEDGMENT

This work was partially supported by the Spanish Ministry of Economy and Competitiveness (with support from the European Regional Development Fund - FEDER) under contracts CITIES (TEC2010-16870), CESAR (MEC TEC2013-45523-R), LACRE (CSIC 201550E039) and MISAL (CSIC).

TABLE VII. ANALYSIS OF CLOCK DELAY FOR FAULTY BITS AND COMPARISON WITH NON FAULTY BITS.

	Trivium 1	Trivium 2
Bit	Clock Delay (ps)	Clock Delay (ps)
2	117	149
3	122	136
96	142	163
126	132	133
97	178	165

REFERENCES

[1] S. Patranabis, *et al.*, ”Using State Space Encoding To Counter Biased Fault Attacks on AES Countermeasures”, *Constructive Side-Channel Analysis and Secure Design* (*COSADE'15*), 2015.

[2] D. Boneh, R.A. DeMillo, R.J. Lipton, ”On the Importance of Checking Cryptographic Protocols for Faults”, *Lecture Notes in Computer Science*, vol. 1233, pp. 37-51, 1997.

[3] M. Hojsík and B. Rudolf, ”Differential fault analysis of Trivium”, *Fast Software Encryption (FSE'08)*, pp. 158-172, 2008.

[4] M. Hojsík and B. Rudolf, ”Floating fault analysis of Trivium”, *International Conference on Cryptology in India (INDOCRYPT'08)*, pp. 239-250, 2008.

[5] Y. Hu, *et al.*, ”Fault analysis of Trivium”, *Designs, Codes and Cryptography*, vol. 62, no 3, pp. 289-311, 2012.

[6] M.S.E. Mohamed and J. Buchmann, ”Improved differential fault analysis of Trivium”, *Constructive Side-Channel Analysis and Secure Design* (*COSADE'11*), pp. 147-158, 2011.

[7] M.S.E. Mohamed and J. Buchmann, ”Mutant Differential Fault Analysis of Trivium MDFA”, *International Conference on Information Security and Cryptology (ICISC'14)*, pp. 433-446, 2014.

[8] P. Dey and A. Adhikari, ”Improved Multi-Bit Differential Fault Analysis of Trivium”, *International Conference in Cryptology in India (INDOCRYPT'14)*, pp. 37-52, 2014.

[9] C. De Canniere and B. Preneel, ”Trivium, A Stream Cipher Construction Inspired by Block Cipher Design Principles”, eSTREAM, ECRYPT Stream Cipher Project.

[10] H. Bar-El, H. Choukri, D. Naccache, *et al.*, ”The Sorcerer's Apprentice Guide to Fault Attacks”, Proc. of IEEE, vol. 94, pp. 370-382, 2006.

[11] M. Agoyan *et al.*, ”When clocks fail: On critical paths and clock faults”, *Smart Card Research and Advanced Application (CARDIS'10)*, pp. 182-193, 2010.

[12] T. Fukunaga, J. Takahashi, ” Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers”, *Fault Diagnosis and Tolerance in Cryptography (FDTC'09)*, pp. 84-92, 2009.

[13] Y. Ren, A. Wang, L. Wu, ”Transient-Steady Effect Attack on Block Ciphers”, *Cryptographic Hardware and Embedded Systems (CHES'15)*, pp. 433-450, 2015.

[14] F.E. Potestad-Ordóñez, C.J. Jiménez-Fernández, M. Valencia-Barrero, ”Fault Attack on FPGA implementations of Trivium Stream Cipher”, *International Symposium on Circuits and Systems (ISCAS'16)*, pp. 562-565, 2016.