

and 0.2 switching activities, the energy saving of TSC decreases since the encoding and decoding switching energy overhead increases while the probability that two encoded signals switch simultaneously is still low. As the switching activity increases and the probability of simultaneous switching increases, the energy saving of TSC increases due to the large reduction in the number of switching wires. Also, more aggressive repeater configurations and longer wire length yield more energy saving [5].

REFERENCES

- [1] M. T. Bohr, "Interconnect scaling—the real limiter to high performance ULSI," in *Proc. Int. Electron Devices Meeting*, Dec. 1994, pp. 241–244.
- [2] P. Saxena *et al.*, "Repeater scaling and its impact on CAD," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 451–463, Apr. 2004.
- [3] Z. Pan, "Interconnect synthesis and planning for high performance IC designs," Ph.D. dissertation, Univ. California, Los Angeles, 2000.
- [4] S. P. Khatri, "Cross-talk noise immune VLSI design using regular layout fabrics," Ph.D. dissertation, Univ. California, Berkeley, 1999.
- [5] C. J. Akl and M. A. Bayoumi, "Transition skew coding: A power and area efficient encoding technique for global on-chip interconnects," in *Proc. Asia South Pacific Design Automation Conf.*, 2007, pp. 696–701.

Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers

Sammy H. M. Kwok and Edmund Y. Lam

Abstract—This paper presents an effective field-programmable gate array (FPGA)-based hardware implementation of a parallel key searching system for the brute-force attack on RC4 encryption. The design employs several novel key scheduling techniques to minimize the total number of cycles for each key search and uses on-chip memories of the FPGA to maximize the number of key searching units per chip. Based on the design, a total of 176 RC4 key searching units can be implemented in a single Xilinx XC2VP20-5 FPGA chip, which currently costs only a few hundred U.S. dollars. Operating at a 47-MHz clock rate, the design can achieve a key searching speed of 1.07×10^7 keys per second. Breaking a 40-bit RC4 encryption only requires around 28.5 h.

Index Terms—Brute-force attack, field-programmable gate array (FPGA), RC4 encryption.

I. INTRODUCTION

RC4 has been the most popular and powerful stream cipher since its invention in 1987 [1]. To break an RC4 encryption requires either guessing the internal state of the cipher or brute-force search of the whole key space [2]. In an RC4 encryption using an 8-bit word, the total number of possible internal state is 256!

Tomasevic *et al.* [3], [14] proposed a cryptanalytic attack that employs the tree representation of an RC4 cipher to guess its internal state with complexity of 2^{731} . However, RC4 encryptions usually use rela-

tively short keys such as 40-bit keys, implying that the total number of possible keys is just 2^{40} . Thus, it is more effective to use brute-force key search to break the RC4 encryption instead of guessing the internal state of the cipher [2]. Such a brute-force RC4 key searching system can be implemented in hardware or software. In hardware, system resources such as system clock can be shared by all key searching units easily, and the communication among these units can be effective because they are collocated in a single chip. However, in software, all the key searching units are implemented in separate computers which are connected together through computer network. Compared to the communication links used in hardware such as interconnection links among CLBs in a field-programmable gate array (FPGA), a computer network is slow and less efficient, and it may become the performance bottleneck of the software key searching system.

Tews *et al.* [4] demonstrated a practical key recovery attack on WEP which is based on a partial key exposure vulnerability in the RC4 stream cipher used in the WEP protocol. They presented a WEP attack implemented in software which can recover a 104-bit WEP key in less than 60 s. However, this kind of technique cannot be applied to other RC4 encryptions which do not use the same flawed session key generation scheme as that in the WEP protocol.

Hamalainen *et al.* [5] and Kitsos *et al.* [6] proposed FPGA-based hardware implementations of RC4 ciphers. However, as large FPGA resource usage per encryption engine is required in their designs, both implementation schemes are inefficient for implementing a massively parallel RC4 key searching system. Meanwhile, Couture *et al.* [7] proposed a FPGA-based brute-force attack on RC4 encryption using both software and hardware. In the hardware approach, they use a very large FPGA chip to implement a parallel RC4 key searching engine, which consists of 500 key searching units in the FPGA chip. Such an engine can break a 40-bit RC4 encryption in 40 h. However, as their design was not optimized to fully utilize the FPGA resources and minimize the clock cycles per key search, it requires a large and expensive FPGA device and relatively long running time to break a 40-bit RC4 encryption.

Tsoi *et al.* [8], using a massively parallel implementation of an RC4 key searching engine in a single Xilinx XCV1000E chip, managed to break a 40-bit RC4 encryption in 50 h. This is one of the most optimal implementations for RC4 brute-force key search. However, their proposed architecture of implementing S-Box using dual-port RAM is not optimized enough that extra resources are wasted for solving the memory contention problem and one block RAM can only implement one S-Box.

In this paper, we present a highly optimized FPGA-based massively parallel RC4 key searching system, integrating 176 key searching units on a single Xilinx XC2VP20-5 chip. Our proposed implementation scheme achieves a significant speedup in the key searching process because of the following novel implementation techniques.

- All the key searching units are synchronized so that many of the system components can be shared among all the units, reducing the total resource usage.
- An innovative architecture for implementing the key scheduling process is used so that the total number of clock cycles for searching one key can be reduced to 772, compared to 792 in [8].
- The dual-port simultaneous read/write feature of the Block RAMs in the FPGA chip is fully exploited so that one Block RAM can be used to implement two key searching units, which outperforms all other proposed implementations in [5]–[8].

We implement a massively parallel RC4 key searching system that breaks a 40-bit RC4 encryption in 28.5 h. This is the fastest among other implementations in [7] and [8].

Manuscript received April 9, 2007; revised August 6, 2007 and September 29, 2007; accepted October 1, 2007.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: samkwo@graduate.hku.hk; samkwo@i-cable.com; elam@eee.hku.hk).

Digital Object Identifier 10.1109/TVLSI.2008.2000670

The rest of the paper is organized as follows. The RC4 encryption algorithm is described in Section II, followed by our RC4 key searching system architecture in Section III. Section IV describes the implementation details and results. We discuss and conclude our findings in Sections V and VI.

II. RC4 ENCRYPTION

A. Algorithm

RC4 is a binary additive stream cipher [2]. It uses a variable-sized key ranging from 8 to 2048 bits in multiples of 8 bits. It is actually a class of algorithms parameterized on the size of its block. The parameter, n , is the word size for the algorithm. In most applications, n is chosen to be 8. The internal state of RC4 consists of a table, S-box (known as S), of size 2^n words and two word-sized counters, i and j .

The core of RC4 encryption consists of two functions: key schedule algorithm (KSA), which is responsible for the initialization of a key-dependent permutation in S, and keystream generator which generates a sequence of bits that can be XOR'ed with the plaintext for encryption and with the ciphertext for decryption.

1) *Key Schedule Algorithm*: It consists of two phases: initialization phase in which S is set to the identity permutation, and scrambling phase in which it uses a key (K) with L bytes long to continuously swap values of S to produce new unknown key-dependent permutations. As the only action on S is value swapping, the fact that S contains a permutation is always maintained. Algorithm details are as follows.

Initialization phase:

$$j = 0, S[i] = i \forall i \in [0, 2^n - 1]$$

Scrambling phase:

$$j = j + S[j] + K[i \bmod L], \text{swap}(S[i], S[j]) \forall i \in [0, 2^n - 1]$$

2) *Keystream Generator*: It continuously shuffles the permutation stored in S and picks up a different value from the S permutation as output. One round of the cipher outputs an n -bit word as the keystream. Algorithm details are:

Initialization phase:

$$i = 0, j = 0$$

Keystream generation loop:

$$i = i + 1, j = j + S[i], \text{swap}(S[i], S[j])$$

$$t = S[i] + S[j], \text{output } S[t]$$

B. Characteristics

Based on the above key scheduling and keystream generation algorithms, RC4 has the following characteristics.

- S starts as an identity permutation of the 2^n possible words and remains a permutation throughout.
- Since S is always a permutation, the state just holds $\log_2(2^n!) + 2n \approx 1700$ bits of information [9].
- Knowing the internal state of the cipher at a given time is sufficient to predict all the keystream bits in the future and so to break the cipher.
- As the permutation of S depends solely on K, knowing the latter can break the cipher.
- The period of the output keystream depends only on K, which is normally very long and hard to predict.

- In many commercial applications, $n = 8$ and $L = 5$ (40 bits). Thus, to break an RC4 encryption, it is more effective to find K by brute-force attack on the entire key space instead of finding the internal state of the cipher.

III. PROPOSED TECHNIQUES

A. RC4 Key Searching System

Our proposed RC4 key searching system consists of multiple key searching engines which can be implemented in a single FPGA chip to carry out massively parallel key search of an RC4 encryption. Each key searching engine consists of multiple key searching units.

B. Operations for Testing One Key

To test a new key, key scheduling and keystream generation must be performed. For the former, 256 clock cycles are required to initialize the S-box, and then 256 iterations are required to scramble it. In each iteration, four clock cycles are required (reading $S[i]$, calculating j and reading $S[j]$, writing $S[i]$ and writing $S[j]$). For the latter, generating a byte of keystream requires five clock cycles (reading $S[i]$, calculating j and reading $S[j]$, writing $S[i]$, writing $S[j]$, and calculating t , reading $S[t]$ and then outputting the result). After getting the first byte from the key searching unit, the output data is checked against the first byte of the chosen ciphertext. If they do not match, the key being tested is incorrect and the next key testing should start. Otherwise, the above iteration is repeated until the first five bytes output from the keystream generator match with the corresponding bytes of the chosen ciphertext, in which the key of the RC4 encryption is found. If the key being tested is incorrect, the expected value of the number of iterations in the keystream generation process is approximately one. Thus, the total number of cycles required for testing one key is $256 + 256 * 4 + 5 * 1 = 1285$ cycles.

C. Reducing the Number of Clock Cycles for Testing One Key

1) *Using "Read-Before-Write" Access Mode*: As both Distributed RAM and Block RAM in modern FPGA chips support "read-before-write" access mode, we can use it to read $S[j]$ and write $S[i]$ in a single cycle, reducing the total number of clock cycles for testing one key to $256 + 256 * 3 + 4 * 1 = 1028$ cycles.

2) *Using 16-Bit Wide On-Chip Block RAM*: A S-Box can be implemented using a 256 block of 8-bit wide Distribution RAM or Block RAM. However, if 16-bit wide RAM is used instead, the S-box initialization can be embedded in the S-box scrambling process. The mechanism is to divide the 16-bit wide RAM into two halves (S_H and S_L) via the high-order byte and the low-order byte of the data bus. For testing a new key, only one half of the RAM is used for the key scheduling and keystream generation, the other half is initialized to the identity permutation for the next key testing. The detailed scheme is as follows.

- In key scheduling: scramble $S_H[i]$ and write $S_L[i]$ with i .
- In keystream generation: calculate and output $S_H[t]$ using $S_H[i]$ and $S_H[j]$ and write $S_L[i]$ with i .
- In testing the next key: swap the use of S_H and S_L .

As a result, 256 clock cycles used for the initialization of the S-box can be saved, reducing the total number of clock cycles for testing one key to: $256 * 3 + 4 * 1 = 772$ cycles.

D. One Key Searching Unit Tests Two Keys Simultaneously

The Block RAM can be used to implement a true dual-port RAM. If we use 512-byte Block RAM and divide it into two halves (S_A and S_B) via the most significant bit of the address, one key searching unit can use these two halves as two independent S-boxes for testing two keys simultaneously. For example, a single key searching unit can use S_A to test a key, K, and use S_B to test another key, $K + 1$, simultaneously.

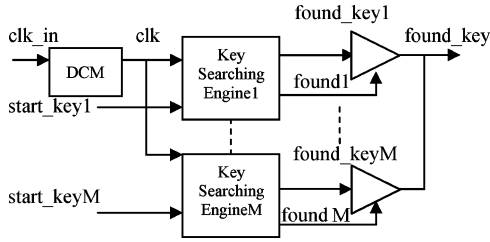


Fig. 1. Schematic of the proposed RC4 key searching system.

E. Synchronization Between Key Searching Units

We synchronize all the key testing operations within a key searching engine using a single system clock. Because of the synchronization of all the units, many system resources can be shared among the units, resulting in a high density of key searching units per FPGA chip.

F. Key Search

40-bit test keys in the whole 40-bit key space are sequentially fitted into the key searching units to search for the correct key. To generate the test keys, one key searching engine uses a 40-bit counter, K , shared among all the key searching units inside the engine. In each unit, an 8-bit register $K5$ is used to store the last byte of the test key. The total number of key searching units in a key searching engine, n , is then restricted to be of form of 2^m where $m \leq 8$. Thus, only one 40-bit counter and n 8-bit registers are required. In comparison, the straightforward key generation approach requires n 40-bit counters.

IV. IMPLEMENTATION DETAILS AND RESULTS

The proposed RC4 key searching systems using different techniques described earlier were synthesized and implemented in VHDL codes [10] using Xilinx ISE 8.1i software development tool. The FPGA platform used was a Xilinx HW-AFX-FF1152-300 FPGA Development Board which consists of a Virtex-II Pro XC2VP20-5 FPGA. A ChipScopePro 8.1.03i software tool was used for probing signals in the FPGA platform during testing.

A. Main Framework

The system consists of M key searching engines as shown in Fig. 1. Each key searching engine consists of n key searching units. By passing test keys ($start_keys$) in different key ranges to the key searching engines, searching the whole key space can be carried out in parallel.

B. Key Searching Unit

Signal flow in a key searching unit is shown in Fig. 2. Its core component is an S-box implemented by either Block RAM or Distributed RAM. Other components include address and data multiplexers for multiplexing addresses and data for the access of the S-box, counters for implementing i and K , adders for calculating j and t , and a comparator for testing the output from the keystream generator against the chosen ciphertext. Multiplexers can be implemented by either CLBs or tristate buffers (TBuFs) with similar gate delay performance. As TBuFs in the FPGA chip are insufficient for implementing all the address and

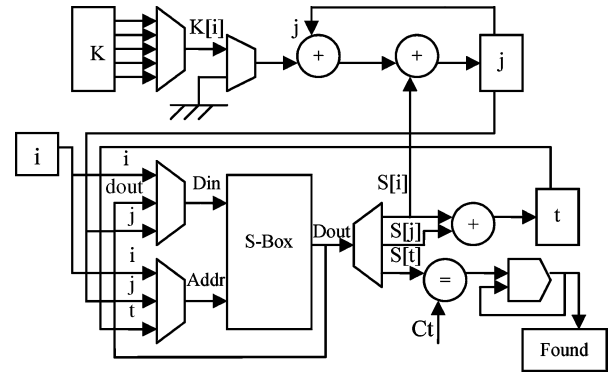


Fig. 2. Signal flow in a key searching unit.

data multiplexers, we implement some of them by TBuFs and others by CLBs.

C. FPGA Resources Usage

An XC2VP20-5 chip has 9280 slices, 4640 TBuFs, and 88 Block RAMs. Resources such as Block RAM, Distributed RAM, adders, multiplexers and comparators are required in each key searching unit while other resources such as counters, FSM are shared among all the units. The amount of slices for implementing a key searching engine is $a + bn$, where a denotes number of slices shared by all key searching units, b indicates number of slices used by a key searching unit, and n is the total number of key searching units.

D. Implementation Schemes and Their Results

We developed three different implementation schemes. The total amount of FPGA resources used and the maximum system clock frequency in the three schemes were compared to find out the most optimal and effective scheme for implementing an RC4 key searching system.

1) *Scheme 1*: Distributed RAM was used to implement the S-Boxes and the technique described in Section III-C-1 was used in the KSA process. However, the techniques described in Section III-C-2 and III-D cannot be used as Distributed RAM is not a true dual-port RAM. According to equation in Section IV-C, maximizing n can reduce the overhead in the resource usage. Because of the limited slice resources in the FPGA chip, the largest size for the 1st key searching engine is 64. Following the same rule, we implemented three more key searching engines of size of 8, 4, and 1 to fully utilize all the on-chip resources.

2) *Scheme 2*: Block RAM with 8-bit bus width and 512 data depth was used to implement the S-Boxes and the technique described in Section III-D was employed to test two keys simultaneously by one key searching unit. We implemented three parallel RC4 key searching engines with sizes of 64, 16, and 8. As there were slices and TBuFs left on the chip after implementing the three engines, we implemented three more key searching engines of sizes of 16, 8, and 4 using Distributed RAM.

3) *Scheme 3*: A Block RAM with 16-bit bus width and 512 data depth was used to implement two S-boxes for each key searching unit as described in Section III-D and the technique described in Section III-C2 was employed to reduce the total clock cycles for testing one key to 772. We implemented three key searching engines with size of 64, 16, and 8.

TABLE I
RC4 IMPLEMENTATIONS COMPARISON

Implementation scheme	Scheme 1	Scheme 2	Scheme 3
No. of key searching units	90	204	176
CLB Slices used	9107	9245	9125
TBufs used	3600	3808	2816
Block RAM used	0	88	88
Frequency (MHz)	63	50	47
Throughput (MKey/s)	5.5	9.9	10.7
Time required for a 40-bit key search (hour)	55	30.8	28.5

Performance comparison among the three proposed implementation schemes is summarized in Table I.

V. DISCUSSION

A. Using Distributed RAM and Block RAM

We show from the implementation that S-boxes used in the key searching units can be built by using Distributed RAM or Block RAM. As both support “read-then-write” access mode [11], the technique described in Section III-C-1 can be used to reduce the number of clock cycles per iteration in the key scheduling process to 3 compared to 4 in [7] which did not use this “read-then-write” access mode, although the FPGA chip used in their implementation also supports it. Meanwhile, [8] employed the dual-port writing feature to write $S[i]$ and $S[j]$ simultaneously in a single clock cycle to achieve throughput of three cycles per key scheduling iteration. However, under this implementation scheme, a possible memory contention problem may exist in the last clock cycle of each iteration in the key scheduling and key generation. Thus, extra resources such as comparators are required to solve the problem.

Comparing the performance of Distributed RAM and Block RAM, the former is slightly faster [11]. However, Distributed RAM requires resource of slices to implement but Block RAM does not, because it is a standard primitive in most Xilinx FPGA chips.

In an X2VP20-5 chip, one Block RAM consists of 18-kb memory [12]. As one S-Box requires only $256 * 8 = 2$ kb memory, one Block RAM can be used to implement two S-Boxes for testing two keys simultaneously. In [7] and [8], the FPGA chips used consist of Block RAM with 4 kb or above. However, because of the limitation in their implementation architecture, one Block RAM can only be used for implementing one S-Box, resulting in smaller density of key searching units per chip compared to our architecture. Meanwhile, Distributed RAM requires slices [12] (one Distributed RAM of size of $256 * 8$ bits requires 16 slices), thus more slice resources are required if one Distributed RAM is used to implement two S-Boxes. Hence, it is not recommended to use one Distributed RAM to implement two S-Boxes at a time.

Choice of using Distributed RAM or Block RAM to implement S-Boxes depends on the resource of slices and Block RAM in the FPGA device. We should first exploit all the Block RAM resources. If there are slice resources left, we can then use Distributed RAM to implement other S-boxes.

B. Number of Clock Cycles in Key Scheduling Process

By using the technique described in Section III-C1, the total number of clock cycles in the key scheduling process can be reduced from 1285 to 1028. In addition, we proved that by applying the technique described in Section III-C2, we can embed the S-Box initialization process into the S-Box scrambling process and so the total number of clock cycles in the key scheduling process can be further reduced to

772, which is the smallest compared to other proposed implementations in [5]–[8].

C. Implementation of Multiplexers

Similar to [8], we use many two-bit and three-bit multiplexers to control the access of the S-Box. In Xilinx FPGA, a multiplexer can be built by slices or TBufs. By implementing some of the multiplexers by slices and others by TBufs, the resources of slices and TBufs in a FPGA chip can be fully utilized to maximize the total number of key searching units in the chip.

D. System Performance Considerations

The time required to crack an RC4 encryption with brute-force depends on the total number of parallel key-searching units in the FPGA chip, the maximum system clock frequency and the total number of clock cycles for testing one key.

1) *Total Number of Parallel Key Search*: Resource usage per key searching unit determines the maximum number possible in a FPGA chip: the higher the resource usage, the less the number of units.

In Scheme 1, as it uses Distributed RAM which consumes slice resource to implement S-Boxes, resource usage per RC4 key searching unit is higher than that in Scheme 2 which uses Block RAM to implement S-Boxes. In addition, in Scheme 1, only one key searching unit can test a key at a time, whereas in Schemes 2 or 3, one key searching unit can test two keys simultaneously. Thus, the total number of key searching units using Schemes 2 and 3 is much larger than that using Scheme 1 and those in [7] and [8].

In Scheme 3, more resources are required for embedding the S-Box initialization process into the S-Box scrambling process. Thus, the total number of key searching units is less than that in Scheme 2.

2) *Maximum Clock Frequency*: Complexity of each key searching unit determines the maximum clock frequency: the more complex the unit, the slower the maximum clock frequency can be used. Among the three schemes, Scheme 1 is simplest and Scheme 3 is the most complicated. Thus, the maximum clock frequency is the highest in Scheme 1 and the lowest in Scheme 3.

3) *Clock Cycles for Testing One Key*: For Schemes 1 and 2, the total number of clock cycles for testing one key is 1028. For Scheme 3, the total number can be reduced to 772.

VI. CONCLUSION

In this paper, a highly parallelized RC4 key searching system based on a FPGA device is presented. Three different implementation schemes were proposed and tested. Their implementation results indicate that Scheme 3 can be used for an effective hardware implementation of a parallel key searching system in a Xilinx XC2VP20-5 chip. Such system can search more than 10^7 keys per second, which is the fastest among all single-FPGA-chip RC4 key searching systems proposed previously in the literature. Besides, the slice resource per key searching unit is 51 slices, which is also the smallest among other previously proposed implementations in [7], [8] and [13]. Using this implementation, a complete 40-bit RC4 key space can be tested in 28.5 h. The design is modularized and can be easily adapted to other hardware platforms containing multiple FPGA chips. When bigger or faster FPGA devices are used, the time for cracking an RC4 encryption using brute-force can be further reduced. For example, if an XC2VP50-5 chip is used, we simulated that an RC4 key searching system consisting of 458 key searching units can be implemented in the chip. Such system can complete a 40-bit RC4 key space search in 11 h.

Similar techniques can be applied to hardware implementations of other ciphers or cryptographic algorithms with a view to carrying out

brute-force attack on the ciphers or constructing high-speed processing engines for the algorithms.

REFERENCES

- [1] B. Schneier, *Applied Cryptography*. New York: Wiley, 1994.
- [2] R. Wash, Lecture Notes on Stream Ciphers and RC4 [Online]. Available: <http://www.crimelabs.net/docs/stream.pdf> unpublished
- [3] J. Tomasevic, S. Bojanic, and O. Nieto-Taladriz, "Finding an internal state of RC4 stream cipher," *Inf. Sci.*, vol. 177, no. 7, Apr. 1, 2007.
- [4] E. Tews, R.-P. Weinmann, and A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds," *IARC's Cryptology ePrint Archive* Apr. 2007 [Online]. Available: <http://eprint.iacr.org/2007/120>
- [5] P. Hamalainen, M. Hannikainen, T. Hamalainen, and J. Saarinen, "Hardware implementation of the improved WEP and RC4 encryption algorithms for wireless terminals," in *Proc. Eur. Signal Processing Conf.*, Sep. 2000, pp. 2289–2292.
- [6] P. Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware implementation of the RC4 stream cipher," in *Proc. IEEE Midwest Symp. Circuits and Syst.*, Dec. 2003, pp. 1363–1366.
- [7] N. Couture and K. Kent, "The effectiveness of brute force attacks on RC4," in *Proc. 2nd Annu. Conf. Commun. Networks and Services Res. (CNSR'04)*, 2004, pp. 333–336.
- [8] K. Tsoi, K. Lee, and P. Leong, "A massively parallel RC4 key search engine," in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach.*, 2002, pp. 13–21.
- [9] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Proc. 8th Annu. Workshop on Selected Areas in Cryptography*, Aug. 2001, pp. 1–24.
- [10] D. L. Perry, *VHDL Programming by Example*, 4th ed. New York: McGraw-Hill, 2002.
- [11] "Virtex-II Pro and Virtex-II Pro X FPGA User Guide" ver. 4.0, March 23, 2005 [Online]. Available: <http://direct.xilinx.com, UG012>
- [12] "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet" ver. 4.3, Jun. 20, 2005 [Online]. Available: <http://direct.xilinx.com, DS083>
- [13] M. Galanis *et al.*, "Comparison of the hardware architectures and FPGA implementations of stream ciphers," in *Proc. IEEE Int. Conf. Electron., Circuits Syst.*, 2004, pp. 571–574.
- [14] J. Tomasevic, S. Bojanic, and O. Nieto-Taladriz, "Finding an internal state of RC4 stream cipher," *Fuzzy Set Appl. Industrial Eng.*, pp. 1715–1727.