Article

Design and Implementation of a UART Module on FPGA Using RTL for Cryptographic Encryption and Decryption Techniques

Christian-Antonio Colin-Cejudo[1,†,‡], **Gonzalo-Issac Duchén-Sánchez** [2,‡] **and Gabriel Sánchez-Pérez** [3,†]

[1] christian.upiita@gmail.com
[2] gduchen@ieee.org
[3] caaann@gmail.com

Abstract: The increasing demand for robust and efficient information security has led to the growing adoption of specialized hardware for cryptographic operations. In response to the rise in cyber threats and the need to process large volumes of data in real time, hardware-based cryptographic solutions offer significant advantages in terms of performance, resistance to attacks, and secure storage of cryptographic keys. This paper presents the implementation of a secure communication system using the UART (Universal Asynchronous Receiver-Transmitter) protocol as the foundation for a Register Transfer Level (RTL) design on an FPGA platform. The base protocol was modified to introduce an additional hardware-level security layer. Furthermore, cryptographic techniques—specifically encryption and decryption—were integrated into the design to enhance data protection and integrity during transmission. The results demonstrate the feasibility of embedding cryptographic mechanisms directly into communication hardware, providing a scalable and efficient solution for secure embedded systems.

Keywords: Cryptographic hardware; Information security; Encryption, Decryption; Cryptographic keys; Digital signatures; Authentication; Cyber threats; Data processing; Communication protocols; UART (Universal Asynchronous Receiver-Transmitter); FPGA (Field-Programmable Gate Array); RTL design (Register Transfer Level); Hardware security layer; Cryptographic techniques; Critical infrastructure.

1. Introduction

Currently, information security has become a fundamental pillar for the development of reliable digital systems. The increasing sophistication of cyber threats, combined with the exponential growth in data generation and transmission, demands increasingly robustand efficient solutions. Cryptographic hardware offers significant advantages, such as higher performance,lower latency, reduced energy consumption, and greater resistance to both physical and logical attacks.Field Programmable Gate Arrays (FPGAs) have become one of the most versatile technologies for electronic system design. These devices provide a cost-effective solution,especially for low-volume production, since the initial cost of prototyping is considerably lower compared to Application-Specific Integrated Circuits (ASICs). Moreover, a key advantage is their reconfigurability during operation, which allows a single device to perform multiple predefined functions, thereby optimizing space and reducing costs.Modern integrated circuit technology makes it possible to integrate complex systems into highly compact dimensions, referred to as embedded systems or Systems on Chip (SoC). These systems, designed to fulfill specific functions, combine hardware and software tailored for each task. In this context, FPGAs, thanks to their reconfigurability, are valuable tools for developing prototypes or small-scale series at

Received:
Revised:
Accepted:
Published:

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

affordable costs.Among the most widely used communication protocols in embedded systems is the Universal Asynchronous Receiver-Transmitter (UART), due to its simplicity, low implementation cost, and broad compatibility. However, this protocol lacks native security mechanisms, making it vulnerable to interception, manipulation, and unauthorized access. This work proposes the design and implementation of a cryptographic security layer on top of the UART protocol, using a Register Transfer Level (RTL) approach on an FPGA platform. The implementation leverages a Hardware Description Language (HDL) to model and design digital circuits. The modification of the protocol not only ensures secure data transmission but also integrates encryption and decryption techniques directly into hardware. This solution aims to demonstrate the feasibility of incorporating efficient cryptographic security into embedded communication systems without compromising performance or scalability.

2. Materials and Methods

This chapter reviews the foundations of classical cryptography, emphasizing the Caesar cipher as an illustrative example of monoalphabetic substitution techniques. While obsolete for modern applications due to its vulnerability to brute-force and frequency analysis attacks, the Caesar cipher remains valuable for educational purposes and as a test case for implementing sequential processing, finite state machines, and resource evaluation in FPGA-based platforms. Its simplicity makes it an effective starting point for understanding the relationship between mathematical algorithms and their hardware representation.

2.1. Operation of the Algorithm

From a mathematical perspective, the Caesar cipher can be represented by the following function: $C(x) = (x + k) mod n$

Where:

- $C(x)$ is the encrypted character.
- x is the index of the character in the alphabet $A = 0, B = 1, \ldots, Z = 25$.
- k is the shift key.
- n is the total number of characters in the alphabet, usually $n = 26$ for the Latin alphabet.

Decryption consists of applying the inverse operation:

$$P(x) = (x - k) \bmod n$$

For example, if the word ***"HOLA"*** is encrypted with a shift of $k = 3$, the result is ***"KROD"***. To decrypt, the inverse shift is applied.

The discussion then traced the evolution of digital design methodologies, from early breadboard prototyping to the integration of computer-aided design (CAD) tools, which enabled accurate simulation and validation of electronic systems. The emergence of hardware description languages (HDL) such as VHDL and Verilog marked a turning point by providing standardized, structured methods for system modeling and verification. The subsequent development of SystemVerilog extended these capabilities, incorporating object-oriented programming, functional verification, and reusable testbench environments, thus consolidating the design and verification process within a unified framework.

A comparative analysis with software engineering highlighted both syntactic similarities and conceptual differences, underlining the importance of adopting a hardware-oriented mindset when aiming for physical synthesis and implementation on FPGA or ASIC platforms. This distinction is essential to ensure that HDL-based models accurately reflect hardware behavior rather than abstract algorithmic processes.

Finally, the Altera DE2-115 development board was presented as a versatile and widely adopted prototyping platform. Equipped with a Cyclone IV FPGA, embedded and external memories, extensive I/O interfaces, and compatibility with tools such as Quartus Prime, this board provides the necessary infrastructure for developing, simulating, and validating digital systems. Its relevance in academic and research contexts makes it particularly suitable for projects focused on integrating cryptographic algorithms into reconfigurable hardware architectures.

3. Results

In order to validate the proposed obfuscation mechanism as an enhancement to the traditional Caesar cipher, a series of experiments were conducted both in software and hardware domains. The primary objective was to demonstrate that the integration of an additional obfuscation layer not only increases the complexity of deciphering the encrypted message but also preserves functional equivalence between the software prototype and its hardware implementation.

The first stage of the experimentation involved the development of a Python script that implemented the Caesar cipher combined with the proposed obfuscation technique. This environment allowed rapid prototyping and verification of functional correctness. Various input datasets, including alphanumeric strings and binary sequences, were subjected to encryption and decryption tests. Statistical measures were then applied to evaluate the distribution of ciphertext characters Fig 1, aiming to confirm that the obfuscation achieved a reduction in predictable frequency patterns typically exploitable in classical substitution ciphers. The resulting charts demonstrated a noticeable flattening of character frequency distributions, indicating that the modified cipher effectively mitigates one of the main weaknesses of the traditional Caesar scheme.

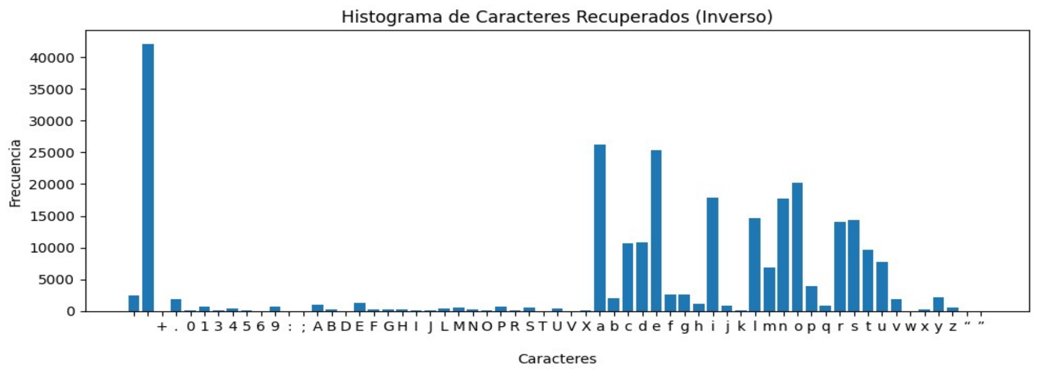

Figure 1. Plain Spanish text distribution.

Following successful validation in the software domain, the algorithm was translated into a hardware description suitable for FPGA implementation. Specifically, an RTL (Register Transfer Level) block was designed using VHDL to replicate the operations of both the Caesar cipher and the obfuscation layer. The choice of RTL ensured that the design remained synthesizable and could be physically deployed on the Altera DE2-115 FPGA platform. This translation process required special attention to timing, resource utilization, and concurrency management, as hardware execution introduces constraints not present in sequential software execution.

Simulation of the RTL block was performed prior to synthesis, ensuring that the output matched the reference Python implementation across all test vectors. Waveform analysis confirmed correct alignment of data paths, clock synchronization, and control logic behavior. Subsequently, the design was synthesized, placed, and routed onto the FPGA. Resource utilization reports indicated that the additional obfuscation logic introduced only a modest increase in logic element consumption, remaining well within the capacity of the Cyclone IV device.

To further validate the effectiveness of the hardware design, experimental results were collected directly from the FPGA implementation. Input text sequences were encrypted, processed through the obfuscation mechanism, and then decrypted in real time. The resulting ciphertexts were again subjected to statistical frequency analysis Fig 2. Comparison between the software-generated and hardware-generated distributions confirmed functional equivalence: both platforms achieved similar levels of character dispersion and unpredictability.

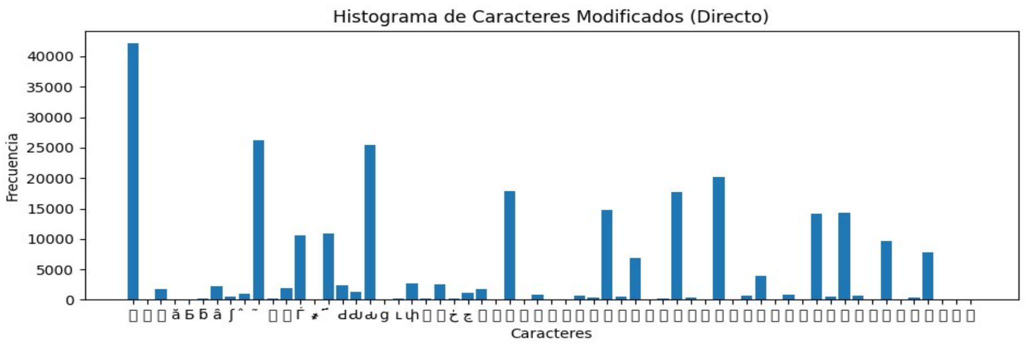

Figure 2. Exit text hardware-generated obfuscation distribution.

3.1. Conclusions and Future Work

In summary, the experimental results provide strong evidence that the proposed obfuscation technique enhances the Caesar cipher without compromising its functional reliability. The consistency observed between the Python prototype and the FPGA-based RTL implementation demonstrates that the approach is both feasible and scalable across computational platforms. Furthermore, the low hardware resource overhead highlights the practicality of deploying the system in constrained environments where lightweight yet effective cryptographic mechanisms are required.

References

1. EEE Standards Association. (1995). IEEE Standard Verilog Hardware Description Language (IEEE Std 1364-1995). Institute of Electrical and Electronics Engineers.
2. alnitkar, S. (2003). Verilog HDL: A Guide to Digital Design and Synthesis (2nd ed.). Prentice Hall.
3. ahn, D. (1996). The codebreakers: The comprehensive history of secret communication from ancient times to the Internet (2nd ed.). Scribner.
4. chneier, B. (2015). Applied cryptography: Protocols, algorithms, and source code in C (20th anniversary ed.). Wiley.
5. ingh, S. (2000). The code book: The science of secrecy from ancient Egypt to quantum cryptography. Anchor Books.
6. tallings, W. (2017). Cryptography and network security: Principles and practice (7th ed.). Pearson.
7. EEE Standards Association. (2005). IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language (IEEE Std 1800-2005). IEEE. https://doi.org/10.1109/IEEESTD.2005.93210
8. EEE Standards Association. (2008). IEEE Standard Verilog Hardware Description Language. IEEE.
9. adrón, L. (1997). Implementación de modelos de circuitos neuronales electrónicos. Laboratorio de Computación Adaptable.
10. erasic Technologies. (s.f.). DE2-115 User Manual. Recuperado de https://www.terasic.com.tw
11. ntel Corporation. (2020). Cyclone IV Device Handbook (Volume 1 & 2). Retrieved from https://www.intel.com
12. ahid, F., & Givargis, T. (2010). Embedded System Design: A Unified Hardware/Software Introduction (2nd ed.). Wiley.