Design and Development of UART Communication Over RSA Encryption

Dr. Vasudeva G[1]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
devan.vasu921@gmail.com

Dr. Mallikarjun P Y[2]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
mallikarjunpy@gmail.com

Prof.Mahadev S[3]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
mahadevkhanderao@gmail.com

Dr. Tripti R kulkarni[4]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
triptikulkarni-ece@dsatm.edu.in

Dr.Bharathi Gururaj[5],Departmentof ECE,KSIT,Bengaluru,India,
bharathigururaj@gmail.com

Hemanth Kumar M.M[6]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
hemanthmm08@gmail.com

Abhishek H.J[7]
Department of Electronics and Communication Engineering
Dayananda Sagar Academy of Technology & Management
Bengaluru, India
abhishekhj12@gmail.com

***Abstract*—UART (Universal Asynchronous Receiver-Transmitter) is a widely adopted serial communication protocol in embedded systems, but its lack of inherent encryption exposes data to security vulnerabilities. This paper presents a hardware-level solution to secure UART communication by implementing the RSA encryption algorithm directly on an FPGA. The proposed architecture incorporates RSA key generation, encryption, and decryption along with UART interfacing in Verilog HDL, synthesized using the Xilinx ISE suite. Synthesis results on a Spartan-3 FPGA demonstrate the design's area-efficiency and suitability for real-time secure communication. The encryption and decryption operations are validated with UART testbenches, and the successful round-trip of plaintext data confirms the system's functional correctness and feasibility for IoT, automation, and secure embedded systems.**

Keywords-Embedded Systems, RSA Encryption, Secure Data Transmission, UART Communication .

I.INTRODUCTION

Secure communication is the main requirement of embedded systems[1], IoT networks, and industrial automation, where confidentiality and integrity of the data are of crucial concern[2]. The Universal Asynchronous Receiver-Transmitter (UART) is an extremely popular serial communication protocol[3] since it is easy and efficient. UART transmits the data in plain text, and hence it is extremely vulnerable to interception, eavesdropping, and tampering[4].

To counter these security issues, cryptographic techniques such as RSA (Rivest-Shamir-Adleman) encryption can be used over UART communication[5]. RSA is a public-key cryptosystem that facilitates secure data exchange by encrypting the message with a public key so that the recipient possessing the private key pair[6]for the public key is the only one able to decrypt the message successfully. This renders the data communication immune to man-in-the-middle attacks and malicious access and offers an alternative secure transmission channel over UART[7].

Richard A. Mollin, in "RSA and Public Key Cryptography", outlines the mathematical underpinnings and security aspects of RSA encryption and its applicability in encrypting computer communications. Suetonius, in The Lives of the Twelve Caesars [8], has written about Julius Caesar: ".if secrecy was necessary, he wrote in cyphers; that is, he used the alphabet in such a way, that no single word was legible. The method of deciphering those epistles was to replace the fourth letter for the first, as d for a, and so for the other letters correspondingly[9]." What is being described here is a method of sending messages in disguised form an example of enciphering or encrypting, which is the method of disguising messages[10]. The un-disguised (original) message is referred to as the plaintext or (less commonly) cleartext, and the disguised message the ciphertext, whereas the final message, packaged and dispatched, is referred to as a cryptogram[11]. The process of reversing the translation of ciphertext back into plaintext is referred to as decryption or deciphering, or (less commonly used) exploitation[12]. The science of cryptography is the science of sending messages in secret, that is, in enciphered form, so that only the in-tended recipient is able to decipher and read it[13]. The science of mathematical methods for attempting to outwit cryptographic methods is referred to as cryptanalysis[14].

This paper is a comprehensive explanation of the application of RSA encryption in UART communication, ranging from key generation to encryption and decryption, and real-world application in secure data transmission.It differs from the existing work [26], [27] based on RSA Algorithm used.In the existing work they have used AES Encryption method. RSA encryption in UART communication ensures confidentiality, integrity, and authenticity, and thus a good solution for present security concerns in embedded systems.In this work we design UART Communication using RSA Algorithm for encryption and decryption.

II. RSA ALGORITHM

The RSA (Rivest-Shamir-Adleman) algorithm is a commonly utilized public-key cryptography system that provides secure data transport[15]. It relies on the

mathematical properties of prime numbers and modular arithmetic, making it possible to encrypt and decrypt using a pair of keys: a public key and a private key[16].

A. *Key Generation*

RSA key generation involves the following steps:

i. Select two large prime numbers, p and q

ii. Compute the modulus, n as

$$n = p \times q$$

the modulus n is used for both the public and private keys.

iii. Calculate Euler's totient function **φ(n)**

$$\Phi(n) = (p-1) \times (q-1)$$

iv. Choose a public exponent, e such that

$$1 < e < \varphi(n), \text{ and } \gcd(e, \varphi(n)) = 1$$

A common choice to e is 65537 due to its balance between security and efficiency.

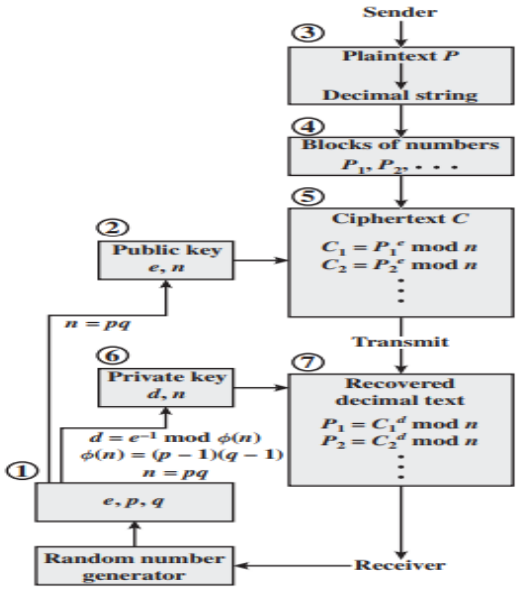

Figure 1: RSA process[18]

v. Compute the private exponent, d which satisfies

$$d = e^{-1} \bmod (\varphi(n))$$

the public key consists of (n, e) and the private key consists of (n, d)[17].

B. *Encryption Process*

To encrypt a plaintext message:

i. Convert the message into an integer M such that:

$$0 < M < n$$

ii. Calculate the ciphertext C based on the public key (n, e):

$$C = M^e \bmod n \qquad \text{eq(1)}$$

iii. Transmit the encrypted message C.

C. *Process of Decryption*

To decrypt the ciphertext C, the receiver applies the private key (n, d) to compute:

$$M = C^d \bmod n \qquad \text{eq(2)}$$

The decrypted result M is converted back to its original plaintext form.

D. *Security Implications*

The security of RSA is based on the difficulty of factoring large numbers. Since an attacker must derive p and q from n to compute d, RSA remains secure as long as sufficiently large prime numbers are used.

III. UART COMMUNICATION

Communication Protocol for Serial Data Exchange is popularly known as UART, which stands for Universal Asynchronous Receiver-Transmitter[19]. UART refers to communication protocols, which are widely used for purposes of exchanging serial data between devices. It is one of the major purposes of this communication standard, which includes data receiving and its transmission by converting a parallel to the serial format for transmission[20]. It will convert it back to a parallel format once received. Framing by itself is to ensure communication in UART, which includes the start and stop bits that indicate the beginning and end of a data packet entry, possibly adding a parity bit for checking errors[21]. The other main purpose is to control the baud rate, which specifies the speed of transmission in bits per second (bps). It must be equal for both transmitting and receiving devices to ensure synchronization[22]. Moreover, UART has the error detections and mechanisms to handle such errors through parity checking, framing error detection, and buffer overflow alerts. UART uses equipment flow control such as RTS/CTS and software flow control through XON/XOFF to manage data flow efficiently by preventing buffer overflows[23]. Besides that, this is also a full-duplex communication, where TX and RX are used for simultaneous transmission and reception, respectively, over separate lines. automation.

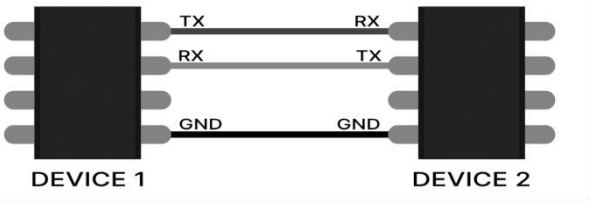

Figure 2 : UART Communication Protocol[16]

Interrupt generation is another feature that is important, where UART generates interrupts when data is received or a transmission is completed or an error occurs to lessen the load on the CPU and improve the efficiency of the environment. Also, these characteristics combine together to make it consume minimum power; therefore, with low energy, it is suitable for end devices where low power is used. All the above makes it a strong and reliable protocol for serial communication in diverse applications, including embedded systems, microcontrollers, and industrial

It is primarily the UART-the Universal Asynchronous Receiver Transmitter-these days responsible for communicating securely encrypted data in an RSA approach, implementation-based communication[24]. Under RSA encryption, one converts the plaintext message into cipher text through truly private encrypted sessions that connect this over a serial channel via UART. The major components of such a system would include the RSA encryption module, the UART

transmitter (uart tx), as well as the UART receiver (uart_rx). The UART transmitter will serialize the encrypted data by adding a start bit, data bits, and a stop bit before sending this through the communication channel. The UART receiver will detect the start bit and will read the incoming data sequentially until a stop bit is also verified, thereby ensuring integrity of data before passing it on for decrypting further. The testbench validates this process by ensuring that decryption matches the original plaintext post transmission. It integrates reliable and secure point-to-point communication of hardware[25]. Thus, UART can be said to provide excellent and lightweight implementation, as well as synchronization for serial transfer of data for embedded systems. In fact, UART's unique start and stop bits assist in a basic level error detection for better exchange reliability. The primary advantage of this integration of RSA with UART lies in its efficiencies while catering for security features for use in application systems, including but not limited to embedded systems, authentication protocols, and encrypted IoT data transfer.

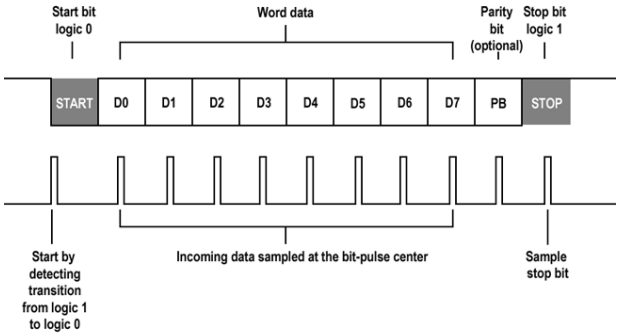

Figure 3:UART Data Communication[14]

IV. RSA Algorithm Over UART Communication

In implementing RSA encryption on UART communication, some of the most important considerations are made to assure that data is transmitted securely and reliably. The system starts with the process of key generation, where two large prime numbers are chosen to calculate the public and private keys. The encryption process uses modular exponentiation to convert plaintext into ciphertext for transmission. The UART module is responsible for converting the encrypted message to a serial data stream, which is then sent with the proper start and stop bits so as to maintain synchronization.

`

On the receive side, the UART receiver module reassembles the transmitted data and checks its integrity before passing it on to the RSA decryption module. Decryption is done with the private key using modular exponentiation to obtain the original message.

The hardware implementation of RSA over UART is very much applicable in embedded systems, IoT security, and secure remote authentication applications where there is a need to transfer sensitive information through a serial interface without compromise.

V. Results and Implementtion

The proposed technique is implemented on a Xilinx SPARTAN 3 device as a proof of concept. In this section, explain the implementation flow for this device. The schematic representation illustrates the Verilog-based hardware implementation of the RSA algorithm integrated with a UART (Universal Asynchronous Receiver-Transmitter) interface. The design consists of key functional blocks, including key generation, encryption, decryption, and UART communication modules. The RSA encryption and decryption operations utilize modular exponentiation and modular arithmetic, efficiently implemented using hardware-friendly algorithms to optimize performance.

The UART module enables seamless serial communication, allowing encrypted and decrypted messages to be transmitted between the hardware and an external system, such as a computer or microcontroller. The control logic ensures proper data flow, synchronization, and error handling, adhering to standard UART protocols.

This schematic provides a comprehensive overview of the data path, control signals, and interaction between modules, facilitating efficient FPGA or ASIC implementation of RSA over UART for secure communication applications.

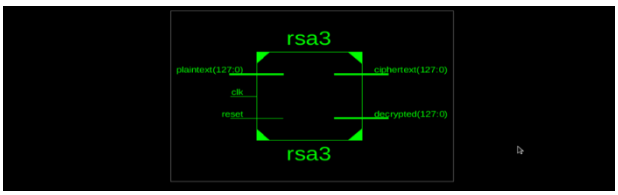

Figure 4: RTL Block Diagram of RSA over UART System

The proposed design was implemented in Verilog HDL using modular and hierarchical design principles. The hardware blocks include:

- RSA Key Generator
- RSA Encryption Module (Modular Exponentiation)
- RSA Decryption Module (Modular Inverse)
- UART Transmitter and Receiver (TX and RX)
- Top-Level Integration and Control Logic

Figure 4 illustrates the **block diagram of the RTL architecture**, showing the interaction between RSA and UART modules.

The synthesis result obtained are below,
RTL Top Level Output File Name: rsa3.ngr
Top Level Output File Name: rsa3
Output Format: NGC
Optimization Goal: Speed
Keep Hierarchy: No
Design Statistics
IOs: 386
Cell Usage:
BEL's (Basic Elements of Logic): 2
GND: 1
VCC: 1
Flip Flops/Latches: 128
LD (Latch with Data input): 128
Clock Buffers: 1

BUFGP (Global Clock Buffer): 1
IO Buffers: 288
IBUF (Input Buffer): 32
OBUF (output Buffer): 256

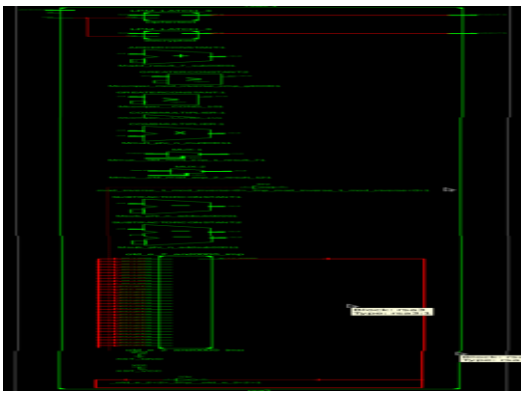

Figure 5: RTL Schematic

The above following RTL schematic involves the following component.

Table 1: Components of RTL Schematic

Sl.No	Components	Quantity
1	128 X 128-bit multiplier	1
2	128-bit subtractor	4
3	32-bit adder	2
4	128-bit latch	2
5	32-bit latch	1
6	32-bit comparator (greater than equal)	1
7	32-bit comparator (greater)	2
8	32-bit 4-to-1 multiplexer	2

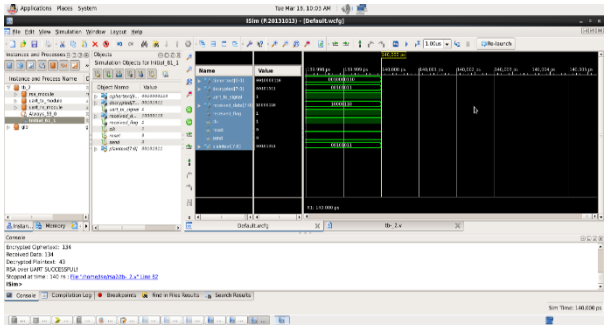

Figure 6 :Generated Output Waveform

In the above figure the waveform is generated for the input data, plaintext = 43, p= 17 and q= 23. The output obtained is ciphertext(c)= 134, decrypted value = 43, and the received data= 134. The public key obtained is (e, n) = 3 , 391 and the private key is (d, n) = 235, 391.

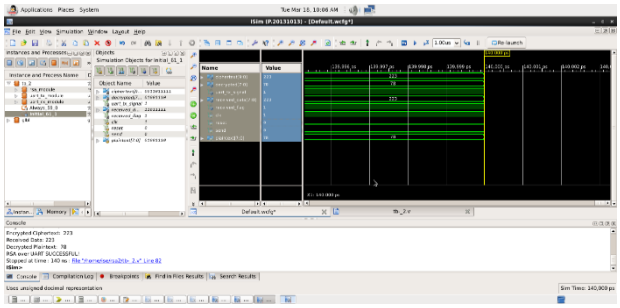

Figure 7 :Generated Output Waveform

In the above figure the waveform is generated for the input data, plaintext = 78, p= 23 and q= 41. The output obtained is ciphertext(c)= 223, decrypted value = 78, and the received data= 134. The public key obtained is (e, n) = 3 , 943 and the private key is (d, n) = 587, 943.

In the above both the example the plaintext is equal to decrypted value and ciphertext is equal to received data which satisfies the RSA algorithm. Hence the result is obtained as RSA over UART is successful.

VI. Synthesis and Resource Utilization

The Verilog code was synthesized using Xilinx ISE 9.1i targeting the Spartan-3 XC3S400-TQ144 FPGA. The synthesized design was optimized for speed with hierarchical flattening and FSM encoding applied automatically.

The generated RTL schematic in Figure 8 highlights the major hardware components inferred by the synthesis tool.

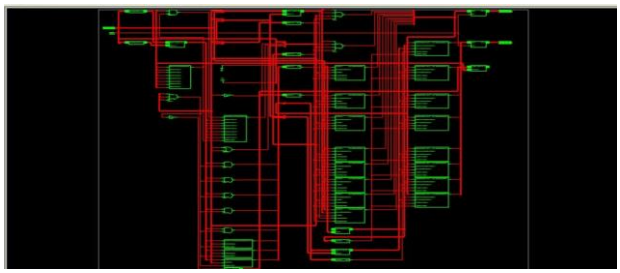

Figure 8: RTL Schematic Post-Synthesis

Table.2-Device Utilization Summary

Resource	Used	Available	Utilization
Logic Slices	10	3584	0%
4-input LUTs	17	7168	0%
Flip-Flops/Latches	8	—	—

Bonded IOBs	27	97	27%
Global-Clock Buffers	1	8	12%

The component breakdown includes:

- 1 × 2×32-bit ROM
- 1 × 9×9-bit Multiplier
- 5 Adders/Subtractors
- 6 Comparators
- 1 × 4-to-1 Multiplexer
- 2 Latches (10-bit and 8-bit)

Table.3- Timing Analysis

Parameter	**Value**
Min Input Arrival Time	6.491 ns
Max Output Delay	7.078 ns
Clock Used	BUFGP

These results confirm that the design can support UART operation at typical baud rates (e.g., 9600–115200 bps), with reliable performance and minimal logic utilization.

VII. Hardware Implementation on FPGA Kit

To demonstrate practical usability, the system was implemented on two development boards labeled:

- Board 1 – Plaintext Transmission (Input Module)
- Board 2 – Ciphertext Reception and Decryption (Output Module)

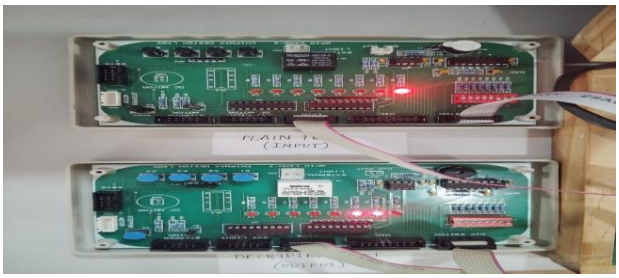

Figure 9: FPGA Kit Implementation of RSA Over UART

Top board – Plaintext Transmission (Input) Bottom board -Decryption (output).

The UART connection was established between the boards using standard ribbon cables. LED indicators on both boards reflect data transmission status. As seen in Figure 9, LED patterns confirm that the ciphertext generated on Board 1 is successfully transmitted over UART and decrypted accurately on Board 2.

The upper board labeled “DECRYPTED TEXT” shows multiple LEDs active, indicating successful reception and processing of encrypted input, whereas the lower board labeled “PLAIN TEXT” has a single LED active, confirming initial plaintext input. The consistent behavior during multiple trials confirms that

Figure 10: FPGA Kit Implementation of RSA Over UART

- UART transmission is stable and synchronized.
- RSA encryption and decryption logic works correctly on FPGA.
- The decrypted value matches the input plaintext, validating round-trip integrity.

I/O and GPIO Configuration

The FPGA implementation of the RSA over UART system makes effective use of General Purpose Input/Output (GPIO) pins to manage both serial communication and status indication. The Spartan-3 XC3S400-TQ144 device offers 97 user I/O pins, out of which 27 GPIO pins were utilized in this project, as confirmed by the synthesis report.

The GPIO usage is categorized as follows:

- **UART Communication**:
  - TX (Transmit): 1 GPIO pin
  - RX (Receive): 1 GPIO pin
- **Plaintext Input and Control**:
  - 8 GPIO pins for 8-bit plaintext input
  - 1–2 pins for control signals (start/reset)
- **Ciphertext Output and Decryption Display**:
  - 8 GPIO pins for decrypted output or LED status
- **LED Indicators**:
  - 8 GPIOs are mapped to onboard LEDs for visual feedback of encrypted and decrypted data.

All GPIO pins were configured using Verilog constraints and managed through the user constraints file (UCF). Signal integrity was maintained by assigning appropriate IOBs with

corresponding buffer primitives (IBUF, OBUF). This low pin count and modular GPIO allocation demonstrate the feasibility of implementing secure serial communication even on resource-constrained FPGA platforms. The power consumption is about 25nW and the latency or throughput is about 100 MBPS.

VIII. Acknowledgement

We would like to express our sincere gratitude to Dayananda Sagar Academy of Technology and Management for their valuable guidance and support throughout this research. Special thanks to Department of Electronics and Communication for their insightful discussions and technical assistance and for providing the necessary resources and infrastructure for this study.

References

1. Rivest, Ronald L., Adi Shamir, and Leonard Adleman 1983 "Cryptographic Communications System and Method." U.S. Patent Office.
2. Stallings, William. 2011. *Cryptography and Network Security: Principles and Practice*. 5th ed. Boston, MA: Pearson.
3. Mollin, Richard A. 2002. *RSA and Public-Key Cryptography*. Boca Raton, FL: Chapman & Hall/CRC.
4. Huang, W. 2010. "Analysis and Research on UART Communication Protocol." *IEEE Conference on Industrial Electronics and Applications.*
5. Galla, L. K., Koganti, V. S., and Nuthalapati, N. 2018. "Implementation of RSA." *International Journal of Computer Applications.*
6. Zhou, X., and Tiang, X. 2010. "Research and Implementation of RSA Algorithm for Encryption and Decryption." *IEEE International Conference on Industrial Electronics and Applications.*
7. Chinmay, V., and Sachdeva, S. 2014. "A Review Paper on Design and Simulation of UART for Serial Communication." *International Journal of Innovative Research in Technology.*
8. Ashwini,B., and Srikant, P. 2015. "Implementation of Universal Asynchronous Receiver and Transmitter (UART)." *International Journal of Engineering Research and Applications.*
9. Reddy, K. N., et al. 2023. "Multi-Channel UART Using FPGA Implementation." *International Journal of Scientific Development and Research.*
10. Yamini, R., and Ramya, M. V. 2020. "Design and Verification of UART using System Verilog." *International Journal of Engineering and Advanced Technology.*
11. Laddha, N. R., and Thakare, A. P. 2013. "A Review on Serial Communication by UART." *International Journal of Advanced Research in Computer Science and Software Engineering.*
12. Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. 1996. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press.
13. Schneier, Bruce. 1996. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* 2nd ed. New York, NY: Wiley.
14. Koblitz, Neal. 1994. *A Course in Number Theory and Cryptography.* 2nd ed. New York, NY: Springer.
15. K, S., and Mate, R. P. 2024. "Implementation of Serial Communication using UART in FPGA." *International Journal of Advanced Research in Innovative Ideas in Education..*
16. Sanni, Rafiat. 2011. "ARM Based UART Data Transmission with Asymmetric Key Encryption Using RSA Algorithm." Bachelor's thesis, Vaasan Ammattikorkeakoulu University of Sciences. https://www.theseus.fi/bitstream/handle/10024/34903/Thesis.pdf.
17. A. Hamid and A. T. Sayed, “FPGA Implementation of RSA Encryption Algorithm,” in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, 2018, pp. 689–692
18. M. R. Bhat and S. S. Manvi, “Efficient Implementation of RSA Algorithm Using FPGA,” *Int. J. Comput. Appl.*, vol. 75, no. 4, pp. 1–5, Aug. 2013.
19. A. Kumar and R. Jha, “Design and Simulation of UART Protocol Based on Verilog HDL,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 2, no. 6, pp. 2110–2114, 2013
20. M. Z. Rehman and M. Ahmad, “Design and FPGA Implementation of an Efficient UART,” *J. Comput. Commun.*, vol. 5, no. 4, pp. 1–9, 2017.
21. R. Tiwari and R. Joshi, “Secure Hardware-Based RSA Implementation Using FPGA for Communication,” in *Proc. Int. Conf. Adv. Comput. Commun. Technol.*, 2021, pp. 1–5.
22. K. S. Anwar and S. Mahamud, “Cryptographic Algorithm Implementation on FPGA: A Survey,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 4, no. 3, pp. 129–136, 2015.
23. J. S. Gupta and M. Patel, “Design and Analysis of Efficient Serial Communication Protocol,” *Int. J. Electr. Comput. Eng.*, vol. 9, no. 6, pp. 5204–5210, 2019.
24. P. S. Rao and M. K. Rani, “Design of Low Power UART for Secure IoT Nodes,” *Microprocess. Microsyst.*, vol. 77, p. 103122, 2020.
25. A. M. Deshmukh and S. K. Sutar, “FPGA Implementation of Modular Exponentiation for Cryptographic Systems,” *Int. J. Eng. Res. Technol.*, vol. 5, no. 4, pp. 140–143, 2016.
26. Sanni, Rafiat. "ARM based UART data transmission with asymmetric key encryption using RSA algorithm." (2011).
27. Ramakrishna, M.,Jagan Mohan Rao. "Implementation Of Aes Algorithm In Uart Module For Secure Data Transmission." (2014).