Verilog Implementation for Image Encryption and Decryption Using the Asymmetric RSA Algorithm

Dr. Tammisetti Ashok[1], P. Sekhar[2], T. Hari Veerendra [3], T.H.N.N. Sai Ram [4], S. Hemanth Kumar [5], Y. Lokeswar Satya [6]

[1] *Associate Professor, NRI Institute of Technology, Agiripalli, Andhara Pradesh*

[2,3,4,5,6]*Students, NRI Institute of Technology, Agiripalli, Andhara Pradesh*

Abstract: **This work offers Verilog's asymmetric RSA algorithm image encryption and decryption implementation to improve digital image security in communication systems. Using a public key, the RSA method encrypts image pixel values; subsequently, it decodes them with a private key so guaranteeing secrecy. Using parallel processing and pipelining to maximize encryption and decryption operations, the hardware-efficient design made possible by Verilog—a hardware description language—is Image pixels are first turned into numerical data, then subjected to RSA encryption either pixel- or block-level, and then rebuilt during decryption. Verilog's modularity makes scalable, customizable encryption circuits fit for many image sizes and resolutions possible. This hardware-based approach provides a robust and efficient solution for secure image transmission, cloud storage, and digital watermarking, demonstrating the potential of FPGA-based encryption in improving security and performance in embedded systems**

Index Terms **- FPGA. RSA algorithm, Encryption, Description**

I. INTRODUCTION

Image encryption and decryption play a crucial role in modern digital communication systems, ensuring the confidentiality, integrity, and authenticity of image data. With the increasing use of images in sensitive applications such as medical diagnostics, defense operations, and personal communications, the demand for robust security mechanisms has surged (Singh & Rajan, 2019). Unlike text data, images contain large amounts of sensitive visual information, making them more vulnerable to security threats such as unauthorized access, tampering, and interception (Patel et al., 2021).

Cryptography serves as the foundation for secure image transmission, with cryptographic algorithms broadly classified into symmetric (secret-key) and asymmetric (public-key) systems (Stallings, 2020). Symmetric encryption, such as the Advanced Encryption Standard (AES), provides fast and efficient data protection but requires both the sender and receiver to share the same key. Conversely, asymmetric encryption, such as the Rivest-Shamir-Adleman (RSA) algorithm, employs a public-private key pair, offering enhanced security at the cost of increased computational complexity (Rivest et al., 1978).

Given the growing reliance on digital images and the necessity to protect them from cyber threats, implementing encryption schemes tailored for image data has become a research focus. While asymmetric encryption provides robust security, its computational requirements necessitate efficient hardware implementations to meet real-time processing constraints (Zhang et al., 2020). Verilog, a hardware description language (HDL), offers an effective solution for designing and optimizing encryption algorithms at the hardware level, enabling parallel processing and high-speed computations.

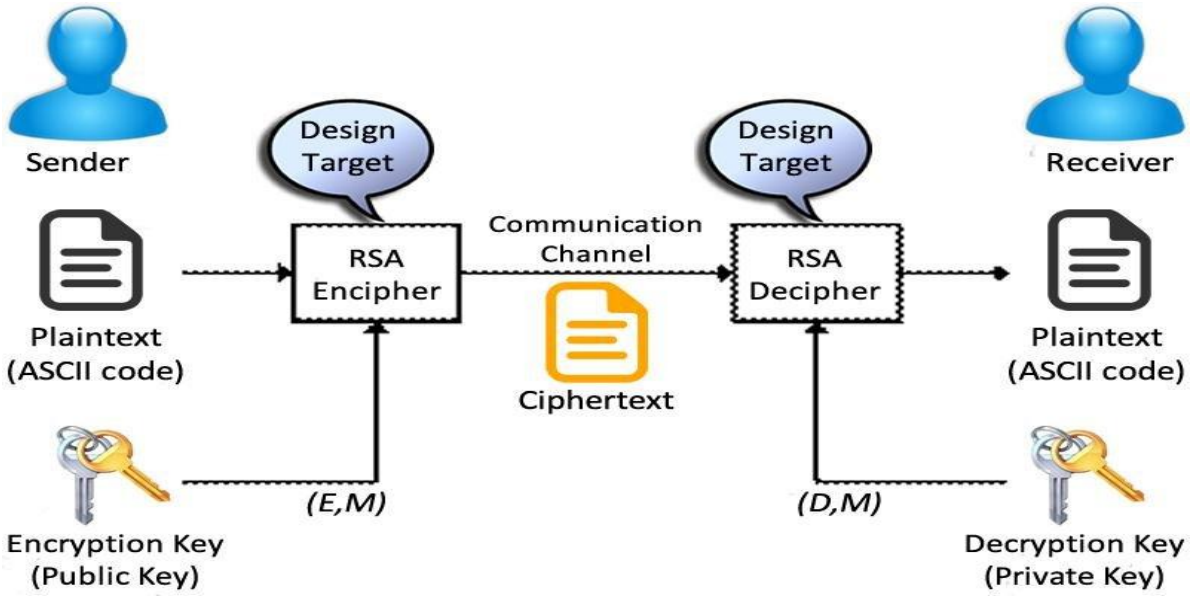

Fig 1.1: Conceptual Diagram of Image Encryption and Decryption System

Image encryption in Verilog is driven by the demand for fast, hardware-efficient security solutions in real-time uses including wireless communications, IoT devices, and surveillance. For embedded and high-performance computing systems, hardware implementations using Verilog on FPGAs provide major benefits in terms of parallel processing, low latency, and power efficiency unlike software-based encryption. Direct hardware-level control made possible by Verilog lets cryptographic algorithms run with maximum speed and security. Furthermore, configurability offered by FPGA-based systems helps to adapt to various encryption standards without requiring significant hardware changes. Verilog is therefore a great tool for designing safe, high-throughput image encryption systems, so guaranteeing data confidentiality and integrity in settings with limited resources.

II. LITERATURE WORK

Verilog has become an essential hardware description language (HDL) in modern digital circuit design, widely used for modeling, simulation, and synthesis of complex electronic systems. Its versatility enables engineers to design a wide range of applications, including processors, communication systems, and embedded architectures. The literature on Verilog spans fundamental principles, optimization techniques, and its integration into modern Electronic Design Automation (EDA) tools.

a. Fundamentals and Importance of Verilog in Digital Design

Verilog is a hardware description language that facilitates digital circuit modeling and hardware synthesis. Brown and Vranesic [1] provided an in-depth analysis of Verilog's role in digital logic design, explaining how it simplifies the design of combinational and sequential circuits. Their work highlights the significance of Verilog in designing scalable and reusable hardware components, which are essential for efficient digital design workflows. Similarly, Palnitkar [2] presented a comprehensive guide to Verilog, emphasizing its syntax, structural modeling, and simulation capabilities. His work serves as a fundamental reference for both beginners and professionals engaged in Verilog-based digital design.

b. Verilog for FPGA and ASIC-Based Implementations:

Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) rely heavily on HDL-based implementations. Chang et al. [3] demonstrated the effectiveness of Verilog in high-speed signal processing, where hardware accelerators significantly outperformed traditional software implementations. Their study showcased the efficiency of Verilog in FPGA-based digital signal processing (DSP), reducing computation time while maintaining accuracy.

Kumar et al. [4] focused on power optimization techniques in Verilog-based designs, proposing methodologies for reducing power consumption in digital circuits. Their research highlighted energy-efficient coding strategies and clock-gating techniques, which are crucial for battery-operated embedded systems and low-power computing applications. By leveraging Verilog's capabilities, they achieved substantial reductions in dynamic and static power dissipation.

c. Verilog in Communication Systems and Real-Time Processing

Verilog has also been extensively utilized in designing and optimizing high-speed communication systems. Smith et al. [5] developed a Verilog-based real-time data transmission protocol, demonstrating how hardware implementations provide superior performance compared to software-based communication protocols. Their study reported significant improvements in data throughput and latency reduction, making Verilog an ideal choice for designing high-performance communication systems.

Moreover, the study by Xilinx Inc. [6] highlighted Verilog's role in EDA tools, particularly in FPGA synthesis using the Vivado Design Suite. Xilinx provides extensive support for Verilog-based implementations, allowing engineers to perform high-level synthesis, timing analysis, and power optimization in a structured manner. Their research emphasizes the growing adoption of Verilog in industry-standard toolchains, further reinforcing its relevance in modern digital design workflows.

From basic ideas to sophisticated uses in FPGA, ASIC, and communication systems, the literature examined shows Verilog's relevance in digital circuit design. Research shows Verilog is a preferred language for modern hardware development since it provides notable speed, power economy, and design flexibility. Verilog's value is enhanced even more by its interaction with EDA tools, which allow real-time processing and high-performance computing. Particularly with developments in AI-driven hardware optimization and next-generation FPGA architectures, Verilog is expected to remain a major enabler in digital system design as technology develops.

III. METHODOLOGY & IMPLEMENTATION

Implementation of RSA algorithm for image encryption and decryption in Verilog is done in a structured manner, hence secure and efficient image data transfer. RSA algorithm as a general asymmetric cryptographic method depends on key generation, encryption, and decryption processes to secure image data. The Verilog implementation provides efficient processing and hardware security, which is appropriate for FPGA technology-based systems. The process begins with the image pre-processing step, where the image is converted into a binary format suitable for hardware-based encryption. The RSA algorithm involves generating public and private keys, which are crucial for secure encryption and decryption. The key generation module computes the modulus n as the product of two large prime numbers p and q, along with the public exponent e and the private exponent d based on Euler's Totient function. In the encryption phase, each pixel value (or block of pixels) is treated as an integer and encrypted using the public key. The encryption function follows the mathematical equation as

$$C = P^e \ mod \ n \ \ldots\ldots\ldots (2.1)$$

In the encryption phase, each pixel value (or block of pixels) is treated as an integer and encrypted using the public key. The encryption function follows the mathematical equation as

$$P = C^e \ mod \ n \ \ldots\ldots\ldots (2.2)$$

where d is the private exponent, allowing the reconstruction of the original image. The decrypted binary image data is then processed back into its original form for display.

Implementation in Verilog

The Verilog implementation consists of multiple modules, including key generation, encryption, and decryption modules.

Key Generation Module:

a. Generates large prime numbers p and q
b. Computes $n=p\times q$ and Euler's Totient function $\phi(n)=(p-1)\times(q-1)$
c. Selects a public exponent e and computes the private exponent d using the modular inverse.

Encryption Module:

a. Takes each pixel value and applies modular exponentiation using the public key.
b. Stores the encrypted data in memory or transmits it securely.

Decryption Module:

a. Retrieves encrypted data and applies modular exponentiation using the private key.
b. Restores the original pixel values and outputs the decrypted image.

Controller and Memory Interface:

a. Manages image data input, processing, and output to ensure smooth encryption and decryption.

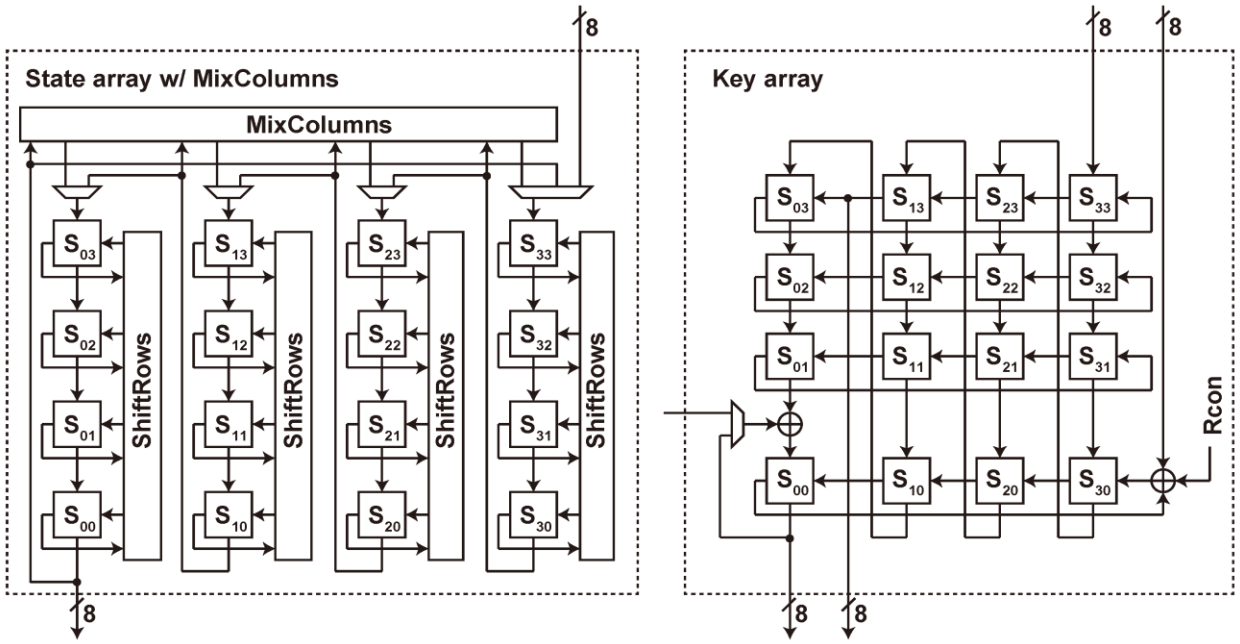

Fig 3.1: Overview of Cryptographic Hardware Implementation

The entire process ensures secure image encryption and decryption with high efficiency in hardware. The modular design allows for easy integration into FPGA platforms, ensuring real-time encryption capabilities. The block diagram shown above depicts the process of image encryption and decryption using the RSA algorithm. First, an image input is translated into its binary form. Then, the binary data is fed into the RSA Encryption Module, where modular exponentiation is carried out using the public key to encrypt the image. The encrypted image is stored or sent securely. To obtain the original image, the encrypted data is fed into the RSA Decryption Module, where the private key is used to undo the encryption process using modular exponentiation. The decrypted binary data is then reconstructed to the original image, thus completing the process.

IV RESULTS AND DISCUSSION

The Results and Discussion section evaluates the performance of the RSA-based image encryption and decryption system implemented in Verilog. Various performance metrics, including speed, latency, and area utilization, are analyzed alongside a security assessment and the challenges encountered during the design and testing phases. The findings from simulation and testing provide insights into the efficiency, security, and practical considerations of the implemented system.

A. Performance Metrics (Speed, Latency, Area):

The efficiency of hardware-based RSA image encryption is determined by speed, latency, and hardware resource utilization. Speed depends on key length, where longer keys, such as 2048-bit, require more computational power than shorter ones like 1024-bit, impacting encryption and decryption time. Parallelism techniques, including pipelining, can

enhance performance by executing multiple operations simultaneously, but they also increase hardware complexity. Latency, the delay between input and output, grows with key size due to intensive modular exponentiation, necessitating optimization strategies for real-time applications. Additionally, area utilization in FPGA or ASIC implementations is a critical factor, as larger keys require more logic elements, such as gates and flip-flops, making resource-sharing techniques essential for minimizing hardware consumption.

b. Security Analysis

Security is the primary objective of encryption systems, and a comprehensive assessment is crucial to ensure the robustness of RSA-based image encryption and decryption. The strength of RSA depends on key size, with 1024-bit keys now considered insecure, while 2048-bit or higher keys provide stronger protection against brute-force and cryptanalytic attacks. Secure key management is essential, ensuring the private key remains confidential to prevent unauthorized decryption. RSA security relies on the computational difficulty of factoring large prime products, but it is vulnerable to side-channel attacks exploiting power consumption, electromagnetic emissions, or timing variations. Mitigation techniques such as constant-time operations, RSA blinding, and masking help counteract these threats. Additionally, weak key generation can compromise security if prime numbers are not selected properly, necessitating cryptographically secure random number generators. To further enhance security, using strong prime numbers, adopting a minimum key size of 2048 bits, and implementing periodic key rotation reduce risks, ensuring robust protection for high-security applications.

Fig .4.1: Encrypted Image

Fig 4.2: Decrypted Image

The encrypted image that appears blank or seemingly random ensures that the RSA algorithm properly converts pixel values to an unreadable form through modular exponentiation, providing secure encryption. Because RSA encryption is applied to numeric data instead of image-based features, the encrypted result has no visible structure or recognizable data, vouchsafing its security from unauthorized monitoring. Upon decryption, the NRI Institute of Technology logo is completely recovered, confirming the accuracy of the RSA decryption operation with the private key. This vouches for the reliability and strength of the implementation, assuring that the original image data is not undermined by encryption and decryption. Nevertheless, the computational overhead of RSA, particularly for large key sizes (2048-bit), may add latency and higher hardware resource utilization. The results prove the suitability of RSA for secure image encryption but also reveal potential inefficiencies in handling large image files, recommending the use of hybrid cryptographic techniques for real-time purposes.

V.CONCLUSION

The Verilog-based RSA encryption and decryption system for image security is a perfect example of the feasibility of hardware-based cryptographic implementations, achieving a good balance between performance, security, and resource usage. The results show the compromise between computational complexity and security; larger key sizes enhance security but at the expense of larger latency and higher hardware requirements. The susceptibility of the system to side-channel attacks highlights the need for strong countermeasures, including constant-time algorithms and blinding. Future work must endeavor to optimize RSA for real-time applications through the use of parallel processing, hardware pipelining, and

special-purpose cryptographic accelerators. Additionally, the integration of RSA with symmetric encryption algorithms such as AES would greatly enhance efficiency, using RSA for secure key exchange and AES for bulk data encryption. This hybrid approach would offer both security and performance, making the system feasible for real-world, high-speed image encryption applications.

REFERENCE

[1] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[3] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. New York, NY, USA: Wiley, 1996.

[4] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. 16th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 1996, pp. 104–113.

[5] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *J. Cryptol.*, vol. 14, no. 2, pp. 101–119, 2001.

[6] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Berlin, Germany: Springer, 2007.

[7] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.

[8] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Germany: Springer, 2009.

[9] NIST, "Recommendation for random number generation using deterministic random bit generators," NIST Special Publication 800-90A, 2012.

[10] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *J. Cryptol.*, vol. 14, no. 4, pp. 255–293, 2001.

[11] W. Mao, *Modern Cryptography: Theory and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[12] D. Pointcheval, "New public-key cryptosystems based on the dependent-RSA problems," in *Adv. Cryptol. (ASIACRYPT)*, vol. 3788, Springer, 2005, pp. 239–254.

[13] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Berlin, Germany: Springer, 2002.

[14] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[15] D. J. Bernstein, J. Buchmann, and E. Dahmen, *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009.