FPGA-Based inline encryption bridge using AES-XTS for storage systems

DOI: https://doi.org/10.54654/isj.v2i25.1141

Tran Van Khanh*, Phan Van Ky, Vu Van Viet

***Abstract*— This paper presents a hardware-based AES-256 XTS encryption system implemented on FPGA, providing a complete inline bridge between a storage controller and the storage device. Unlike prior works that focused only on AES core optimization, this design integrates the core into the full SATA protocol and evaluates end-to-end storage-path performance. The pipelined XTS-AES core enables high-throughput, real-time sector-level encryption with minimal performance impact. FPGA implementation offers flexibility in key sizes and encryption modes, supports algorithm updates through partial reconfiguration, and allows scalability to various storage systems, including NAS storage systems. The main contributions are: (i) proposing an FPGA-based inline encryption architecture with an AES-XTS core fully integrated into the SATA protocol; (ii) implementing and evaluating the encryption performance on a real storage system, demonstrating practical feasibility and transparency in real-time operations.**

***Tóm tắt*— Bài báo này trình bày một hệ thống mã hóa AES-256 XTS phần cứng được triển khai trên FPGA, cung cấp một cầu nối inline hoàn chỉnh giữa bộ điều khiển lưu trữ và thiết bị lưu trữ. Khác với các công trình trước chỉ tập trung tối ưu lõi AES, thiết kế này tích hợp lõi vào giao thức SATA đầy đủ và đánh giá hiệu năng đầu-cuối trên đường truyền lưu trữ. Lõi XTS-AES dạng pipeline cho phép mã hóa theo sector với tốc độ cao và thời gian thực, đồng thời gần như không ảnh hưởng đến hiệu năng. Việc triển khai trên FPGA mang lại khả năng linh hoạt về kích thước khóa và chế độ mã hóa, hỗ trợ cập nhật thuật toán thông qua tái cấu hình từng phần và mở rộng cho các hệ thống lưu trữ khác, bao gồm cả các hệ thống lưu trữ kết nối mạng NAS. Các đóng góp chính là: (i) đề xuất kiến trúc mã hóa inline dựa trên FPGA với lõi AES-XTS tích hợp đầy đủ vào giao thức SATA; (ii) triển khai và đánh giá hiệu năng mã hóa trên hệ thống lưu trữ thực tế, chứng minh tính khả thi và minh bạch trong các tác vụ thời gian thực.**

***Keywords*— Storage system, hardware encryption, AES-XTS, FPGA, secure storage.**

***Từ khóa*— Hệ thống lưu trữ, mã hóa phần cứng, AES-XTS, FPGA, lưu trữ bảo mật.**

I. INTRODUCTION

With the rapid growth of digital data, storage systems have become increasingly important for managing and securing sensitive information [1]. Various encryption solutions have been implemented, including software-based tools such as BitLocker and VeraCrypt, as well as built-in encryption features provided by operating systems. Additionally, modern storage devices are often equipped with hardware-based Self-Encrypting Drives (SEDs), which enhance data security directly at the device level. However, these conventional approaches exhibit notable limitations. Software-based solutions typically introduce significant latency and impose additional load on the CPU [6], whereas SEDs lack the flexibility required to accommodate system-specific requirements. Specifically, SEDs are tied to drive firmware (Opal/IEEE1667) and cannot be reconfigured for new encryption modes, tested in research environments, or deployed on commodity SSDs without firmware support. This distinction also highlights the difference from Self-Encrypting Drives (SEDs): while SEDs provide built-in encryption confined to specific hardware, the proposed bridge ensures broader applicability by operating with commodity drives and allowing reconfiguration for research and deployment flexibility.

This manuscript was received on September 19, 2025. It was reviewed on September 25, 2025, revised on October 10, 2025 and accepted on October 29, 2025.

* Corresponding author

To address these limitations, this study proposes a hardware-based encryption between the controller and the storage device, extending beyond prior works [2, 10, 11]. While [2, 10, 11] and similar works focused on optimizing the AES core, without integrating it into a full storage protocol (SATA) or measuring the overall impact on storage system performance. In contrast, this study presents a fully integrated, system-level encryption bridge. The solution implements the AES-256 algorithm in XTS mode-an advanced encryption standard recommended for storage systems [8, 9]. The entire encryption and decryption process is performed at the sector level without requiring any modifications to the existing system software. Although initially designed to meet the high-throughput data-sharing and security requirements, the modular architecture can also be applied to broader storage environments, such as NAS, SAN, or DAS, where similar challenges in performance and flexibility exist.

The threat model assumes an adversary with physical access to the SSD but not to the FPGA bridge or host system. The primary goal is data confidentiality at rest, protecting against theft or unauthorized reads of stored data. XTS mode provides strong confidentiality but lacks built-in integrity; potential risks like replay attacks or tampering are mitigated in future extensions with authenticated encryption (e.g., AES-GCM) or separate MACs. Compared to prior FPGA XTS-AES works [13 - 15], our contribution includes the full system-level integration of the AES-XTS core into the SATA protocol for transparent storage system operation, together with pipelined tweak generation maintaining under 2% performance overhead. The proposed architecture offers several notable advantages. First, performing encryption entirely on the FPGA significantly reduces the computational burden on the main CPU. Second, the system achieves high performance through an optimized pipelined architecture. Third, the inherent flexibility of the FPGA enables seamless algorithm upgrades or system reconfiguration without requiring hardware replacement. Notably, this solution can be seamlessly integrated into existing storage systems. Although FPGA serves as a prototyping platform for its reconfigurability and rapid development, the design can be ported to Application-Specific Integrated Circuits (ASICs) for cost-effective deployment in mid- to low-end storage products, balancing performance with affordability.

In this study, a prototype system was implemented using a Xilinx ZC706 FPGA development board connected to a Solid-State Drive (SSD) via a SATA interface. Experimental results demonstrate that the system achieves a stable throughput of 540 MB/s, with significantly improved latency compared to conventional software-based solutions.

This paper is organized as follows: Section II presents the overall system architecture. Section III describes the encryption mechanism and data management. Section IV analyzes the experimental results. Finally, the conclusion and potential future work are discussed in Section V.

II. HARDWARE DESIGN ARCHITECTURE

A. Overall Hardware Architecture of the System

The proposed system performs full hardware-based data encryption and decryption using an FPGA as an intermediary between a controller and a SATA-connected SSD. The architecture is implemented on the Xilinx ZC706 development board, based on the Zynq-7000 System-on-Chip (SoC) platform, which combines an ARM processor with programmable logic on a single chip. Communication with the SSD is established via a Quad SFP/SATA FMC expansion module and a SATA cross adapter, as shown in Figure 1.

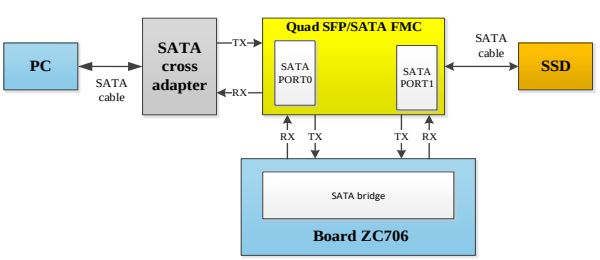


Figure 1. Experimental setup using ZC706 board and Quad SFP/SATA FMC module for hardware encryption integration

During the data write process from the host computer to the SSD, the data is first encrypted by the FPGA before being written to the storage device. In contrast, during the data retrieval process, the encrypted data is decrypted by the

FPGA and then forwarded to the host computer. This system architecture enables the real-time interception, transformation, and secure handling of the SATA data stream by leveraging an XTS-AES encryption core that is implemented directly within the FPGA, without necessitating any modifications to the host computer or the SSD firmware. This approach ensures a seamless integration of encryption and decryption operations while maintaining the performance and compatibility of the existing system.

This hardware-based bridge operates transparently between the data source and the storage device, remaining completely independent of the host operating system and application software. In this configuration, the FPGA functions as a dedicated encryption processor, enabling secure data transmission with low latency and high throughput.

B. Detailed System Architecture on FPGA

To support high-speed and secure data processing entirely in hardware, the FPGA design follows a layered and modular architecture. Each module is specifically built to handle a particular function within the encryption or decryption processing chain, ensuring efficient handling of SATA communication protocols as well as cryptographic operations based on the AES algorithm. This modular design enables real-time data processing with minimal latency, while ensuring the integrity and security of the data throughout the process.

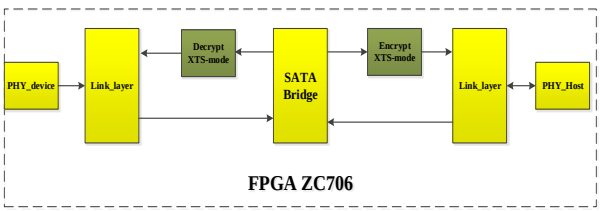


Figure 2. FPGA-based system architecture with integrated AES-XTS encryption module and SATA data bridge

Figure 2 illustrates the implementation diagram of the SATA data encryption/decryption solution on the FPGA. The main functional modules include:

1. Phy_Host / Phy_Device: These modules implement the physical (PHY) layer of the SATA interface for both the device side and the host side [16, 17, 19]. They handle physical signal transmission, serialization/deserialization (serdes), and 8b/10b encoding/decoding, ensuring reliable communication and compliance with the SATA standard.

2. Link_layer: Each Link_layer module is responsible for processing functions at the link layer of the SATA protocol [18]. Key functions include creating and parsing the Frame Information Structure (FIS), performing CRC checks to ensure data integrity, and managing data flow. These functions ensure stable communication and error resistance between the PHY layer and the higher protocol layers.

3. XTS Decryption: This module decrypts the encrypted data stream originating from the SSD before it is transmitted to the PC. It employs the AES-256 algorithm in XTS mode, restoring the data to its original form while ensuring the security and integrity of the data during the read process.

4. XTS Encryption: This module encrypts the data sent from the PC before it is written to the SSD. It operates using the AES-256 algorithm in XTS mode, providing strong protection against unauthorized access and cipher text analysis attacks [5].

5. IP Core Bridge: The Bridge IP Core serves as the central coordination module, managing the bidirectional SATA data flow in the system [15]. This module interfaces with the encryption and decryption modules, directing the data flow based on the transmission direction (read/write) and ensuring that encryption operations are performed correctly.

6. Microprocessor: The microprocessor functions as the central controller, overseeing the entire system's operations and ensuring proper functionality. It is responsible for the initialization of the system, managing configurations, and providing key parameters to the encryption and decryption cores. Furthermore, the microprocessor controls the operation of the Bridge IP Core as well as the encryption and decryption modules during runtime, ensuring that all processes are executed as intended. In addition to these tasks, the

microprocessor has the capability to monitor various status registers, enabling it to track system performance and health. It also supports control and management operations, ensuring smooth coordination of all components throughout the system's operational lifecycle.

As illustrated in Figures 1 and 2, all communication between the controller and the SSD passes through the FPGA, ensuring that every read and write request is transparently encrypted and decrypted at the sector level

This chapter presents the hardware design architecture in both a general overview and a detailed FPGA implementation. The encryption/decryption operations are executed on the FPGA. By offloading all cryptographic operations, including AES-256 encryption and decryption in XTS mode, to hardware, the design ensures flexibility, scalability, and sustainability for data storage-focused applications.

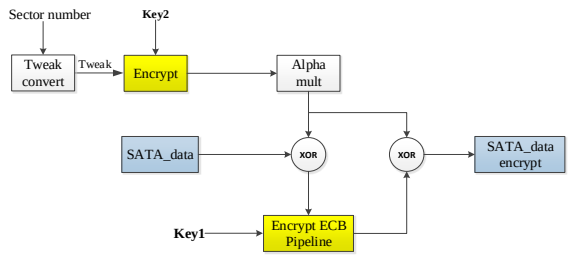
III. HARDWARE-BASED ENCRYPTION ARCHITECTURE AND DATA FLOW MANAGEMENT

A. XTS-AES Module: Architecture and Pipeline Processing

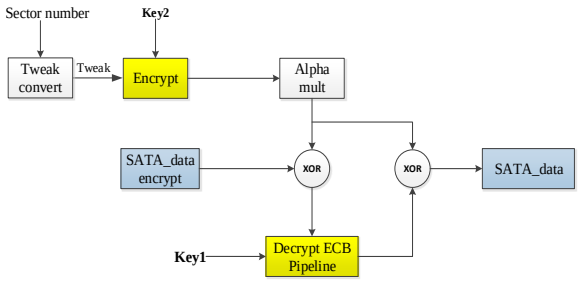
XTS-AES is an encryption mode defined in the IEEE Std 1619-2007 standard [5], designed to protect data on block-based storage devices such as hard drives or SSDs. XTS-AES is a tweakable block cipher, utilizing a 128-bit key or its multiples, with AES as the core algorithm. It is specifically designed to mitigate threats such as ciphertext manipulation and copy-and-paste attacks.

This mode uses two independent AES keys: one to generate the ciphertext for the tweak, and the other to encrypt the SATA data [14]. A proposed implementation for the encryption and decryption process in a storage security system using XTS mode is illustrated in Figure 3.

The encryption process in the XTS-AES architecture is implemented in a pipelined model, with two AES blocks operating in parallel to meet the high throughput and low latency requirements of high-speed data transmission systems such as SATA [10, 12]. This parallel architecture ensures efficient and continuous data processing, allowing the system to handle large volumes of data while minimizing delays. The first AES block (Enc1) processes user data, which represents the actual information to be stored or transmitted. Meanwhile, the second AES block (Enc2) encrypts the Logical Block Addressing (LBA) of the sector using a second key (Key2), specifically designed to generate the Tweak value required for XTS-mode encryption. This process enables secure encryption of data while ensuring that each block is independently encrypted, providing a robust defense against various attack vectors.



a) Encryption Process in XTS Mode



b) Decryption Process in XTS Mode

Figure 3. Encryption and Decryption Diagram in XTS Mode

To ensure uninterrupted pipeline operation, the process of generating the Tweak in Enc2 must be executed simultaneously and timely with the data encryption process in Enc1. Specifically, when Enc1 starts processing the data of the first sector, Enc2 must complete the LBA encryption to provide the Tweak at the correct moment. This Tweak is then XORed with the input data before being fed into the AES ECB Encrypt block. Therefore, Enc2 must be activated early and completed before the data is encrypted in the pipeline, in order to avoid data stalling.

Once the initial Tweak is generated, the system continues to produce subsequent Tweaks for each 128-bit block within the same sector by repeatedly multiplying the previous Tweak with the constant α in the finite field GF(2^{128}). This

iterative process guarantees that a single LBA value can generate a unique Tweak sequence for the entire sector, ensuring that each individual block within the sector is encrypted separately. By doing so, the system maintains the integrity and security of the data, while also allowing the pipeline to operate efficiently and without interruption. This method of generating unique Tweaks for each block enhances the security by preventing patterns from emerging in the encrypted data.

In the SATA protocol, data is typically organized into sectors of 512 bytes. Therefore, each sector consists of multiple 128-bit blocks that must be encrypted independently. The repeated multiplication of the Tweak with α in each cycle is essential to ensure the integrity and security of each data block while allowing the pipeline to operate continuously with optimal performance.

Description of the function of the multiplication block with α [14,5]:

Input: j is a power of α (alpha)

Byte array: $a_0[l]$, where l = 0, 1, 2, ..., 15

Output: Byte array: $a_j[l]$, where l = 0, 1, 2, ..., 15, where $a_0[0]$ is the first byte of the AES block.

The output array is defined iteratively according to the following formula, where the index i is incremented from 0 to j. Equation (1) is used to calculate the output array recursively as follows:

$$\begin{aligned} &(2(a_i[0] \bmod 128) \oplus (135.\, floor\left(a_i[15]/128\right) \rightarrow a_{i+1}[0] \\ &(2(a_i[l] \bmod 128) \oplus (\, floor\left(a_i[l-1]/128\right) \rightarrow a_{i+1}[l] \\ &l = 1,2,...,15 \end{aligned} \quad (1)$$

XOR Operation 1 – Data Preprocessing: The data from the SATA interface (SATA_data) is XORed with the corresponding Tweak to introduce variability between blocks, even if the content repeats.

The data after the XOR operation is encrypted using the AES algorithm with Key1. The AES encryption process is fully described in [3]:

Key Expansion: The main key is expanded into multiple subkeys, including the initial key and round keys.

AddRoundKey: In each round, an XOR operation is performed between the current data state and the corresponding round key.

SubBytes: This is a nonlinear transformation applied to each byte of data independently, using the S-box lookup table to enhance security.

ShiftRows: The rows of the data matrix are cyclically shifted by different byte values to spread the data. The first row remains unchanged, the second row shifts by 1 byte, the third row by 2 bytes, and the fourth row by 3 bytes.

MixColumns: Each column of the state matrix is transformed using matrix multiplication in the finite field GF(2^8), helping to spread the data vertically.

The encrypted data is then combined again with the same Tweak used earlier to ensure the 'tweakable' property of XTS-AES, meaning the ciphertext depends on both the data and its logical position. The final result is the encrypted ciphertext, SATA_data_encrypt.

The decryption process for XTS-AES reverses the steps described during encryption. The result will yield the plaintext SATA_data.

The XTS-AES module is a critical component responsible for performing encryption and decryption at the block data level in hardware. Based on the AES-256 standard operating in XTS mode, this design ensures the secure transformation of storage data while maintaining compatibility with sector-based storage operations.

The pipelined structure of AES processing

To optimize throughput, the AES-XTS core is implemented as a pipelined architecture [4,11,13,15,20]. Each round of the AES algorithm (including SubBytes, ShiftRows, MixColumns, and AddRoundKey) is mapped to a pipeline stage, allowing for the concurrent processing of multiple 128-bit data blocks.

This pipelined structure enables sustained data rates that align with the bandwidth demands of the SATA interface while ensuring minimal latency overhead for encryption and decryption operations.

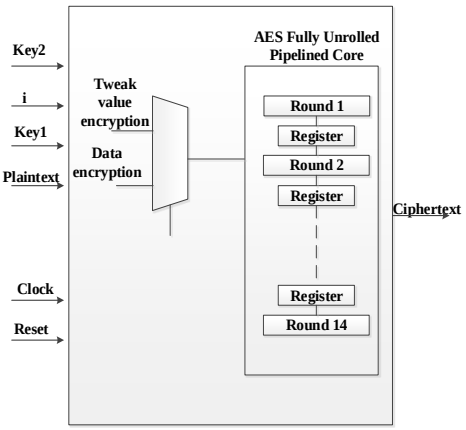


Figure 4. Block diagram of the XTS-AES pipelined processing

Figure 4 illustrates the pipelined architecture of the AES encryption core in the XTS-AES mode [14], which enables parallel data processing and achieves high performance. The input data consists of two keys (Key1 and Key2), the exponent index of the generator element α (denoted as i), the plaintext, and control signals (clock, reset). Key2 and the index i are used to generate the corresponding Tweak value for each data block. This Tweak value is combined with the plaintext through the XOR operation, forming the input data for the AES encryption block.

The AES encryption core is implemented in a fully pipelined architecture with 14 processing stages corresponding to AES-256. Each stage is separated by a pipeline register to hold intermediate states and allow each clock cycle to process a new data block. After passing through all the encryption stages, the data is output as ciphertext.

This implementation leverages the parallelism and data flow capabilities of the pipelined architecture, significantly increasing processing throughput. Furthermore, the integration of the Tweak before encryption and the XOR application after encryption ensures the characteristic security of the XTS mode, protecting against ciphertext manipulation or copy-and-paste attacks.

B. Bridge_data Module

To ensure secure and orderly communication between the server system and the SSD, the Bridge_data module serves as the central control unit for bidirectional data transfer. Positioned between the main board (PC) and the SSD, this module works in conjunction with the SATA Bridge IP core to manage data flow arbitration, intermediate buffering, and handshake protocols to ensure data integrity. Its role becomes particularly crucial in a system performing real-time XTS-AES encryption and decryption, where timing accuracy and data consistency are key factors.

Overview of Functionality and Management Mechanisms

The Bridge_data module is designed to manage the entire data flow passing through the SATA Bridge IP core by:

Synchronizing access to the shared communication interface, ensuring that the SSD and PC do not perform simultaneous operations that could cause errors or data corruption.

Dividing the control flow processing between the two transfer directions (read and write), ensuring that only one valid operation is performed at a time-either from the SSD to the PC or vice versa.

Implementing handshake protocols based on acknowledgment (ACK) signals to verify transaction integrity and prevent data packet collisions.

This mechanism for dividing processing is essential to avoid read/write conflicts, especially in high-throughput situations or when strict real-time encryption is required.

Data Flow Structure

The Bridge data module is responsible for managing two main communication streams: from the SSD to the PC and from the PC to the SSD.

1. Transmission from SSD to PC

When the SSD initiates a transmission, such as sending a control packet (FIS) or data, the SSD sends a WRITE request to the Bridge IP core. In response, the bridge immediately acknowledges this request with an ACK signal. The bridge then forwards the data to the PC via a WRITE operation. Once the PC receives the data and sends an acknowledgment signal back, the bridge sends a final ACK signal to the SSD, completing

the transaction. This process ensures reliable data transmission from the SSD to the PC while maintaining the integrity of the data flow.

2. Transmission from PC to SSD

Conversely, when the PC initiates data transmission, it sends a WRITE request to the Bridge IP core. However, the bridge does not immediately acknowledge this request; instead, it waits for the SSD to signal that it is ready to receive the data. Once the SSD is ready, the bridge performs the WRITE operation to the SSD. After receiving the acknowledgment signal from the SSD, the bridge sends a final ACK signal to the PC, confirming that the transmission has been successfully completed. This step ensures that the data is correctly written to the SSD and that synchronization between the devices is maintained.

3. Dual-Handshaking Mechanism

This dual-handshaking mechanism ensures the consistency and security of the encrypted data during transmission through the pipeline. By performing handshakes in both directions (SSD to PC and vice versa), the Bridge_data module guarantees accurate data sequencing, synchronized transmission times, and the preservation of the integrity of encrypted information under hardware encryption conditions.

C. Security Considerations

Key management is handled by the microprocessor, which loads Key1 and Key2 from a secure source (e.g., encrypted flash memory or external trusted input during boot). Keys are stored in volatile registers to prevent persistent exposure, and the system supports key rotation via FPGA reconfiguration.

XTS provides confidentiality but no integrity or replay protection. To address tamper risks, the design relies on SSD's built-in ECC for bad sectors and atomic sector writes to handle power loss (ensuring partial writes are discarded). Under reset, the microprocessor reinitializes keys and states. Replay attacks are mitigated by sector-level tweaks tied to LBA, but full integrity requires extensions like MACs.

Compared with SEDs, which typically provide fixed encryption functionality with limited configurability, the proposed FPGA-based design offers greater flexibility in key sizes, encryption modes, and algorithm updates through partial reconfiguration. Moreover, unlike SEDs that operate as "black boxes," this architecture enables full system-level integration within the storage path, allowing direct interaction with the controller and detailed performance evaluation.

Compared with recent AES-XTS FPGA implementations that report extremely high core-level throughput (tens or even hundreds of Gbps) [7, 10, 11, 21, 22], the novelty of this work lies in its system-level inline integration. The design operates directly on the SATA 3.0 path (theoretical bandwidth 6 Gbps ≈ 600 MB/s) and achieves an end-to-end throughput of 490 MB/s for sequential read and 433 MB/s for sequential write, as measured with CrystalDiskMark 8. These results show only minor overhead compared to baseline SSD performance.

TABLE I. COMPARISON WITH RELATED FPGA AES-XTS IMPLEMENTATIONS

Work	**Freq (MHz)**	**Clock cycles**	**Latency (ns)**
[11]	740	42	56.7
Our	400	16	40

To further illustrate performance differences, Table I compares our implementation with a representative prior work [11]. While their core runs at a higher frequency (740 MHz vs. 400 MHz) and therefore achieves higher theoretical throughput (~95 Gbps vs. 51.2 Gbps), our design completes encryption in fewer clock cycles (16 vs. 42), resulting in lower absolute latency (40 ns vs. 56.7 ns). This highlights a trade-off: their design maximizes raw core speed on a high-end Virtex-7 (XC7VX690T), whereas our design emphasizes cycle efficiency and full system integration on a Zynq-7000 platform. Importantly, both throughputs far exceed SATA 3.0 requirements, but only our work demonstrates transparent, end-to-end storage-path performance.

Section IV provides the detailed implementation, test setup, and system-level performance analysis.

IV. IMPLEMENTATION AND TESTING

A. Implementation on FPGA

The proposed XTS-AES encryption model has been implemented on the Xilinx ZC706 evaluation board using Vivado 2022.2. The hardware design integrates the XTS-AES encryption/decryption core, data flow management logic, and SATA bridge to control data transmission between the PC and the SSD. The system operates at a clock frequency of 150 MHz, with a per-block latency of 14 cycles (~93 ns) and a per-512B sector latency of approximately 0.93 μs (pipelined throughput 1 block/cycle).

Resource utilization (from Vivado synthesis) is detailed as follows:

TABLE II. RESOURCE UTILIZATION OF XILINX ZC706 FPGA FOR XTS-AES IMPLEMENTATION

Resource	Utilization	Available	Utilization
LUT	38,063	101,400	37.54%
FFs	18,472	202,800	9.11%
BRAMs	214	325	65.85%
GTs	2	4	50%

The design demonstrates effective resource management on the Xilinx ZC706 FPGA. The utilization of 38,063 LUTs (37.54%) and 18,472 FFs (9.11%) reflects a well-optimized logic implementation, utilizing a substantial portion of the available resources while retaining capacity for potential enhancements. The 214 BRAMs (65.85%) indicate significant memory usage, likely attributed to storing encryption keys, tweak values, and data buffers, which aligns with the high-throughput demands of the SATA interface. The 2 GTs (50%) utilization showcases efficient deployment of high-speed transceivers for SATA communication. Overall, this resource allocation supports the system's performance objectives while offering scalability for future improvements or ASIC migration.

Power consumption is approximately 5W baseline, increasing to 5.5W under encryption load (measured via Xilinx Power Estimator).

The system consists of the following main functional blocks:

1. XTS-AES Encryption/Decryption Core;

2. SATA-bridge module managing data flow between the PC and SSD;

3. Soft processor to load key parameters for the XTS-AES algorithm and control peripherals (UART, GPIO).

During the setup, the SATA-bridge block checks the connection with the mainboard (PC) and establishes the connection with the SATA drive. Once the connection signals are received and combined with the loaded key parameters, the XTS-AES core is activated to enable encryption/decryption of the data. At this point, the system operates as a bridge, allowing data from the PC to be encrypted before being written to the SSD and decrypted when read.

This hardware configuration enables the practical implementation of the inline encryption model between the PC and SSD while supporting testing in two modes: encrypted and non-encrypted. This setup is ideal for evaluating real-time data protection capabilities in a storage system, where hardware encryption is required without affecting throughput.

To verify and evaluate the correctness of the data encryption/decryption process, the procedure is performed in the following steps:

1. Connect the integrated encryption/decryption board to the PC and SSD

Connect the FPGA board to the PC and SSD using a SATA cable.

2. Load bitstream and firmware to the device

Load the bitstream and firmware to the device to check the connection and configure the key parameters for the algorithm. Connection and configuration are verified through indicator LEDs.

3. Send test data

From the PC, write test data to a random sector on the SSD through the bridge system

using HxD Hex Editor software, which allows access to the sector at a low level. The test data written from the PC is transmitted through the FPGA, encrypted by the XTS-AES core, and written to the SSD.

4. Direct SSD access

The data written to the disk sector is encrypted. To verify the encrypted data, connect the PC directly to the SSD.

5. Verify the encryption result

Using HxD Hex Editor again, inspect the data at the sector address that was written on the SSD. The data obtained is the encrypted data. Compare this data with the expected data to validate the correctness of the encryption algorithm.

This experiment demonstrates that the proposed architecture can perform real-time, transparent encryption and secure data storage, confirming the reliability of the hardware encryption pipeline for storage applications, including NAS.

B. System Implementation and Validation

To validate the XTS-AES hardware encryption model in the the storage architecture, a prototype has been implemented on the Xilinx ZC706 FPGA board. The FPGA is configured to operate in bridge mode, connecting the host PC to a SATA Gen 3 SSD. In this configuration, the FPGA performs data encryption during the transmission from the PC to the SSD. This section presents the layout, testing methods, and evaluation results to verify the functionality, performance, and correctness of the proposed system.

The firmware, which includes the XTS-AES encryption pipeline and the Bridge_data module, is synthesized in Vivado 2022.2 and deployed on the FPGA. During operation, the system processes raw SATA traffic without altering the data structure, automatically encrypting the data when written to memory. Encryption correctness was first verified using IEEE Std 1619-2007 XTS-AES test vectors, and the FPGA outputs matched the reference values. Additionally, the implementation was cross-validated with the Crypto Tester tool, which confirmed bit-accurate encryption and decryption. Further sector-level validation was performed using the HxD hex editor on encrypted disk images.

The experimental evaluation was conducted using a host PC equipped with an Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz, 8 GB RAM, running Ubuntu 20.04. The FPGA platform was a Xilinx Zynq ZC706 development board, connected inline between the host and the SSD. The storage device under test was a TeamGroup CX2 256 GB 2.5-inch SATA III SSD. Benchmarking tools included CrystalDiskMark 8 for sequential and random read/write tests, and additional fio workloads for validation.

The testing process consists of three main categories:

1. Verification of SSD Detection and Capacity Recognition

The first test aims to verify the system's ability to accurately detect the SSD via the SATA Gen 3 interface. Upon system startup, the SSD should appear in Windows Disk Management with the correct reported capacity. The success condition is that the SSD is detected correctly without manual intervention or configuration.

2. Read/Write Performance Measurement

The CrystalDiskMark 6.0.2 tool is used to evaluate throughput, with the following configuration: test size of 1 GB, repeated 2 times, and a queue depth of 32. The system is required to meet the following minimum thresholds:

Sequential read/write $\geq$ 400 MB/s

Random 4K read/write $\geq$ 200 MB/s

Achieving these thresholds demonstrates that the encryption pipeline and bridge module do not cause significant performance degradation and still maintain SATA Gen 3 throughput.

3. Bridge_data Module Functionality Test in Non-Encryption Mode

To test raw data transmission without the influence of encryption, a write/read validation test is conducted. A sample data (e.g., 0x11) is written to sector 0 of the SSD from the PC via

the FPGA bridge using HxD software. Afterward, the FPGA is turned off, and the SSD is connected directly to the PC. Reading back the sector confirms that the data remains unchanged, verifying the correctness of the bridge module in transparent mode.

4. Bridge_data Module Role and Timing Test

The Bridge_data module is a bi-directional control interface responsible for managing the handshake communication between the PC and SSD. This module ensures stable, lossless data transmission through read/write command arbitration and ACK-based signaling protocols. This mechanism is particularly crucial for maintaining accurate timing and data integrity, especially in real-time encryption processing scenarios.

A unified command sequence diagram has been developed to illustrate the transaction flow between the PC, the bridge IP core, and the SSD. This diagram serves as a reference to verify the timing synchronization and operational consistency across the entire encryption pipeline and storage system.

C. Experimentation and Results

To establish a performance baseline, the system's storage throughput was evaluated without encryption. As shown in Figure 5, the SSD achieves:

Sequential read speed: ~499 MB/s

Sequential write speed: ~441 MB/s

Lower read/write speeds for smaller blocks as expected (e.g., ~166 MB/s read and ~217 MB/s write in the lowest tier).

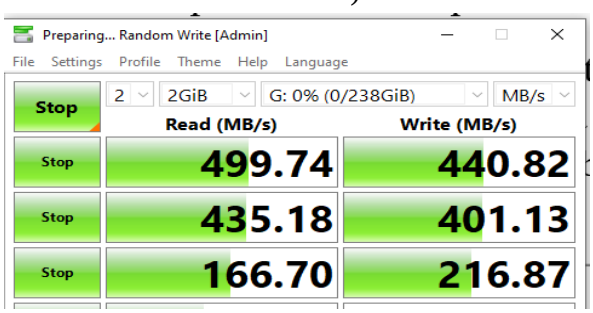


Figure 5. Baseline SSD performance measurement using CrystalDiskMark before encryption intervention

These values serve as a reference to compare the system's behavior after FPGA-based encryption logic is inserted.

Encryption Verification: Functional correctness was confirmed by writing known data (e.g., 0x11) and validating that it was properly encrypted using the AES-XTS algorithm. After removing the FPGA, the stored data remained encrypted and unreadable, confirming secure encryption at the hardware level. The implementation was further verified against IEEE 1619 test vectors, passing all XTS-AES-256 cases.

The test was conducted using CrystalDiskMark 8.0.5 x64, confirming that encryption introduces minimal impact on overall throughput.

Performance Impact: After integrating the FPGA-based encryption logic, a second performance test was conducted using CrystalDiskMark. As shown in Figure 6, the results demonstrate minimal degradation:

Sequential Read (SEQ1M Q8T1): 490.28 MB/s

Sequential Write (SEQ1M Q8T1): 433.15 MB/s

Random Read (RND4K Q32T1): 172.20 MB/s

Random Write (RND4K Q32T1): 212.90 MB/s

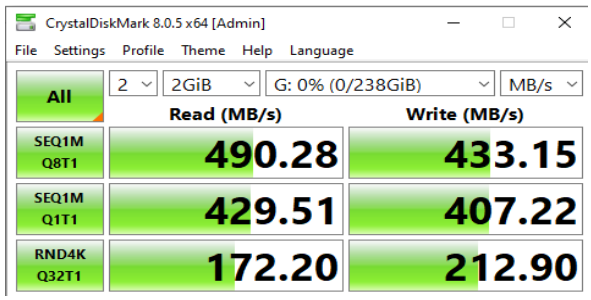


Figure 6. SSD performance benchmark results after applying AES-XTS encryption using FPGA-based logic

To evaluate real-world workloads, additional benchmarks were performed using fio with mixed read/write scenarios (70/30 ratio, queue depth 1-32, iodepth sweeps). Results show ~1-3% overhead (e.g., 450 MB/s mixed baseline vs 440 MB/s encrypted), outperforming software dm-crypt AES-XTS (10-40% overhead with AES-NI) and comparable to SEDs but with greater flexibility.

TABLE III. PERFORMANCE COMPARISON BEFORE AND AFTER ENCRYPTION (CRYSTALDISKMARK)

Test Type	Baseline (MB/s)	Post-Encryption (MB/s)	Difference (%)
SEQ1M Q8T1 Read	499.00	490.28	↓ 1.7%
SEQ1M Q8T1 Write	441.00	433.15	↓ 1.8%
RND4K Q32T1 Read	166.00	172.20	↑ 3.7%
RND4K Q32T1 Write	217.00	212.90	↓ 1.9%

As summarized in Table III, these figures confirm that the encryption logic introduces only minor latency, while maintaining sustained throughput well above 100 MB/s. The performance loss is within 2% for all major operations, indicating that the FPGA-based AES-XTS encryption is efficient and practical for real-time data protection.

Security Assurance: The AES-XTS algorithm implemented at the sector level ensures strong data confidentiality. Even partial leakage of the encrypted data yields no usable plaintext, thus maintaining a high level of data security. Failure analysis under power loss assumes atomic sector operations (discarding incomplete writes), bad sectors handled by SSD ECC, and resets trigger key reinitialization.

V. CONCLUSION

This paper has presented an FPGA-based hardware encryption model for storage systems (including NAS) using AES-256 in XTS mode. The key innovation is the seamless integration into the SATA data pipeline via a pipelined architecture, enabling real-time encryption with high throughput and minimal overhead. The solution operates transparently without modifying existing software, as demonstrated by experimental results confirming stable sector-level encryption/decryption performance.

A significant contribution is the integration of the XTS-AES encryption core with a hardware-based data bridge module, offering a software-independent security solution with extensibility for algorithm upgrades. The system maintains consistent performance across sequential and random I/O operations, accurately detects SSD capacity, and ensures reliable raw data transmission, as confirmed by empirical evaluations.

The FPGA resource utilization analysis indicates an efficient design, utilizing 37.54% LUTs, 9.11% FFs, 65.85% BRAMs, and 50% GTs, supporting high performance while allowing scalability. Future work will focus on optimizing pipeline depth for heavy loads, extending support to algorithms like AES-GCM and ChaCha20, and adding authentication mechanisms such as MACs to enhance integrity. Efforts will also target reducing resource usage and power consumption for broader storage system deployment, alongside advanced security features.

This research provides a practical, flexible, and efficient foundation for secure, high-performance storage systems, with potential for ASIC adaptation in cost-sensitive applications.

REFERENCES

[1] C. Laird, “Taking a Hard-Line Approach to Encryption”, *IEEE Computer Society*, vol. 40, 2007, pp. 13-15.

[2] Dumitru, L. Alexandru, Eftimie, Sergiu, Fostea, Dan, “An FPGA-based cloud storage gateway”, Naval Academy Publishing House, 2016.

[3] FIPS PUB 197, “Advanced Encryption Standard (AES),” National Institute of Standards and Technology (NIST), Nov. 2001.

[4] G. Saggese, A. Mazzeo, N. Mazzocca, A. G. M. Strollo, “An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm” *Field Programmable Logic and Application*, 2003, pp. 292-302.

[5] IEEE Std 1619-2007, “IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices”.

[6] L. Demir, M. Thiery, V. Roca, J. L. Roch, J. M. Tenkes, “Improving dm-crypt performance for XTS-AES mode through extended requests: first results”, *HAL open science*, 2016.

[7] Yi Wang, A. Kumar, Yajun Ha, “FPGA-based high throughput XTS-AES encryption/decryption for storage area network”, IEEE, 2014.

[8] Martin, L., "XTS: A Mode of AES for Encrypting Hard Disks", *Security & Privacy*, IEEE, vol.8, no.3, pp.68-69, May-June 2010.

[9] M. A. Alomari, K. Samsudin, A. R. Ramli, "A Study on Encryption Algorithms and Modes for Disk Encryption", IEEE, 2009.

[10] Ahmed, Shakil, Naseem, Muhammad, "Efficient AES-XTS Pipelined Implementation on FPGA", *Sir Syed University Research Journal of Engineering & Technology*, Vol. 4, No. 1, 2014 pp. 1–6.

[11] Tran, Sy Nam, Luong, The Dung, Nguyen Van Long, "A High Throughput, Low Latency 105Gbps Four-Pipeline Stage AES", *Journal of Science and Technology on Information Security*, Vol. 58, No. 1.CS (21), 2024, pp. 21–35.

[12] M. M. Mansour, M. M. Al-Qutayri, and A. Al-Ali, "A high-speed FPGA implementation of the AES algorithm," *in Proc. Int. Conf. on Electronics, Circuits and Systems (ICECS)*, pp. 1404–1407, 2005.

[13] M. Saqib, F. Anwar, and A. A. Khan, "FPGA-based implementation and performance analysis of XTS-AES encryption for data storage security," *International Journal of Computer Applications*, vol. 179, no. 42, 2018, pp. 1–7.

[14] S. Ahmed, K. Samsudin, A. R. Ramli, F. Z. Rokhani, "Effective Implementation of XTS-AES on FPGA", IEEE, 2011.

[15] S. Ahmed, M. Nassem, "Efficient XTS-AES pipelined Implementation on FPGA", IEEE, 2014.

[16] SATA-IP Bridge reference design, *https://dgway.com/products/IP/SATA-IP/dg_sata_ip_refdesign_bridge_kt7_en/.*

[17] SATA-IP Device reference design, *https://dgway.com/products/IP/SATA-IP/dg_sata_ip_refdesign_device_kt7_en/.*

[18] SATA-IP Host reference design on 7-Series, *https://dgway.com/products/IP/SATA-IP/dg_sata_ip_refdesign_host_7series_en/*

[19] SAT- IP Transport & Link Layer, *https://dgway.com/products/IP/SATA-IP/dg_sata_ip_data_sheet_7series_en/.*

[20] S. An, S. C. Seo, "Designing a new XTS-AES parallel optimization implementation technique for fast file encryption", IEEE, 2022.

[21] Khanh, T. V., Tu, N. V., & Ho, T. P. . (2022). Some issues about upgrading and developing high-speed local IP network encryption devices. *Journal of Science and Technology on Information Security*, 1(15), 46-55. https://doi.org/10.54654/isj.v1i15.838.

[22] Ky, P. V., Cuong, V. T., & Phuc, L. H. (2021). Solution for Cryptographic Intervention in PCI-Express Data Transmission on FPGA Board. *Journal of Science and Technology on Information Security*, 2(12), 59-68. https://doi.org/10.54654/isj.v2i12.108.

ABOUT THE AUTHOR

Tran Van Khanh

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: tvkhanh@bcy.gov.vn

Education: he received the B.Sc. degree in 2009, the M.Sc. degree in 2011, and the Ph.D. degree in 2015 from the Moscow Institute of Physics and Technology (MIPT), Russian Federation.

Recent research direction: His research interests include electronic engineering, cryptographic engineering, hardware security, and embedded cryptographic systems.

Tên tác giả: **Trần Văn Khánh**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: tvkhanh@bcy.gov.vn

Quá trình đào tạo: tốt nghiệp Cử nhân năm 2009, Thạc sĩ năm 2011 và Tiến sĩ năm 2015 tại trường Đại học Vật lý kỹ thuật Mátxcơva (MIPT), Liên bang Nga.

Hướng nghiên cứu hiện nay: kỹ thuật điện tử, kỹ thuật mật mã, bảo mật phần cứng và hệ thống mật mã nhúng.

Phan Van Ky

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: pvk.hvktqs@gmail.com

Education: he received his B.S. degree from Saint Petersburg Electrotechnical University "LETI" in 2013 and his M.S. degree in 2017.

Recent research direction: His research interests include SoC design, hardware design, and hardware security.

Tên tác giả: **Phan Văn Kỷ**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: pvk.hvktqs@gmail.com

Quá trình đào tạo: Tốt nghiệp Cử nhân năm 2013 và Thạc sĩ năm 2017 tại Đại học Kỹ thuật Điện Saint Petersburg "LETI".

Hướng nghiên cứu hiện nay: thiết kế SoC, thiết kế phần cứng và bảo mật phần cứng.

Vu Van Viet

Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission

Email: vietvu1912.97@gmail.com

Education: He received his B.S. degree from Le Quy Don Technical University in 2020 and is currently pursuing an M.S. degree at the Academy of Cryptography Techniques.

Recent research direction: His research interests include electronic engineering, hardware security, and hardware design.

Tên tác giả: **Vũ Văn Việt**

Cơ quan công tác: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ, Việt Nam

Email: vietvu1912.97@gmail.com

Quá trình đào tạo: Tốt nghiệp Cử nhân Đại học Kỹ thuật Lê Quý Đôn năm 2020 và hiện đang học Thạc sĩ tại Học viện Kỹ thuật Mật mã.

Hướng nghiên cứu hiện nay: kỹ thuật điện tử, bảo mật phần cứng và thiết kế phần cứng.