PYNQ-Based Modular Architecture for Real-Time Stream-Cipher Video Encryption at the Edge

Vanukuru Hari Rohit, D. Sreekiran, Raghavendra Kune, D. Mallikharjuna Rao, Novaline Jacob
Advanced Data Processing Research Institute, Department of Space, Hyderabad, India
Email: {hari.rohit, sreekiran, raghav, mallik, novaline}@adrin.res.in

***Abstract*—FPGAs are best suited for hardware acceleration of stream ciphers but are not appropriate to the asynchronous Internet Protocol (IP) traffic generated by modern Commercial Off-The-Shelf (COTS) edge cameras. Fully hardware-based encryption pipelines require integrating proprietary RTL Ethernet Intellectual Property (*IP*) cores, substantially increasing system complexity. In this paper, we present a modular PYNQ-based hardware/software co-design that decouples network stack management in the Processing System (PS) from stream cipher computation in the Programmable Logic (PL). UDP video payloads are aggregated using a fixed-stride based memory mapping scheme to enable batch encryption by AXI DMA. A sequence-aware receiver framework mitigates deep RF burst losses through timeout-driven flushing and boundary detection, preventing cipher desynchronization. Experimental evaluation on HD video streams demonstrates real-time performance consistent with analytical steady-state loss models.**

***Index Terms*—AXI DMA, AXI Lite, AXI memory mapped, AXI stream, H.264, Intellectual Property (*IP*), Processing System (PS), Programmable Logic (PL), Python Productivity for ZYNQ (PYNQ), Register Transfer Level (RTL)**

I. Introduction

Extensive literature indicates Field-Programmable Gate Arrays (FPGAs) as most widely chosen platforms for implementation of stream ciphers. The FPGA is not the ideal solution for implementation of Internet Protocol packet based stream cipher. RTL Ethernet *IP* cores, such as the AMD® Tri-Mode Ethernet MAC (TEMAC), impose considerable integration and maintenance complexity when coupled with custom stream cipher architectures.

In this paper, we propose a hardware/software modular co-design architecture using PYNQ-based SoCs. This architecture bridges software-based IP packet handling with hardware-accelerated stream ciphers, offering a developer-friendly framework for securing real-time edge video streams. This framework eliminates the acquisition cost of production scale Ethernet RTL *IP* whilst also reducing the development time as Layer-3 is handled in software using PYNQ. By confining the stream cipher entirely to the FPGA Programmable Logic (PL) and delivering it as a dynamically loaded PYNQ overlay, the system isolates the cryptographic *IP* from the software stack. This approach retains the fundamental advantage of hardware-accelerated security, i.e., using a compiled bitstream to obfuscate the developer's proprietary algorithm.

While the open-source Rabbit stream cipher (RFC 4503 [1]) is utilized herein for the performance demonstration of edge video stream cipher design, the design's modular architecture eases the integration of proprietary cryptographic *IP* with minimal modifications to the RTL interface logic, expediting the validation process.

Note that all hardware interfaces utilizing the Advanced eXtensible Interface 4 (AXI4) standard are hereafter referred to as AXI. In section II of this paper, we describe the related work available in literature. Section III details the proposed system architecture. Specifically, to overlap PS-PL communication latency, we instantiate three instances of a novel hardware accelerator pipeline—comprising an AXI DMA, a VHDL-based cryptographic key generator, and a High Level Synthesis (HLS)-based Stream synchronizer. We also describe the software multiprocessing model that batches packets and concurrently drives these pipelines using dedicated CPU core affinity. Section IV details the end-to-end cryptographic throughputs for standard video playback workloads achieved from the proposed architecture. In section V, we outline the future directions and conclude.

II. Related work

Recent literature has extensively explored the use of FPGA and PYNQ-based systems for cryptographic acceleration. In [2], they utilize PYNQ platform to implement a novel chaotic stream cipher, though they evaluate their cryptographic architecture strictly on static, locally stored benchmark images. In [3], they use the programmable logic strictly for secure key storage and pre-stream cryptographic authentication rather than payload encryption. Advances in hardware/software co-design, such as the Pynqrypt [4] framework and the PYNQ Offloaded Cryptographic Accelerator (POCA) [5], have successfully demonstrated the use of FPGA overlays to accelerate standard cryptographic libraries (viz. AES-CTR). In [6], an application-specific architecture has been proposed that combines intelligent target detection algorithm with encrypted transmission system to secure analyzed edge data.

While these existing systems excel in their target applications—such as static file-based cryptographic processing, secure authentication, or combining AI workload with encryption, they are not inherently optimized for the continuous, asynchronous packet flow of live HD video networks, nor do they offer the structural flexibility required by regulatory cryptographic standards. In contrast, our design differs from the above works by implementing an end-to-end, multi-core UDP packet dispatch pipeline tailored specifically for real-time edge video stream encryption. Rather than using a tightly coupled cryptographic core, our framework introduces

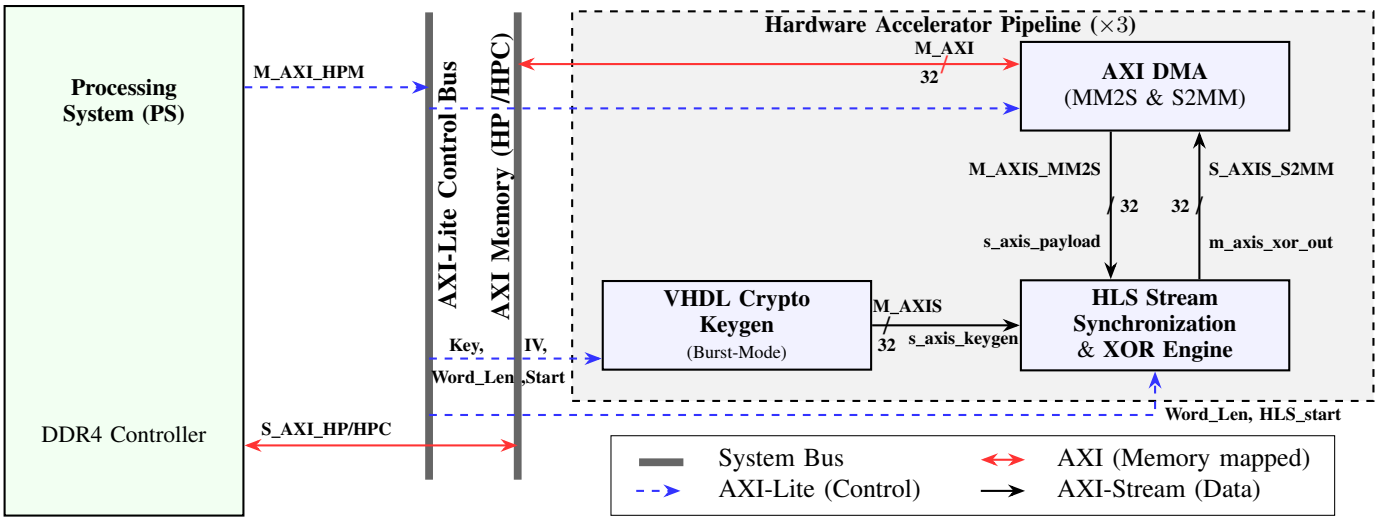

Fig. 1. Simplified block diagram of the system architecture. The PS of Zynq UltraScale+ MPSoC configures three identical parallel hardware pipelines inside PL via the AXI-Lite control bus. The AXI DMA fetches input data from, and writes processed data back to, the PS DDR4 memory via the PS AXI High-Performance (HP/HPC) ports. Within each pipeline, an AXI DMA streams 32-bit input data directly into the custom HLS-based Stream synchronization and XOR Engine. Simultaneously, a burst-mode Rabbit keystream enters the engine, where both streams are clock-synchronized and XORed. The PL operates in a single 50 MHz synchronous clock domain.

a dedicated decoupled hardware Stream Synchronization block that physically separates the key generation logic from the payload encryption. The modular architecture improves real-time streaming performance while providing a well-partitioned design framework that can simplify certification for custom or proprietary stream cipher modules by central agencies.

III. Proposed System Architecture

The selected Zynq UltraScale+ MPSoC (Multi-Processor System on Chip) development kit for realization of the proposed design is ALDEC®TYSOM-3-ZU7EV board[7]. It has a hard Processor system in the form of Quad Core ARM Cortex-A53 as an Application Processing Unit. It also houses a SODIMM 8GB DDR4 RAM attached to the Processing System (ARM CPU). Fig.1 illustrates the abstract block diagram of the proposed system architecture implemented using the AMD® Vivado 2022.1 *IP* Integrator.

The Hardware Accelerator Pipeline inside Programmable logic constitutes the following:

1) The VHDL-based Rabbit keystream generator uses a 4-cycle pipeline to output 32-bit AXI-stream-aligned words.
2) An AXI Direct Memory Access (DMA) core bridges the memory-mapped Processing System (PS) and the streaming domain of the Programmable Logic (PL). The AXI DMA is configured via AXI-Lite by the PS to manage bidirectional access to the DDR4 RAM: a memory-mapped-to-stream (MM2S) transfer reads 32-bit words from the PS DDR4 into the HLS-based Stream Synchronization Engine, while a stream-to-memory-mapped (S2MM) transfer writes the encrypted 32-bit stream back to memory.
3) The High level Synthesis (HLS) stream synchronization logic & XOR engine shown in Fig. 2 physically separates the key generation logic from the payload encryption. It synchronizes 32 bit payload data chunk from AXI DMA with 32 bit key from key generation module and gives XOR output of the two.

The hardware accelerator pipeline is triplicated in design for distributing the workload of incoming packets at software dispatcher on the Processing system shown in Fig. 3.

To reduce per-packet DMA transaction overhead, the software layer aggregates incoming fixed-size UDP video payloads (max_payload_size: 1316 bytes) into a contiguous batch buffer in PS DDR4 memory. Configured via AXI-Lite, the DMA initiates a burst Memory-Mapped to Stream (MM2S) transfer, converting the batched memory region into a sequential AXI-Stream. This stream feeds the custom AXI-Stream Synchronization and XOR Engine, where flow control is governed strictly by TVALID/TREADY (Fig. 2) handshaking. After encryption (XOR), the Stream to Memory-Mapped (S2MM) channel writes the resulting cipher stream into a new DDR4 buffer, returning the processed batch to the software layer for network transmission.

To ensure clean transaction boundaries for the batch-oriented DMA transfers, the PS dictates the batched payload length (in 32-bit words) shown as Word_Len in Fig. 1, to both the VHDL key generator and the HLS XOR Engine via the AXI-Lite control bus. The VHDL module uses this length to assert the AXI-Stream TLAST signal upon generating the final keystream word for this batch. Operating in parallel, the HLS engine uses this length as a strict transaction counter, asserting the TLAST signal on the final data beat to close the AXI DMA's S2MM channel and prevent pipeline stalling.

To guarantee strict alignment between the payload and the keystream, the HLS synchronization engine enforces a strict lockstep barrier on both incoming AXI-Stream interfaces: the engine will not execute the XOR operation unless the TVALID signals from both the payload DMA and the VHDL key generator are simultaneously asserted. If either upstream source stalls, the HLS core automatically blocks by holding the TREADY signal low for the active stream.

The end-to-end software multiprocessing pipeline from

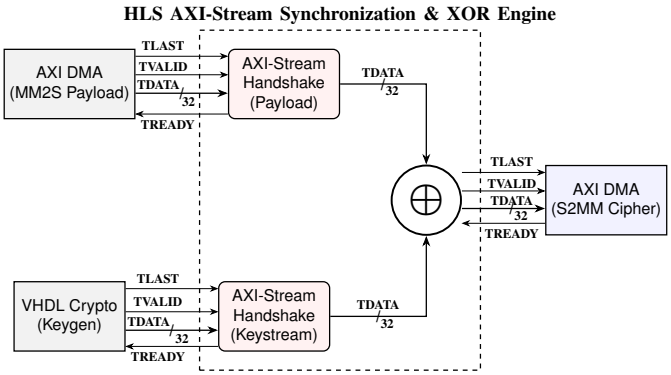

Fig. 2. Hardware dataflow architecture of the AXI-Stream Synchronization and XOR Engine, detailing the lockstep handshaking (TVALID/TREADY) and boundary assertion (TLAST) at all AXI-Stream interfaces

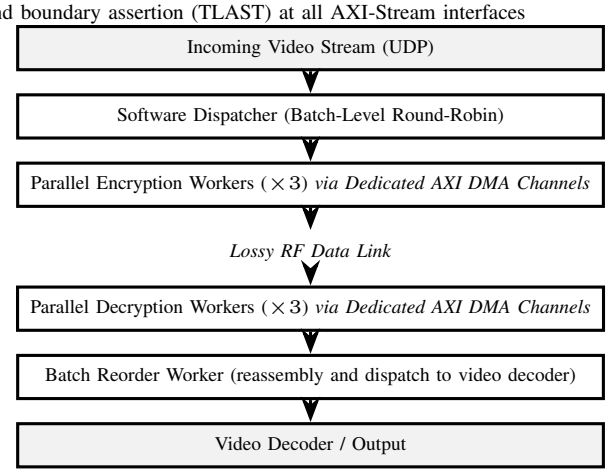

Fig. 3. End-to-end system-level multiprocessing pipeline. The software dispatcher dynamically balances batched video packets across three parallel hardware-encryption pipelines. Each worker is pinned to a dedicated ARM Cortex-A53 core, and DMA completion is detected via polling. The parallel encryption and decryption workers communicate through 3 independent, one-to-one UDP tunnels across the lossy RF link

Edge module to Ground module is shown in Fig. 3. The architecture uses a fixed-stride memory mapping paradigm[8], described below.

1) *Ingress & Batching (Sender side)*: The dispatcher aggregates incoming UDP payloads (up to 1316 bytes) into a batch of size N_{batch}. A strictly increasing *global batch identifier* (ID^{glob}_{batch}) and the accumulated batch are dispatched to encryption workers via three dedicated one-to-one queues in round robin. To enable fixed-stride memory access, the encryption worker maps each payload $i \in [0, \ldots, N_{batch} - 1]$ into an indexed 1316-byte slot within a contiguous DMA buffer of total size $N_{batch}\times 1316$ bytes, zero-padding to the nearest 32-bit boundary. Simultaneously, the exact byte length of each payload is recorded in the validity mask array, L_{batch}. The AXI DMA then transfers this entire fixed-size block to the PL for encryption.
2) *Trim & Transmit*: Post-encryption, the software utilizes the L_{batch} array for trimming the N_{batch} ciphertexts back to their original variable lengths. It prepends a custom 6-byte header to each of ciphertext payloads-containing the original payload length ($payload_len$: 2 bytes) and a strictly increasing *global sequence identifier* of 4 bytes ($seq{=}ID^{glob}_{batch}\times N_{batch}{+}i$). Packets are then sent across the independent radio link tunnels.
3) *Reception & Realignment (Receiver side)*: Upon reception, Ground node reads the custom header. It uses *global sequence identifier* to calculate the packet's original index within the batch: $slot{=}seq \bmod N_{batch}$. The software maps the variable-length payload directly into its corresponding 1316-byte slot at offset $slot\times 1316$ within the receiver's $N_{batch}\times 1316$ byte DMA buffer, and it records the payload length in receiver's local L_{batch} array. Packets dropped over the RF link leave their designated memory slots unfilled, keeping their $L_{batch}[slot]$ entry at zero.
4) *Decryption & Extraction*: The receiver initiates a single DMA transfer of the reconstructed $N_{batch}\times 1316$ byte buffer. Because the PL processes the exact same fixed-stride buffer as the sender, the stream cipher's Pseudo Random Number Generator (PRNG) state remains synchronized, advancing identically over populated data and empty slots. Finally, the software iterates through the L_{batch} array; if $L_{batch}[i] > 0$, it uses that value to extract the valid variable-length plaintext from decrypted buffer and skips any packets lost in transmission.

In Algorithm 1, which outlines the decryption worker, sequence-based synchronization is achieved using the 32-bit sequence number seq, analogous to the sequence-number based logic defined in the RTP standard [9]. The system calculates the incoming packet's batch index as $ID_{pkt}{=}\lfloor seq/N_{batch} \rfloor$ and evaluates it against the current active batch state, ID_{curr}. The variable ID_{curr} is initialized to an out-of-bounds value (-1) to allow the first received packet to safely establish the active batch context without triggering a premature flush. Subsequently, if a delayed RF packet arrives ($ID_{pkt}{<}ID_{curr}$), it is immediately discarded to prevent corrupting the active decryption window. Further, if an incoming packet's identifier exceeds the current active batch ($ID_{pkt}{>}ID_{curr}$) or a socket timeout occurs, the system forces a batch DMA decryption on the partially-filled buffer (where any entry of $L_{batch} > 0$). Algorithm 2 (Batch reorder worker) uses a *global batch identifier*-keyed hash map $B_{reorder}$, to re-serialize out-of-order batches from Q_{sys} (four-process shared queue). The worker dequeues Q_{sys} using a 10ms timeout. This ensures the worker periodically evaluates $B_{reorder}$ depth even during network silence. If a batch matching the expected identifier ID_{exp} is present in $B_{reorder}$, a PopAndForward operation on $B_{reorder}$ is executed to ensure linear delivery to video decoder. If $|B_{reorder}|{>}D_{max}$ due to deep fade RF loss, the system resets ID_{exp} to the minimum available key for recovery.

The specific protocols for Key and Initialization Vector (IV) exchange are considered outside the scope of this paper. However, it is important to note that the proposed framework is fully provisioned to support dynamic Key and IV updates along with reset of keygen module (through AXI-Lite) for every discrete batch of packets dispatched to the AXI DMA.

Algorithm 1 Decryption Worker (seq-based synchronization)

Require: Encrypted UDP stream, N_{batch}, $W_{slot} = \lceil 1316/4 \rceil$
Ensure: Decrypted batches pushed to shared queue Q_{sys}

$ID_{curr} \leftarrow -1$
$L_{batch}[0, \ldots, N_{batch} - 1] \leftarrow 0$
while socket active **do**
  if socket timeout **then**
    if any $L_{batch}[i] > 0$ **then**
      Execute batch DMA decryption
      Push $(ID_{curr}, \text{valid plaintext pkts})$ to Q_{sys}
      Reset L_{batch}
    end if
    continue
  end if
  Receive pkt and parse $(seq, payload_len)$
  $ID_{pkt} \leftarrow \lfloor seq/N_{batch} \rfloor$; $slot \leftarrow seq \bmod N_{batch}$;
  if $ID_{pkt} < ID_{curr}$ **then**
    continue ▷ Discard delayed pkt
  else if $ID_{pkt} > ID_{curr}$ **then**
    if any $L_{batch}[i] > 0$ **then**
      Execute batch DMA decryption
      Push$(ID_{curr}, \text{valid plaintext pkts})$ to Q_{sys}
      Reset L_{batch}
    end if
    $ID_{curr} \leftarrow ID_{pkt}$
  end if
  Zero-pad payload to 32-bit alignment
  Store payload at $slot \times W_{slot}$
  $L_{batch}[slot] \leftarrow payload_len$
  if $slot = N_{batch} - 1$ **then**
    Execute batch DMA decryption
    Push $(ID_{curr}, \text{valid plaintext pkts})$ to Q_{sys}
    Reset L_{batch}; $ID_{curr} \leftarrow ID_{curr} + 1$;
  end if
end while

IV. Test Setup and Results

In Fig. 4, we show the test setup used to evaluate our design. At the edge node, 1080p(HD) video is captured at 30 fps and encoded on a NVIDIA® Jetson AGX Orin using an FFmpeg-based H.264 software encoder (libx264, High profile). To minimize latency and prevent temporal error cascading during RF outages, the encoder is configured with zero-latency tuning and a strictly All-Intra frame structure (no B or P-frames). The output is enforced to a 70 Mbps constant bitrate (CBR) and encapsulated as MPEG-TS over UDP. The maximum UDP payload size is selected as 1316 bytes (<1500 MTU). The AGX Orin is connected to the ALDEC TYSOM-3 ZU7EV edge encryptor board through an Edge L2 Switch (LAN) as shown in Fig. 4. Although the TYSOM board features dual Ethernet RJ45 interfaces, the prebuilt PYNQ v3.0.1 image [10] restricts access to a single physical RJ45 port. The edge encryptor board multiplexes two logical network interfaces (eth0 and eth0:1) onto this single active RJ45 port. The ground decryptor TYSOM board has identical configuration. The IP

Algorithm 2 Batch Reorder Worker

Require: Decrypted batches from shared Q_{sys}, D_{max}=6
Ensure: Ordered plaintext video stream to decoder

$ID_{exp} \leftarrow -1$; $B_{reorder} \leftarrow \emptyset$ ▷ hash map
while system active **do**
  $data \leftarrow$ PopWithTimeout$(Q_{sys}, 10\text{ms})$
  if $data$ is not Null **then**
    $(ID_{batch}, \text{pkts}) \leftarrow data$
    if $ID_{exp} = -1$ **then**
      $ID_{exp} \leftarrow ID_{batch}$
    end if
    $B_{reorder}[ID_{batch}] \leftarrow$ pkts
  end if ▷ Ordered Buffer Flush shown below
  while $ID_{exp} \in B_{reorder}$ **do**
    PopAndForward $B_{reorder}[ID_{exp}]$ to VideoDec
    $ID_{exp} \leftarrow (ID_{exp} + 1) \bmod 2^{32}$
  end while ▷ Deep fade Loss Recovery shown below
  if $|B_{reorder}| > D_{max}$ **then** ▷ key cardinality check
    $ID_{exp} \leftarrow \min(\text{keys in } B_{reorder})$
  end if
end while

TABLE I
IP Address Allocation at the Edge and Ground

Board	IP address
AGX Orin - Edge plain (Red)	192.168.2.xx
TYSOM - Edge plain (Red)-eth0	192.168.2.xx
TYSOM - Edge Cipher (Black)-eth0:1	192.168.3.xx
TYSOM - Ground Cipher (Black)-eth0:1	192.168.3.xx
TYSOM - Ground plain (Red)-eth0	192.168.4.xx
Workstation - Ground plain (Red)	192.168.4.xx

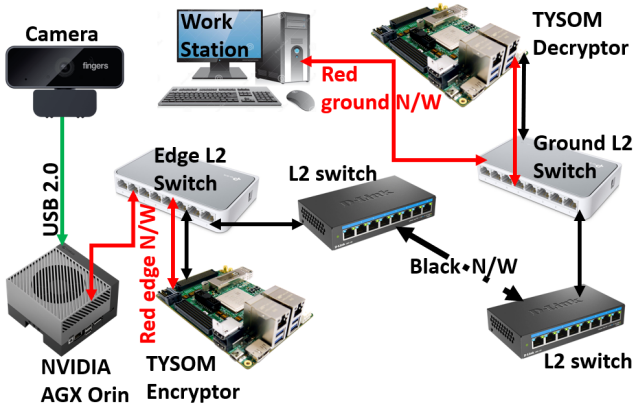

Fig. 4. End-to-end Test Setup

address table on the edge / ground side is shown in Table I.

For Mobile Ad-Hoc Network (MANET) point-to-point links, the Layer-2 switches carrying only the cipher text (black network) shown in Fig. 4 can be replaced with radios such as [11], which provide baseband interfaces for IP packet.

For throughput evaluation, we used standard HD/UHD test files from the Jellyfish video dataset [12] (all at 30 fps). These files provide consistent motion complexity for reliable benchmarking in real-time streaming evaluations. The Jellyfish video files listed in Table II, are streamed from edge for

TABLE II
JELLYFISH BITRATE TEST VIDEO FILES FOR STREAMING

Bit rate (Mbps)	File name (extension .mkv)	I frame (KB)	DMA N_{batch}
70	jellyfish-70-mbps-hd-h264	290	224
120	jellyfish-120-mbps-4k-uhd-h264	496	384
160	jellyfish-160-mbps-4k-uhd-h264	662	512
200	jellyfish-200-mbps-4k-uhd-h264	842	640

TABLE III
MAX. OBSERVED CPU UTILIZATION ON TYSOM DECRYPTOR

Core id	% CPU util 70 Mbps	% CPU util 120 Mbps	% CPU util 160 Mbps	% CPU util 200 Mbps
0	30%	50%	72%	91%
1	35%	60%	80%	100 %
2	35%	60%	80%	100 %
3	35%	60%	80%	100 %

tabulating the CPU utilization at TYSOM decryptor board in Table III. For a given CBR video stream, the Intra (I)-frame size observed at the video encoder and corresponding AXI DMA batch size (N_{batch}) configured on the TYSOM encryptor and decryptor boards are listed in Table II. The AXI DMA batch size N_{batch} is given as:

$$N_{batch} = \left\lceil \frac{R}{10^7} \right\rceil \times \left\lceil \frac{10^7 \times T_{frame}}{P_{size} \times 8} \right\rceil \quad (1)$$

where R is the target video bitrate in bits per second, T_{frame} is the temporal duration of one video frame in seconds (for 30 fps, $T_{frame} \approx 0.0333$ s), P_{size} is the maximum UDP payload size in bytes, fixed at 1316 Bytes (7 MPEG-TS of size 188 B). Based on equation 1, the architecture exhibits a proportional scaling ratio, i.e., for every 10 Mbps increase in bitrate of video stream with 30 fps, the AXI DMA batch size increases by exactly 32 packets.

To prevent user-space starvation during network bursts, CPU Core utilization is strictly capped below 60%. Based on the maximum observed CPU utilization (htop utility) shown in Table III, the operating throughput of the proposed design is established at 120 Mbps to preserve processing headroom. Cryptographic recovery under deep RF fading is evaluated using a 120 Mbps (tx_rate) encoded video stream. A software emulator deployed at each encryptor worker, triggers uniform burst losses (100–500 packets) with probability p. These independent per-tunnel emulators simulate uncorrelated fading, yet preserve the steady-state loss probabilities of an alternating renewal process akin to two-state model in [13]. The steady-state loss probability in Table IV is given by $\pi_B = (p \cdot L)/(1+p \cdot L)$, where $L = 300$ is the mean burst length. Expected goodput: $tx_rate(1 - \pi_B)$, and observed application (video decoder) goodput are shown alongside in Table IV. Table V shows the FPGA resource utilization summary of TYSOM boards at edge and ground.

V. CONCLUSION & FUTURE WORK

The proposed architecture achieves a high degree of modularity through clear functional partitioning enabling ease of integration, and adaptability to future stream cipher extensions. The proposed architecture was rigorously evaluated at data rates scaling up to 200 Mbps using both live HD camera feeds and standard Jellyfish video test files [12] with frame rate 30 fps. The system guarantees consistent operation at 120 Mbps. Experimental results demonstrate that the sequence-based synchronization and fixed-stride mapping logics implemented in software successfully maintain stream continuity even when subjected to multi-batch RF outages.

To address the inherent volatility of a MANET, future implementation could integrate a closed loop controller running on the TYSOM decryptor board which initiates command uplink to encryptor for batch scaling and FFmpeg encoder for bit rate scaling, and validate on-the-move with radios such as [11].

TABLE IV
SYSTEM THROUGHPUT FOR tx_rate=15 MBPS UNDER BURST LOSS

Fade Transition Probability (p)	Loss (π_B) probability	Expected Goodput	Obs. Application Goodput (*ffplay*)
Baseline (0%)	0%	15 MB/s	15 MB/s
0.02%	5.66%	14.15 MB/s	14.09 MB/s
0.1%	23.07%	11.53 MB/s	11.49 MB/s
0.2%	37.5%	9.37 MB/s	9.34 MB/s

TABLE V
FPGA (XCZU7EV-FFVC1156-2-E) RESOURCE UTILIZATION SUMMARY

Resource Type	Used	Available	Utilization %
CLB LUTs	21,652	230,400	9.40%
CLB Registers	15,307	460,800	3.32%
Block RAM Tile	15	312	4.81%
DSP48E2	192	1,728	11.11%

REFERENCES

[1] M. Boesgaard, M. Vesterager, and E. Zenner, "Rabbit stream cipher algorithm." RFC 4503, May 2006.

[2] F. Dridi, S. El Assad, W. El Hadj Youssef, M. Machhout, and R. Lozi, "The design and FPGA-based implementation of a stream cipher based on a secure chaotic generator," *Applied Sciences*, vol. 11, no. 2, p. 625, 2021.

[3] N. Murray-Hill, L. Fontes, P. Machado, and I. K. Ihianle, "Secure video streaming using dedicated hardware," *Journal of Signal Processing Systems*, vol. 95, no. 10, pp. 1265–1275, 2023.

[4] R. A. Bertolini, F. Carloni, and D. Conficconi, "Co-designing an FPGA-accelerated encryption library with PYNQ: the Pynqrypt case study," in *IEEE EUROCON 2023-20th International Conference on Smart Technologies*, pp. 683–688, IEEE, 2023.

[5] R. A. Bertolini, F. Carloni, D. Conficconi, and M. D. Santambrogio, "Poca: A pynq offloaded cryptographic accelerator on embedded fpga-based systems," in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 194–194, IEEE, 2024.

[6] D. Liu, Y. Zhao, Z. Ma, and Q. Ding, "Intelligent target detection and encrypted transmission system based on FPGA," *Computers and Electrical Engineering*, vol. 128, p. 110653, 2025.

[7] Aldec, Inc., *TySOM-3-ZU7EV Embedded Prototyping Board - Zynq UltraScale+ MPSoC Boards*. Aldec, Inc., 2017.

[8] L. Rizzo, "netmap: a novel framework for fast packet I/O," in *21st USENIX Security Symposium (USENIX Security 12)*, pp. 101–112, 2012.

[9] R. Frederick, V. Jacobson, and P. Design, "RTP: A transport protocol for real-time applications," *IETF RFC3550*, 2003.

[10] Advanced Micro Devices, Inc., "PYNQ supported boards and PYNQ pre-built images." [Online]. Available: pynq.io/boards.html, 2024.

[11] Silvus Technologies, "StreamCaster 4200 Enhanced Plus (SC4200EP) Datasheet," datasheet, Silvus Technologies, 2026. 2x2 MIMO MANET Radio.

[12] "Jellyfish video bitrate test files." [Online]. Available: larmoire.org/jellyfish/, 2016.

[13] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.