FPGA RISC-V Softcore Processor for Network Security

Lachlan Comino, s4532119; Supervised by Matthew D’Souza

REIT4814

Project Aims, Goals and Scope

2016 saw the most damaging DDOS attack in history…

Constructed from a botnet of unsecured *IoT* devices, it achieved a throughput of 1.2Tbps and targeted *Dyn* servers, taking down a majority of the internet.

∴ **Security in IoT devices is critical.**

However, additional security protocols can strain IoT devices. Speed-up is required for critical processes.

Aim: Speed-up AES-128-CBC

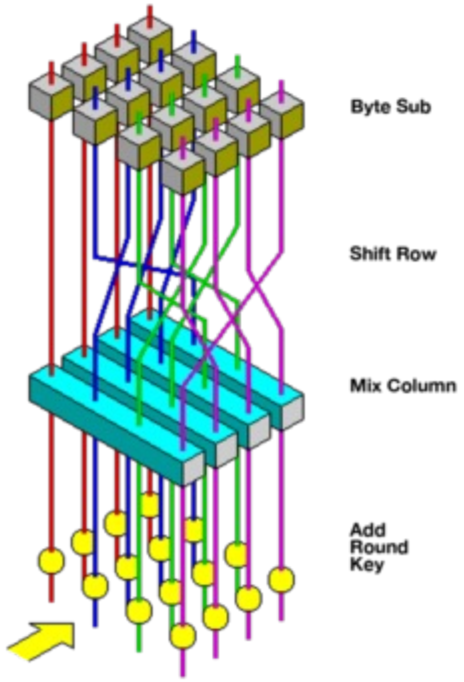


Figure 1: Diagram showing ciphering process of one AES round, Wikipedia

Algorithm Steps

1. ExpandKey
2. AddRoundKey

Loop for *n* rounds:

1. ByteSub
2. ShiftRow
3. MixColumn
4. AddRoundKey

Three ways we can **speed up** AES:

1. Precompute SBoxes in BRAM
2. Implement operations at the ISA-level
3. Combine operations into a single operation

Parallelisation is not possible (CBC relies on previous block ciphers).

Goal: Implement an Encryption SoC Server on FPGA Hardware

Using Open-Source Tools:

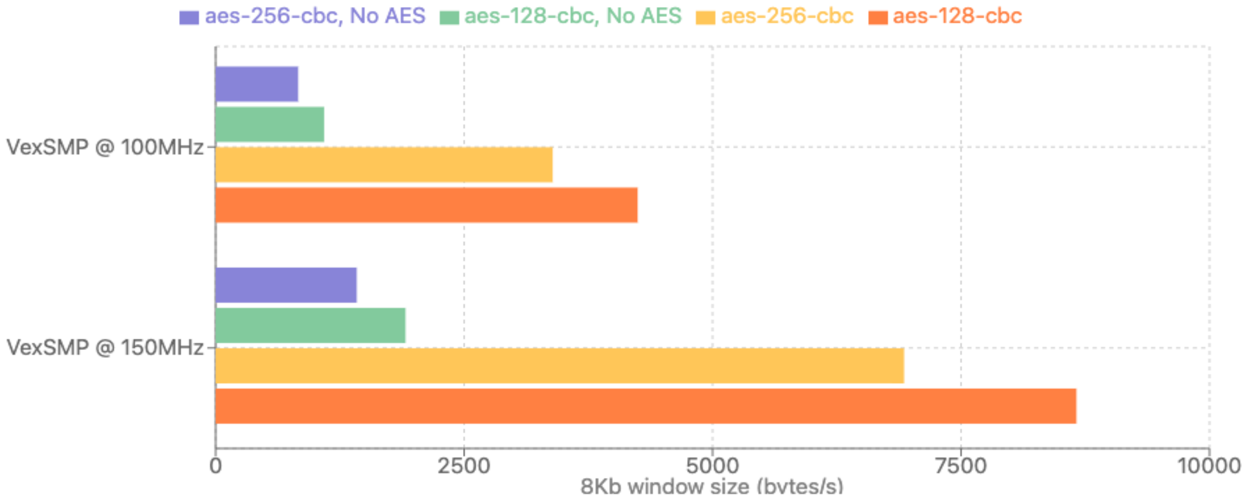
Tool	Xilinx	Open-Source
SC CPU	ARM Cortex M series	VexRiscv (RISC-V)
HDL	Verilog, VHDL	SpinalHDL
Synthesis	Vivado	F4PGA (Yosys)
SoC Building	Add IP Core in Vivado	LiteX SoC Builder
Linux Variant	PetaLinux	Buildroot Linux

School of Electrical Engineering and Computer Science

Performance Benchmarks

Encryption Rate

Figure 3: OpenSSL Speed Tests for VexRiscv Configurations (Single-core)



Ethernet Throughput

With the dual-core 100MHz VexRiscv, an average TCP throughput of **9.9Mbits/s** was achieved in *iPerf3.*

With 150Mhz dual-core, the TCP stream average increased to **22.4Mbits/s.**

Server Application, Python3

Figure 2 (Right): Diagram showing Server-Client Relationship

Implemented in full *Python3* running on Buildroot Linux.

Client streams input chunks to the server.

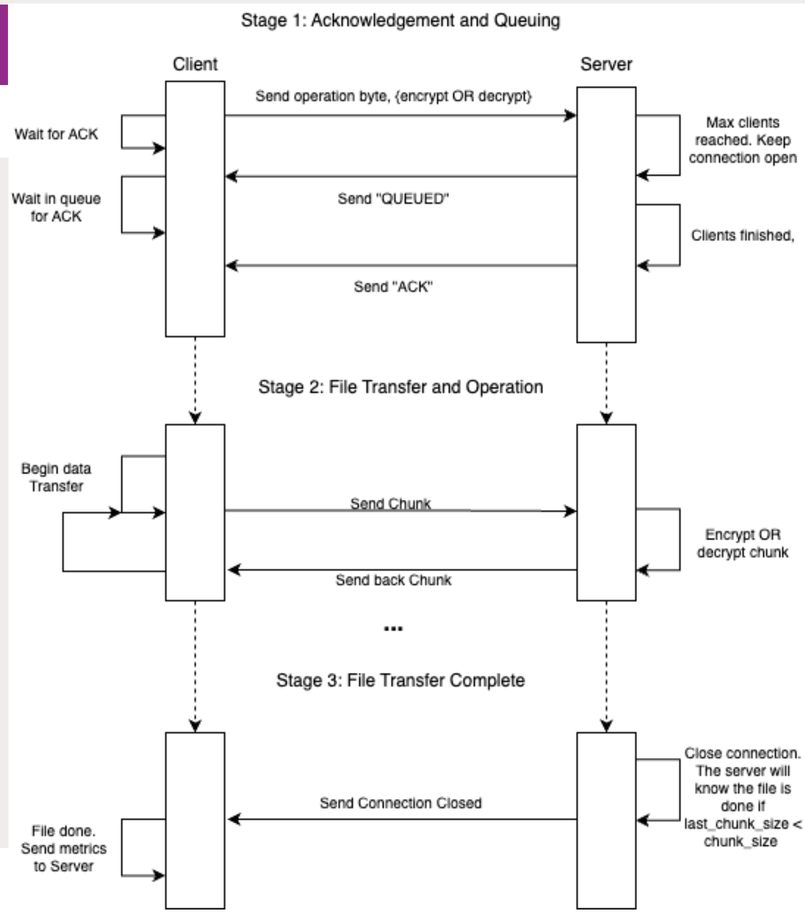
Server then encrypts or decrypts depending on what the client specified.

Server sends back the processed chunk and the transaction is closed.

This process supports the **threading** of multiple clients on the same port.

For analysing performance, “metrics” are collected both server-side and client-side.

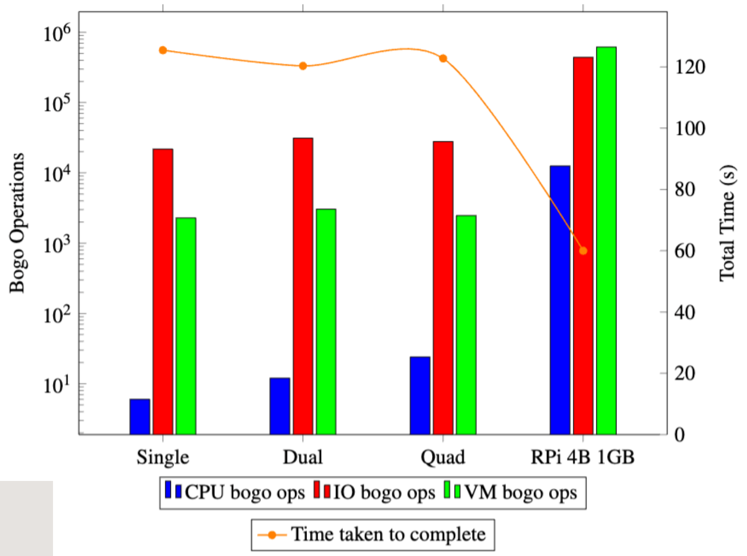
These results are then posted to a HTTP server that uses a SQLite DB.



Stage 1: Acknowledgement and Queuing
Client
Server
Send operation byte, {encrypt OR decrypt}
Wait for ACK
Max clients reached. Keep connection open
Wait in queue for ACK
Send "QUEUED"
Clients finished,
Send "ACK"
Stage 2: File Transfer and Operation
Begin data Transfer
Send Chunk
Encrypt OR decrypt chunk
Send back Chunk
...
Stage 3: File Transfer Complete
Close connection. The server will know the file is done if last_chunk_size < chunk_size
Send Connection Closed
File done. Send metrics to Server

Stress-ng Tests

Figure 4: Stress-ng Performance Comparison Across Configurations



Q: Does Overall Throughput Increase with CPU Core Amount?

A: No. Because of Resource Contention

Server Application Performance

Average throughput from each configuration, serving 100 total clients, in 10 client chunks.

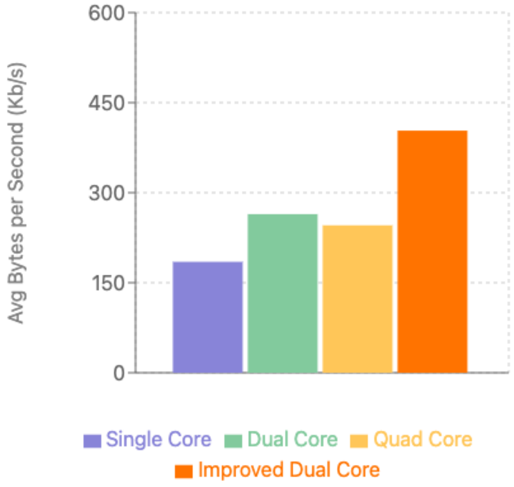


Figure 4: Server Throughputs per core amount

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

CREATE CHANGE