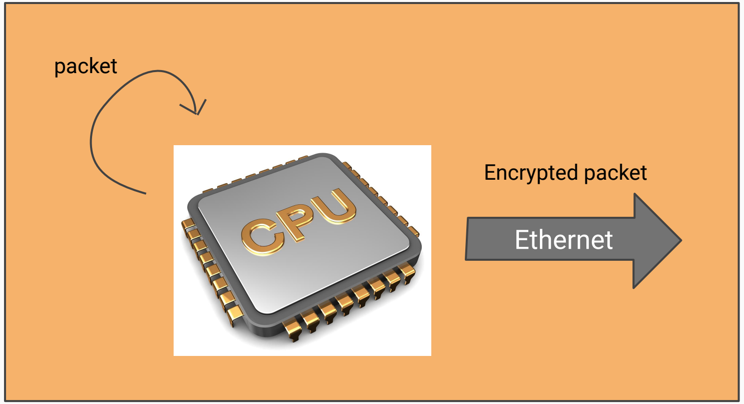
TLS-Inspired Custom Secure Network Communication System

Group Members:
Danlu Liu
Jiahui Wang
Zixuan (Brandon) Nie

Project Overview

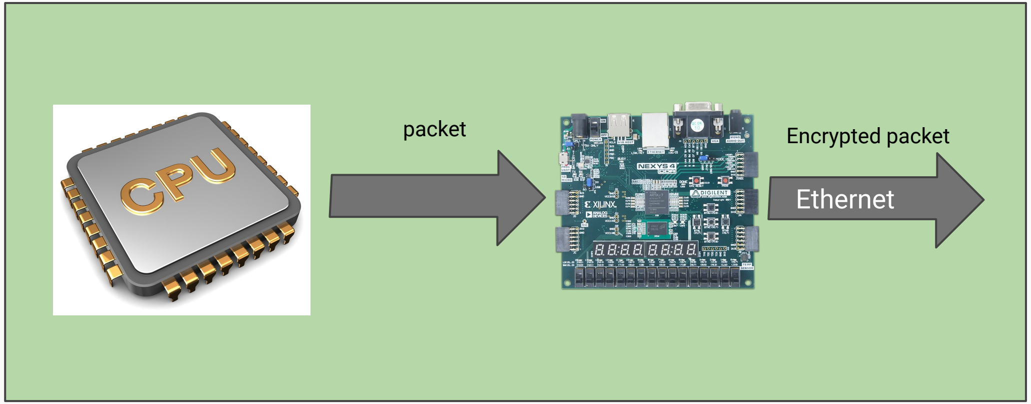
- Our idea is to build a encryption/decryption accelerator on FPGA that support RSA and AES algorithm
- Allow two parties to communicate with each other without their messages being visible to any man-in-the-middle.



packet

Encrypted packet

Ethernet



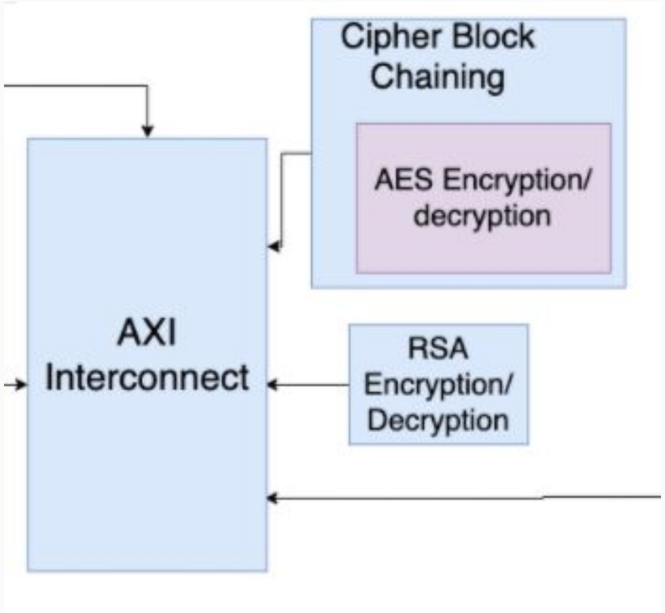
packet

Encrypted packet

Ethernet

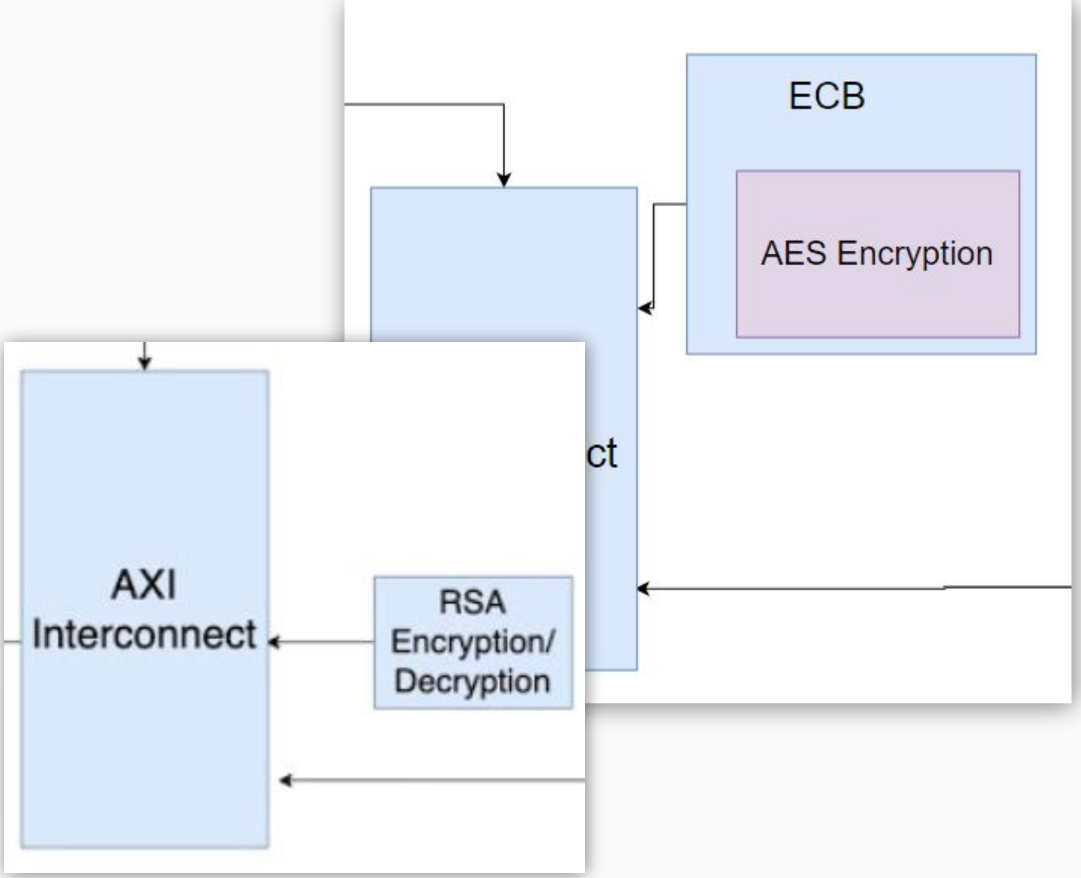
Initial Goal

- Both RSA and AES Encryption and Decryption ready
- AES and RSA block implemented in the same FPGA
- The system use RSA encryption to communicate and share the AES key, then use AES encryption to do the rest message sending



Final Result

- Two seperate system, each have one of RSA block and AES block
- The RSA block is fully functional
- The AES block is in ECB mode and can only encrypt right now



Final Result - Problem Encountered

Large width arithmetics

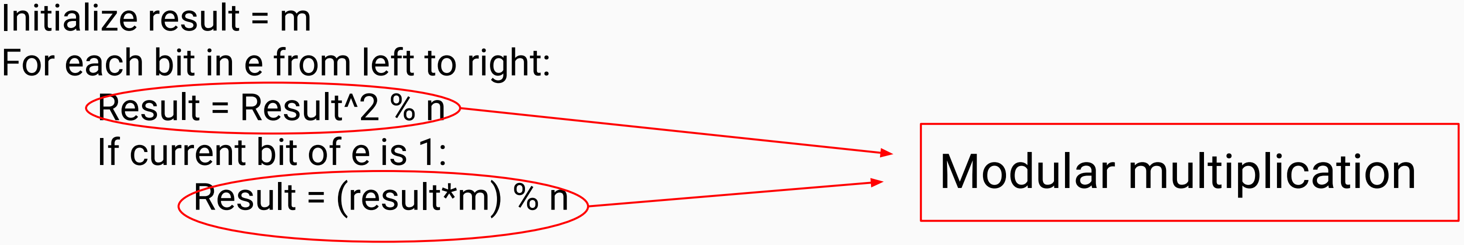
In RSA, one crucial step is modular exponentiation.

Question to solve: $m^e \% n$

It is unrealistic to do the full exponentiation before doing modulation

Final result - Problem Encountered

Solution: Square and Multiply algorithm



```
Initialize result = m
For each bit in e from left to right:
    Result = Result^2 % n
    If current bit of e is 1:
        Result = (result*m) % n
```

Modular multiplication

More problem:

- We originally planned to implement 2048 bit RSA.
- This requires 2048 bit multiplications, which is too wide for FPGA.
- This also lead to very high fanouts of signals.

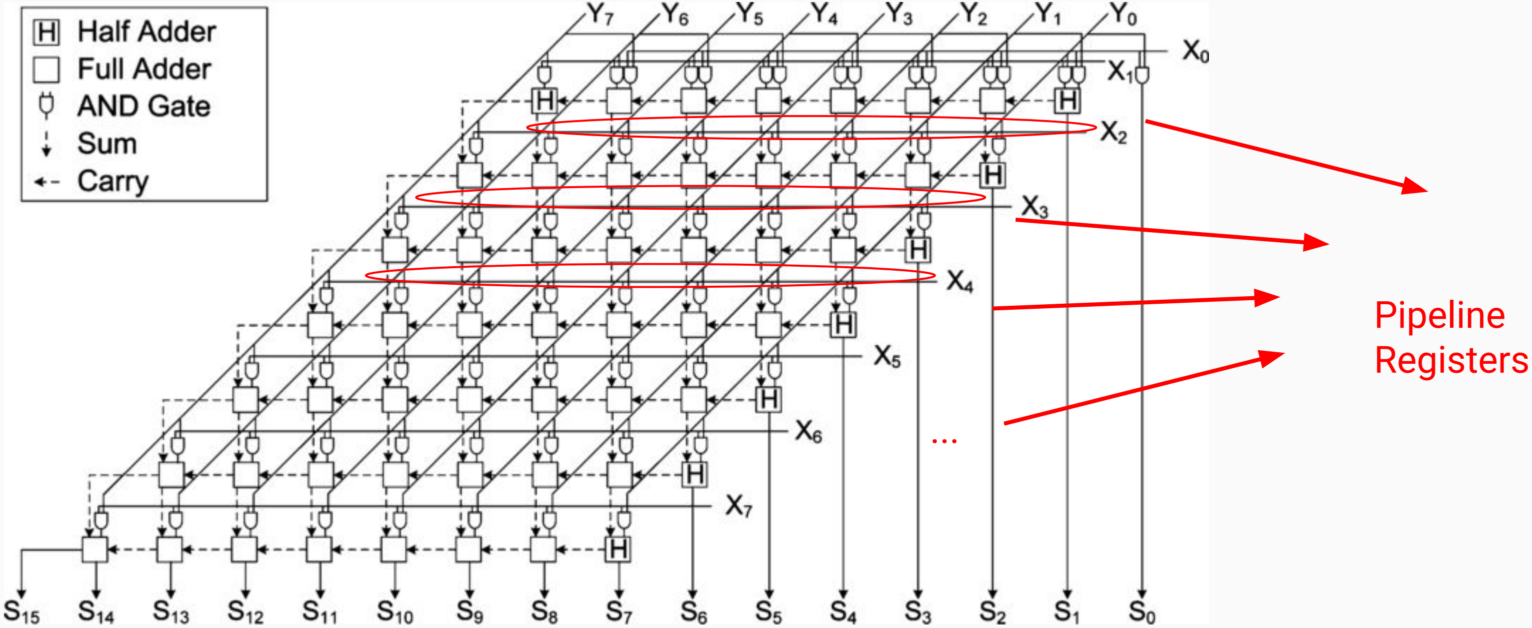
Final result - Changes Made

We changed the width of RSA to 256 bit in order to fit on the board

We tried to pipeline the operation to so we can run the RSA module at a reasonable clock speed.

Final result - Changes Made

We tried to implement unsigned multiplication using FAs to be able to add pipeline.



H Half Adder
Full Adder
AND Gate
Sum
Carry
Y_7 Y_6 Y_5 Y_4 Y_3 Y_2 Y_1 Y_0
X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7
H
S_{15} S_{14} S_{13} S_{12} S_{11} S_{10} S_9 S_8 S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0
...
Pipeline Registers

Final result - Changes Made

This requires 256x256 full adders.

- This does not fit on the board with either DSP implementation or register implementation.

We tried to use the * operator in verilog and allow Vivado to synthesize the hardware for us.

- This does work.
- However, this came at a price of huge Tmin.
- We ended up with only 10 MHz for the RSA module.

Final result - Changes Made

In the end, we changed RSA to 256 bits and managed to implement it on the FPGA

- DSP usage is still very high, at 93%.
- This means we cannot fit both RSA and AES on the same board

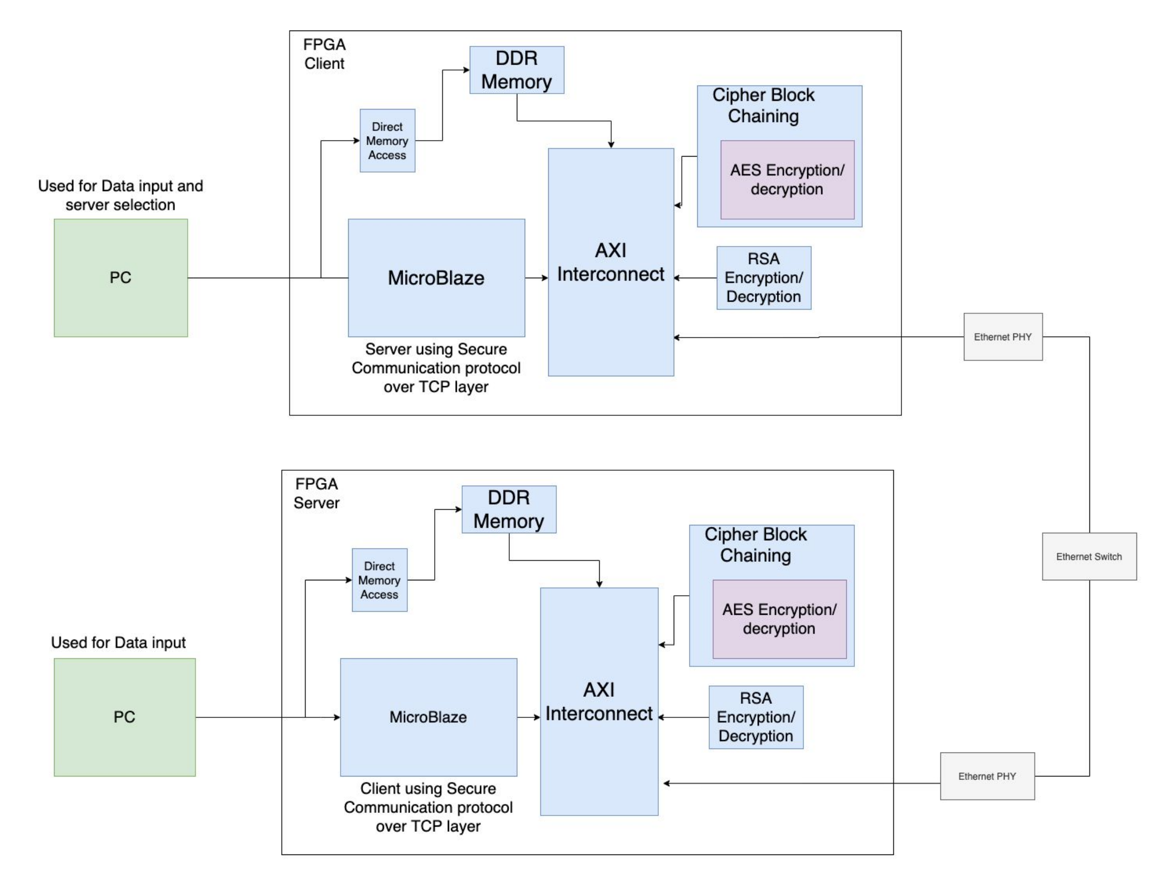
This is the reason why we chose to separate RSA and AES into two systems.

Final result - Changes Made

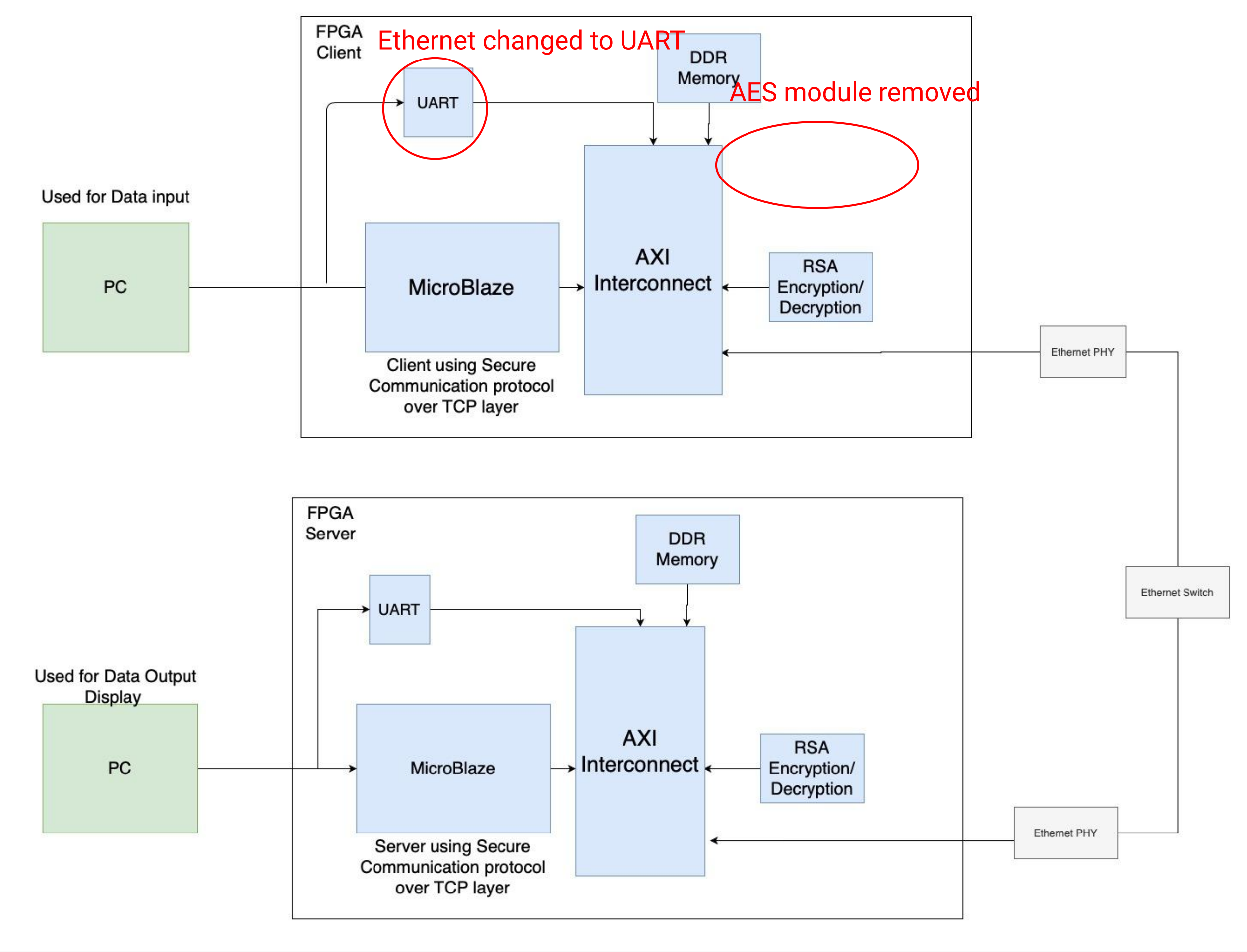
AES cryptographic module supports ECB mode with encryption only

- Due to time limitations

Proposed System


FPGA Client
DDR Memory
Direct Memory Access
Cipher Block Chaining
AES Encryption/ decryption
Used for Data input and server selection
PC
MicroBlaze
AXI Interconnect
RSA Encryption/ Decryption
Server using Secure Communication protocol over TCP layer
Ethernet PHY
Ethernet Switch
FPGA Server
DDR Memory
Direct Memory Access
Cipher Block Chaining
AES Encryption/ decryption
Used for Data input
PC
MicroBlaze
AXI Interconnect
RSA Encryption/ Decryption
Client using Secure Communication protocol over TCP layer
Ethernet PHY

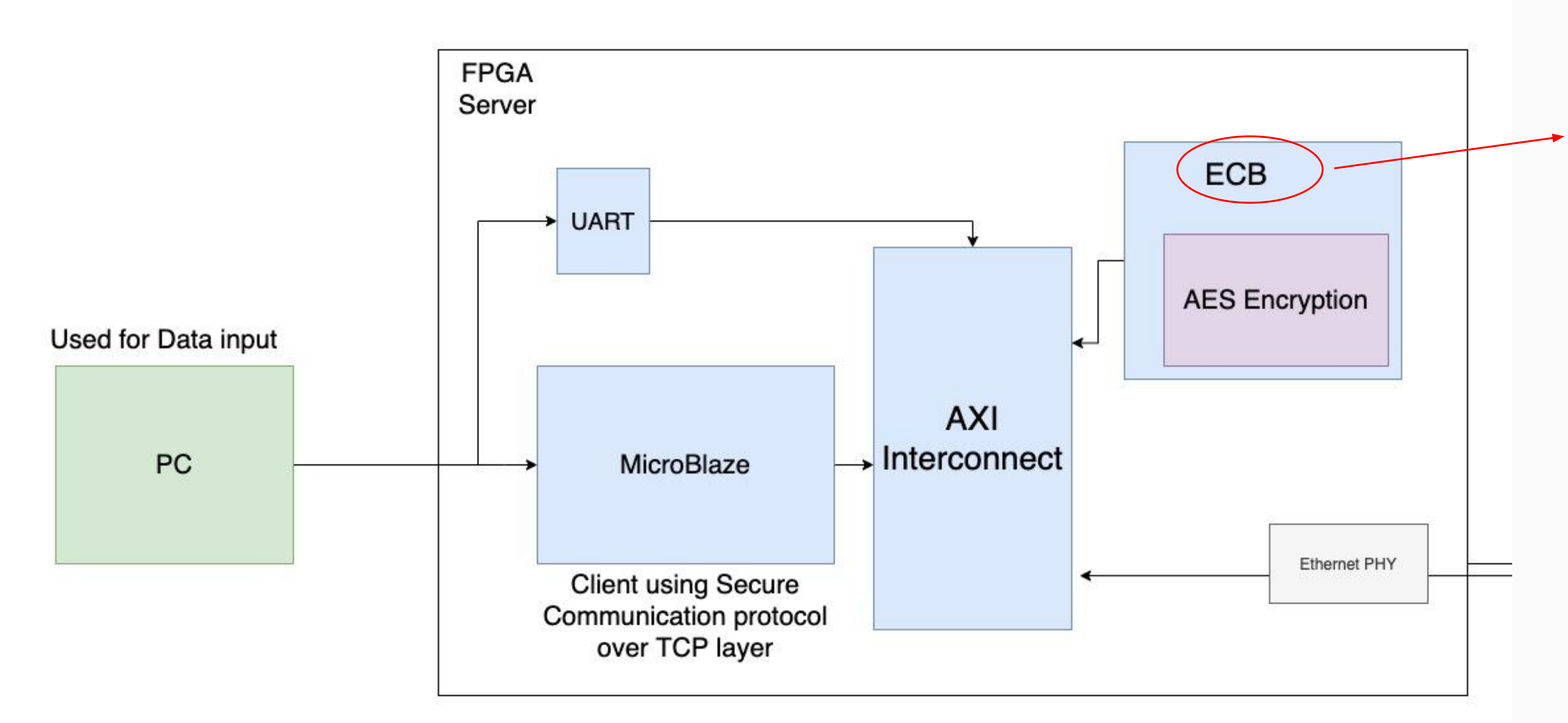
Final Result



- Ethernet is replaced by UART.

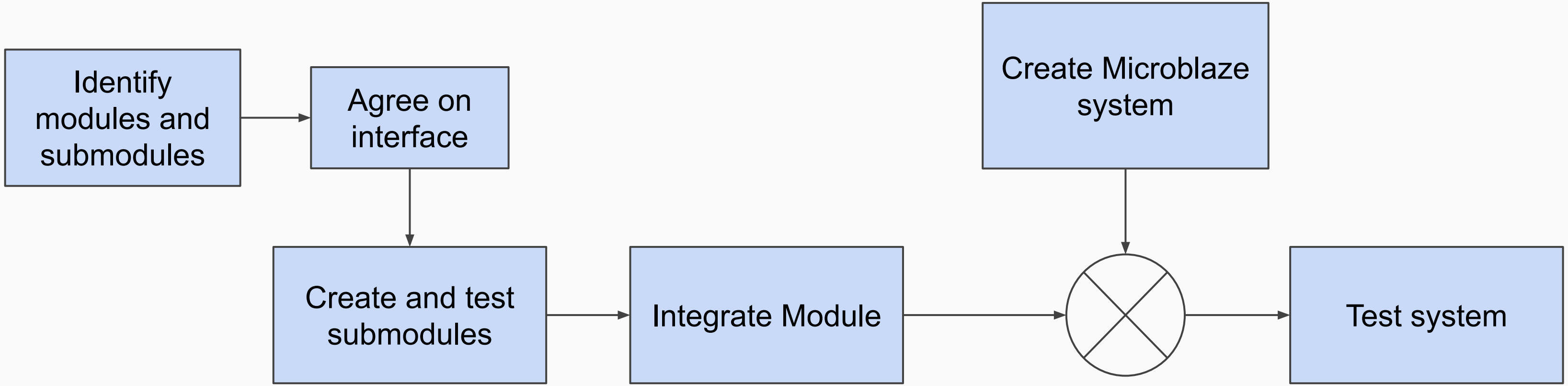
- RSA and AES module will be instantiated in two separate systems.

Final Result



No longer uses CFB mode

Design Process



Our Own Blocks

- All cryptography blocks are created by the team
- Referenced papers and standard

Other Blocks

- Microblaze system blocks use existing blocks in Vivado library

What We Learned

Project Management

- Need to better estimate complexity of modules
- Sunk cost fallacy
- Parallel work flow
- Start integration earlier

Project Implementation

- Testing in software first and/or seeking out verified examples
- Should start with easier modules first
- Standards can be confusing to interpret - look for examples and other diagrams
- IEEE papers have varying writing quality

Demo and Questions