OPEN
MAINFRAME
PROJECT

- **Name:** Sladyn Nunes
- **GitHub username:** @sladyn98
- **Email:** sladynnunes98@gmail.com
- **Location:** Mumbai India
- **Time Zone:** UTC+05:30

Boring SSL Project Plan

What is Boring SSL?

BoringSSL is a fork of OpenSSL that is designed to meet Google's needs. ...**BoringSSL** arose because Google used OpenSSL for many years in various ways and, over time, built up a large number of patches that were maintained while tracking upstream OpenSSL.

Problem Definition:

- We will use the hardware acceleration and associated libraries (libica).
- The project will provide a generic hardware acceleration interface such that other architectures can plug into it.
- If hardware isn't available it default to the existing little-endian code
- For Z there are two libraries which provide access to the hardware and if it's not available will implement things in software
- This project will require you to research and understand the hardware acceleration libraries and how they are invoked
- Take the latest BoringSSL code and enable it to invoke a generic layer of h/w acceleration routines
- For Z, have that generic layer invoke the APIs in the libraries

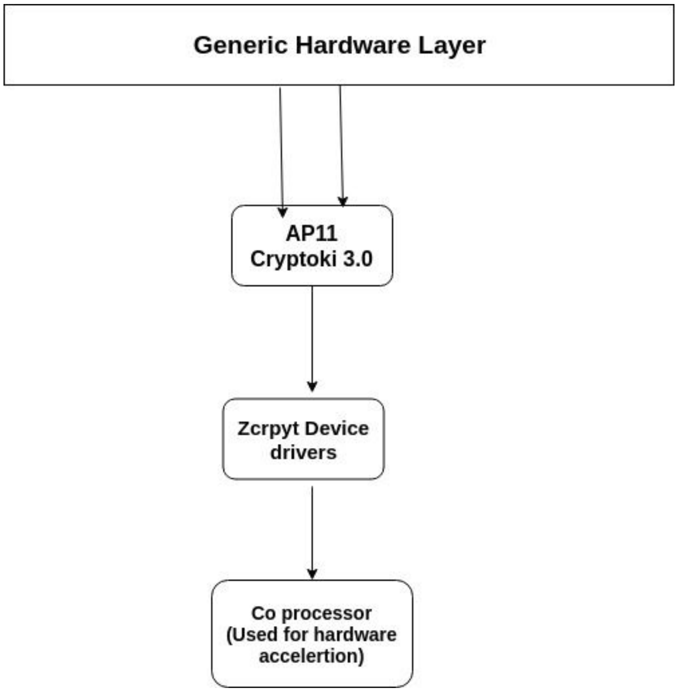
Set of Deliverables:

a) Deliverable : The project will provide a generic hardware acceleration interface such that other architectures can plug into it.

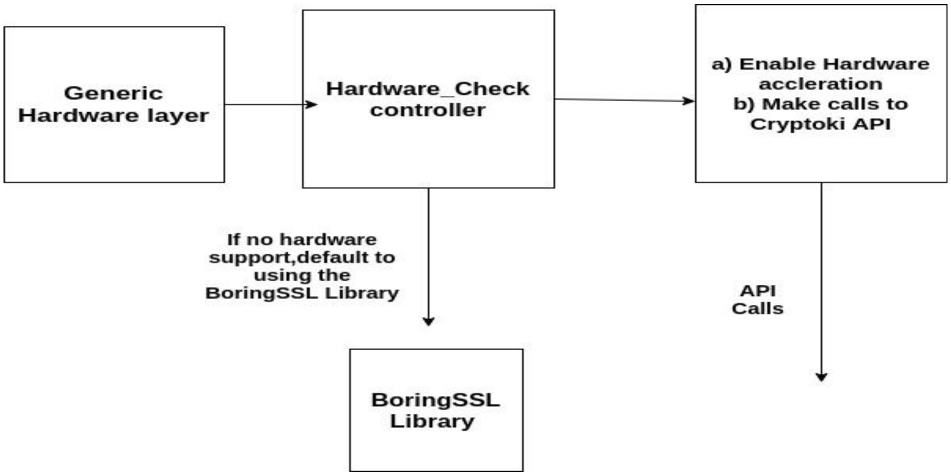
b) Deliverable : Take the latest BoringSSL code and enable it to invoke a generic layer of h/w acceleration routines.

Concept Diagram:

Current Hardware Acceleration Systems



After Implementaion of Hardware Layer:



Community Bonding Period (June 10th - June28th):

- **Get familiar with BoringSSL and s390x systems.**
- Prepare a rough design for the hardware layer and get understand the Boring SSl code.
- Learning more about the hardware acceleration libraries and how they are invoked in linux Z.

CODING PHASE

Coding Phase (July 1st - Aug 15th):

Week	Tasks	Goals
Week 1-2 (July 1th - July 15th)	● Preparing the design of the hardware layer ● Begin implementation of the above generic layer	● Complete design and a partly implemented layer.
Week 3-4 (July 16 th - August 1st)	● Complete the entire layer	● Complete and deployable layer of the hardware layer
Week 5 -6 (August 1st - August 15th)	● Writing tests for the above design layer to make sure the layer works as desired with maximum test coverage. ● Start design for the accommodation of layers into this architecture	● Complete test architecture for the design layer ● Design for multi-architecture interface.

Coding Phase II(August 16th - September 9th)

Week	Tasks	Goals
Week 7 -8 (August 16 - September 2)	● Implementation of hardware acceleration invocation through boring ssl code.(part 1)	● Completion of Test suites for all builds
Week 8 - 12 **(**September 2- 9)	● Complete the complete linking of the boring ssl code with the hardware acceleration hardware. ● Complete testing of the above integration **(part 2)**	● Complete integration of the boring ssl code and the hardware libraries. ● Complete testing architecture for the same

Rough Code Modules:

Code Module 1 :

```
Hardware_Check_Controller
 Check_hardware_acc_availability( ) :
 #Checking whether the hardware acceleration is enabled or whether it is possible to access the hardware acceleration

 Swith_to_software_module() :
# Function to switch to software module if non availability of hardware acceleration
```

```
Swith_to_hardware_layer() :
# Function to switch to the generic hardware layer.
```

Code Module 2:

```
Class API_Invocation:
 private void sha256_cryptoki ( ) :
 # Implementation of the crypto functions via the standard libraries.This
would make an internal call to the cryptoki API.

   private void crypto2 ( ) :
# Sample function
 This module will exist separately since the generic hardware layer will
take care of calling and invoking the crypto libraries.And it would benefit
being a separate module  since it would be a part of a bigger interface
where architectures can be plugged into.
```