#!/bin/bash

echo "Content-type:application/json"

echo

INPUT=${QUERY\_STRING}

echo "Random bytes input: $INPUT"

if [$INPUT == ""]; then

RN\_BYTE=8

else

RN\_BYTE=$INPUT

fi

get\_puf\_vrf() {

# Execute commands

PUFS\_UTIL\_PATH=/home/root/projects/pufs\_util

PID=$$

OUT\_FILE=get\_random\_$PID.txt

TMP\_FILE=tmp\_$PID.txt # ???

$PUFS\_UTIL\_PATH/pufs\_utility -R 2 -l $RN\_BYTE -a ECDSAP256 -O $OUT\_FILE > $TMP\_FILE

rm $TMP\_FILE # ???

cat $OUT\_FILE

rm $OUT\_FILE

sleep 2 # this is to simulate get\_puf\_vrf processing time

# Remove lock file

rm "$lockfile"

}

# Set path for lock file

lockfile=puf-fpga.lock

# Check if lock file exists

if [ -f "$lockfile" ]; then

# If lock file exists, puf-fpga is locked for in use

echo "locked-in-use"

else

# If lock file does not exist, create it

touch "$lockfile"

# Execute function

{ time get\_puf\_vrf 2; } 2>> get\_puf\_vrf\_execution\_time.txt

fi

This is a bash script that uses a lock file to ensure that only one instance of the script is running at any given time. When the script starts, it checks for the existence of the lock file and exits if it is already present. If the script is done running, it removes the lock file. This way, if the script is called again while it is still running, it will exit immediately and not create a new process.

The script starts with an echo statement that outputs "Content-type:application/json". It then reads the QUERY\_STRING environment variable, which contains the input parameters passed to the script. If the input is an empty string, the script sets the RN\_BYTE variable to 8; otherwise, it sets RN\_BYTE to the input value.

The get\_puf\_vrf function executes a set of commands that generate a random string of bytes using the pufs\_utility program and stores it in the OUT\_FILE file. It then reads the content of OUT\_FILE, removes OUT\_FILE and a temporary file named TMP\_FILE, and sleeps for 2 seconds to simulate processing time. Finally, it removes the lock file.

The lock file is created with the path "/puf-fpga.lock". If the lock file exists, the script outputs "locked-in-use" and exits. Otherwise, it creates the lock file, executes the get\_puf\_vrf function, and appends the execution time to the get\_puf\_vrf\_execution\_time.txt file.

The script could be improved by adding error handling and more detailed comments. It also assumes that the pufs\_utility program is installed in the specified path, and that the user running the script has permission to create and remove files in the file system.

Create a shell script httpd-puffpga.sh under /etc/init.d

# jshuo@raspberrypi:~ $ sudo cp httpd-puffpga.sh /etc/init.d

# jshuo@raspberrypi:~ $ sudo ln -s /etc/init.d/httpd-puffpga.sh /etc/rc5.d/S99httpd-puffpga.sh

# sudo unlink -s /etc/init.d/httpd-puffpga.sh /etc/rc3.d/S99httpd-puffpga.sh

# sudo unlink -s /etc/init.d/httpd-puffpga.sh /etc/rc5.d/S99httpd-puffpga.sh

# check /var/log/boot.log

rm -rf /home/jshuo/projects/cgi-bin/puf-fpga.lock