How to use the vagrantfile to spin up our vagrant box:

In oder to use a vagrant file you need to have vagrant and virtualbox downloaded and installed. This link contains information on both of these: https://learn.hashicorp.com/tutorials/vagrant/getting-started-index?in=vagrant/getting-started

The parts of interest are the links to donwload

- **vagrant: https://www.vagrantup.com/docs/installation**
- **And virtualbox: https://www.virtualbox.org/**

***Note the following instructions are based on our experience running vargrant and virtual box on a Ubuntu 20.04 host machine. For Windows or Mac hosts you may need to do some digging in the vagrant docs https://www.vagrantup.com/docs/index for troubleshooting.**

***Note in the instructions below we use curly brackets {} to denote a variable part of a file name. The {} is replaced with either 89,107,127 where appropriate. These represent the field sizes of the arithmetic circuits that we use.**

Instructions:

Download the vagrantfile and put it into the directory you would like to access the VM from.

Execute: vagrant up

In the same terminal that you ran vagrant up (terminal 1) run vagrant ssh. This will ssh you into the running VM. You can now cd into pFairSwap/experiments directory on the VM.

Experiment 1: Measure the time in seconds to execute Circuit Required:

Required Files:

- Circuit files: circuits/uniHash-{}.arith

- Randomized Circuit files: circuits/expanded-uniHash-{}.arith

- Circuit Input: inputs/inputs-{}

- Randomized Circuit Input: inputs/encoded-inputs-{}

++

Edit Files: *Note that these scripts have these values as default so you can skip editing these files if you run python2 execute-circuits-127.py first

~/msr-vc/pinocchio/ccompiler/src/aritheval.py

- Line 25 return(pow(input*self.const,1,(2**127)-1))

~/msr-vc/pinocchio/ccompiler/src/BaseEval.py

- Line 45 return(pow(result,1,(2**127)-1))

Execute:

- $ python2 execute-circuits-127.py

++

Edit Files: This experiment executes the arithmetic circuits using the pinocchio suit inside the msr-vc directory. To execute each python script we first need to edit some files in the pinocchio suit. Edit the files as below:

~/msr-vc/pinocchio/ccompiler/src/aritheval.py

- Line 25 return(pow(input*self.const,1,(2**89)-1))

~/msr-vc/pinocchio/ccompiler/src/BaseEval.py

- Line 45 return(pow(result,1,(2**89)-1))

Execute:

- $ python2 execute-circuits-89.py

++

Edit Files:

~/msr-vc/pinocchio/ccompiler/src/aritheval.py

- Line 25 return(pow(input*self.const,1,(2**107)-1))

~/msr-vc/pinocchio/ccompiler/src/BaseEval.py

- Line 45 return(pow(result,1,(2**107)-1))

Execute:

- $ python2 execute-circuits-107.py

++

Process:

The python programs execute-circuits-{}.py execute the original and randomized circuits for each circuit size. the circuits are executed by the pinocchio suit's aritheval.py program. Each circuit is executed several times and the average time is written to a file.

Each time the circuit is executed, two files are produces. One that contains the wire execution trace, this file shows the value carried on all the wires, and an output file which shows the values of the output wires only. We only use the wires file, not the output file.

Timing information is stored in timing-execute-{}

Experiment 2: Measure the time in seconds to Encrypt each wire values

Required Files:

- Wire files: wires/wires-{}
- Randomized Wire Files: wires/expanded-wires-{}

Execute:

- $ pyenv activate web3 (if not already active)
- $ python3 encode-wires.py

Process:

The python programs encode-wires.py reads each wires file (original and randomized) and runs them through the encodeWire function from the Protocol->Encode file with the wire values and a randomly generated key

encode-wires.py does this several times in a loop. The number of loop iterations its set by variable loopCount on line 4 in each pythong script. You can change this value and re-run the experiments. Timing information is recording in files called timing-execute-{} that are located in the timing-info directory.

Experiment 3: Generate PoM and send it to the smart contract

For this experiment we will need to edit the code again, details are described later below.

First open another terminal (terminal 2) in the same directory as the Vagrantfile on the host machine. Now run vagrant ssh from terminal 2, this connects you to the VM through again, and cd into pFairSwap/experiments directory. Now execute:

- $ truffle develop

This brings you to the truffle development environment, now execute the command:

- migrate

This compiles and deploys the Jude.sol contract onto the local blockchain

Back in terminal 1, execute the following command:

- $ python3 generate-pom.py

A python script will run and communicate a PoM to the deployed blockchain. The python script prints to standard output the gas cost to store and merkle proofs and to execute the judge function.

This default setting runs the experiment for the uniHash-89.arith circuit. To experiment can be executed for the other circuits (uniHash-107.arith, uniHash-127.arith) and the expanded version of each circuit (expanded-uniHash-89.arith, expanded-uniHash-107.arith, expanded-uniHash-127.arith)

We run each experiment we need to make some changes to the files in pFairSwap and to files in msr-vc. below is summary of all changes for each experiement: After you make each edit to the Jude.sol you will need to re-deploy the blockchain and smart contract in terminal 2. Reminders on this are included below.

An example output is shown further down below

*Note the corruptIdx variable is the index of the “corrupted gate” we will complain about. The only requirement for this is that it is a multiplication gate. You can take a look at the circuit files in the circuits sub directory to see the entire circuit and how multiplication gates are represented there.

UniHash-89.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 6781
- Line 13 wireFile = “wires/wires-89”
- Line 14 cirFile = “circuits/uniHash-89.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 15;
- Line 11 uint256 Mp = 89;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- migrate

Expanded-UniHash-89.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 18241
- Line 13 wireFile = “wires/expanded-wires-89”
- Line 14 cirFile = “circuits/expanded-uniHash-89.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 18;
- Line 11 uint256 Mp = 89;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- migrate

UniHash-107.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 6014
- Line 13 wireFile = “wires/wires-107”
- Line 14 cirFile = “circuits/uniHash-107.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 14;
- Line 11 uint256 Mp = 107;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- migrate

Expanded-UniHash-107.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 15508
- Line 13 wireFile = “wires/expanded-wires-107”
- Line 14 cirFile = “circuits/expanded-uniHash-107.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 18;
- Line 11 uint256 Mp = 107;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- migrate

UniHash-127.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 4484
- Line 13 wireFile = “wires/wires-127”
- Line 14 cirFile = “circuits/uniHash-127.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 14;
- Line 11 uint256 Mp = 127;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- migrate

Expanded-UniHash-127.arith:

PFairSwap/experiments/generate-pom.py:

- Line 12: corruptIdx = 13576
- Line 13 wireFile = “wires/expanded-wires-127”
- Line 14 cirFile = “circuits/expanded-uniHash-127.arith”

PFairSwap/experiments/contracts/Judge.sol

- Line 9 unit256 constant depth = 18;
- Line 11 uint256 Mp = 127;

Start Blockchain, deploy smart contract execute commands below in terminal 2:

- $ truffle develop
- Migrate

APPENDIX

Details on constructing the vagrant box (This information is not needed to use to box, just information on how the environment was set up if curious).

New Box from base image ubuntu-20.04.3-desktop-amd64.iso in VirtualBox downloaded from (https://ubuntu.com/download/desktop)

User: vagrant

Password: vagrant

_____

Packages installed:

Pip:

- $ sudo apt install python3-pip

Vim:

- $ sudo apt install vim

Curl:

- $ sudo apt install curl

Git:

- $ sudo apt install git

Npm:

- $ sudo apt install npm

Pyenv: (https://realpython.com/python-virtual-environments-a-primer/) and (https://github.com/pyenv/pyenv-installer)

- $ curl https://pyenv.run | bash
- need to export pyenv to .bashrc (https://github.com/pyenv/pyenv-installer/issues/103)
    - $ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
    - $ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
    - $ echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n eval "$(pyenv init -)"\nfi' >> ~/.bashrc
    - $ exec “$SHELL”
- need to create a sane build environment for pyenv (https://github.com/pyenv/pyenv/wiki)

    - $ sudo apt-get update; sudo apt-get install make build-essential libssl-dev zlib1g-dev \
      libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm \
      libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
- need to install requires python versions for pyenv
    - $ pyenv install 3.8.5
- need to create virtualenvs (https://github.com/pyenv/pyenv-virtualenv)
    - $ pyenv virtualenv 3.8.5 web3
- need to activate web3 and install web3.py package,
    - $ pyenv activate web3
    - $ pip install web3

We are finished for now with web3 environment so deactivate it

- $ pyenv deactivate

Now we need to install truffle which allows us to deploy smart contracts (https://trufflesuite.com/truffle/)

- $ sudo npm install truffle –g

*some errors with permissions, but install successfully

Now we need to get two git hub repositories. We do this by downloading the zip file from each so we don't need to worry about authenticating GitHub.

- https://github.com/corda/msr-vc/tree/master/pinocchio
- https://github.com/Prezzy/verbose-octo-garbanzo

From Downloads unzip to /home/vagrant/

Rename msr-vc-master to msr-vc

Rename verbose-octo-garbanzo to pFairSwap

PFairSwap/experiments was already initalized as a truffle box

_____

Finish preparing the box to become a vagrant box following instructions from:

https://oracle-base.com/articles/vm/create-a-vagrant-base-box-virtualbox

https://www.vagrantup.com/docs/boxes/base

https://www.vagrantup.com/docs/providers/virtualbox/boxes