OVERVIEW

Cape is a compiler aided solution to the problem of side-channel attacks. The actual implementation of cape requires access to LLVM version 6.o which is currently depreciated. Hence, we have presented a simulation of cape.

We first analyse a general C program and identify the functions that have secret dependencies. These functions are stored in raw data and these functions are finalised by analyze_on_raw.py and stored in analysis_final.d. Finally we have implemented an LLVM function Pass that adds XTEST() and XEND() at the start and end of each function(the appended functions will only be appended on the functions with secret dependancies).

COMMANDS

```
clang analyze.ll -o analyze

./analyze in.c

python3 analyze_on_raw.py

////<comment> analysis on high level language

clang -S -emit-llvm in.c -o in.ll

///<comment> if the llvm pass is compiled already skip the next step

clang++ pass.cpp -fPIC -shared -o pass.so `llvm-config —cxxflags —ldflags —libs` -fno-rtti

opt -enable-new-pm=0 -load ./pass.so -f -tsx < in.ll > out.ll

clang out.ll -o out

./out
```