I created a Java program call ASM\_Similarity.java that closely follows the bit shredding technique described in the Adkins and Upchurch paper. ASM\_Similarity goes through 4 phases to complete the comparison of 2 executable files. Since this program was written up in a short amount of time, there is much further refinement to be done.

1. The use of objdump to output the file into a .asm file. The reason I currently chose objdump was because it uses AT&T syntax, where IDA Pro uses NASM/Intel syntax. AT&T syntax allows for me to more easily parse the asm file when normalizing, as registers are marked with %, constants marked with $, etc. The normalizing script could be modified to handle NASM/Intel syntax as well.
2. Is the normalization of the file. These means making the code more generic through the following modifications:
   1. Remove comments added by ObjDump
   2. Replace specific registers with “REG”
   3. Replace labels, for example "00000000004008d0 <fputs@plt>:" to "LABEL:"
   4. Replace constants with “CONST”
   5. Replace specific locations with “LOC”
   6. Replace variables with “VAR’

The following example is a piece of code before and after it has been normalized:

0000000000400860 <\_init>:

sub $0x8,%rsp

mov 0x20178d(%rip),%rax # 601ff8 <\_DYNAMIC+0x1e0>

test %rax,%rax

je 400875 <\_init+0x15>

callq 4009a0 <socket@plt+0x10>

add $0x8,%rsp

retq

LOCAL-LABEL:

sub CONST,%REG

mov VAR(%REG),%REG

test %REG,%REG

je LOC

callq LOC

add CONST,%REG

retq

1. After the files have been normalized, we shred the file into n-grams. These n-grams are then hashed into a hash table.
2. We compare the hash table. Using the Jaccard index, we return a percentage similarity between the two files.

Results are contained in the spreadsheets