## Conclusion

Results of Testing and Related Metrics

**⚠** Download Research

## Project Roadmap



#### **IDEA**

Compare encryption algorithms.



### **ENCRYPTION**

Identify encryption algorithms.



#### **OBJECTIVES**

Create encyption tests.



Performs tests for data.



### **DISCOVERIES**

Present findings & comparisons.



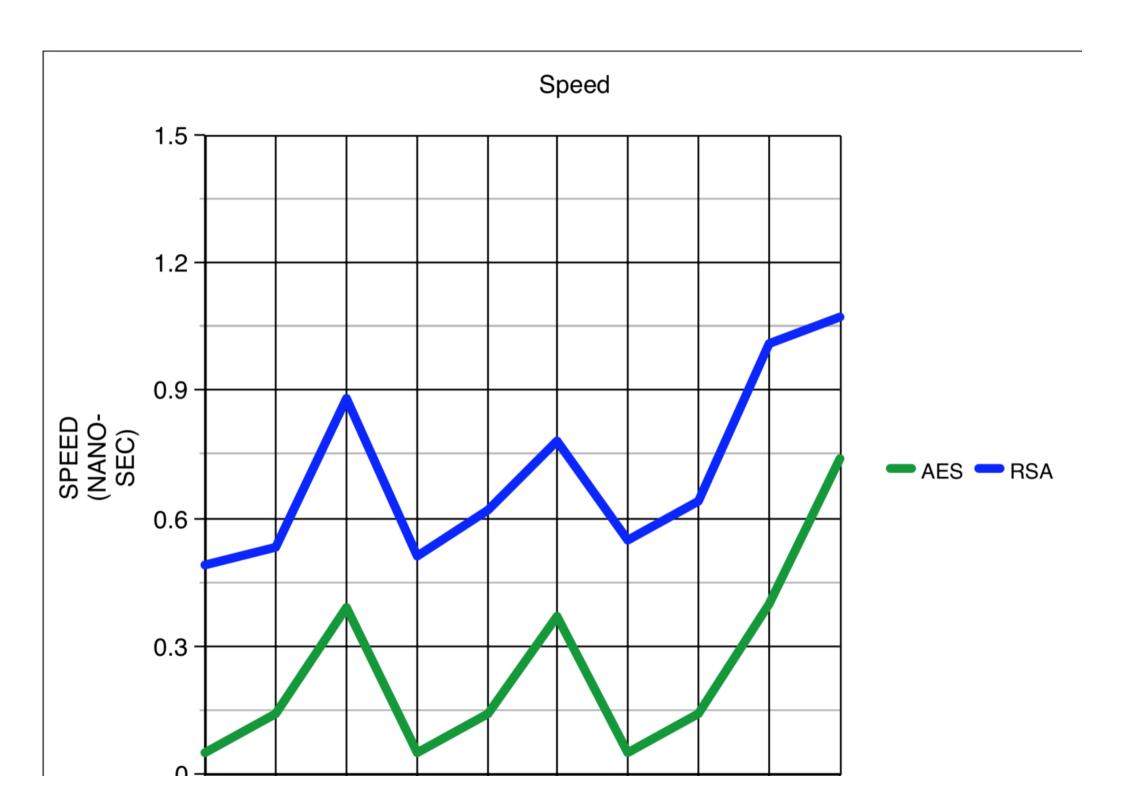
### **CONCLUSION**

Determine the best encryption.

## **Types of Comparison**

Encryption Speed and Encryption Strength.

- Encryption Speed
  - I will perform 50 tests for each encryption algorithm to determine speed efficiency. Each test takes in three files of differing sizes and encrypts the files while recording the start and end time.
- Encryption Stength
- For these same tests and results I will determin strength of the encrypted algorithm. Without a better metric, I chose to compare the size of the encrypted files created to gauge strength.



### Speed Test

After calculating the results of over 100 tests I combined similar results, usually over the same size data, to show a simple 10 test chart. As shown, the results are overwhelmingly obvious that AES is on a high order of magnitude faster than RSA Encryption. This is a result of the difference in complexity between the two. RSA has additional instruction sets and byte-chunks that slow down encryption time.

**AES Speed**: Relatively Fast: Simple implementation does lead to a significant difference in speed.

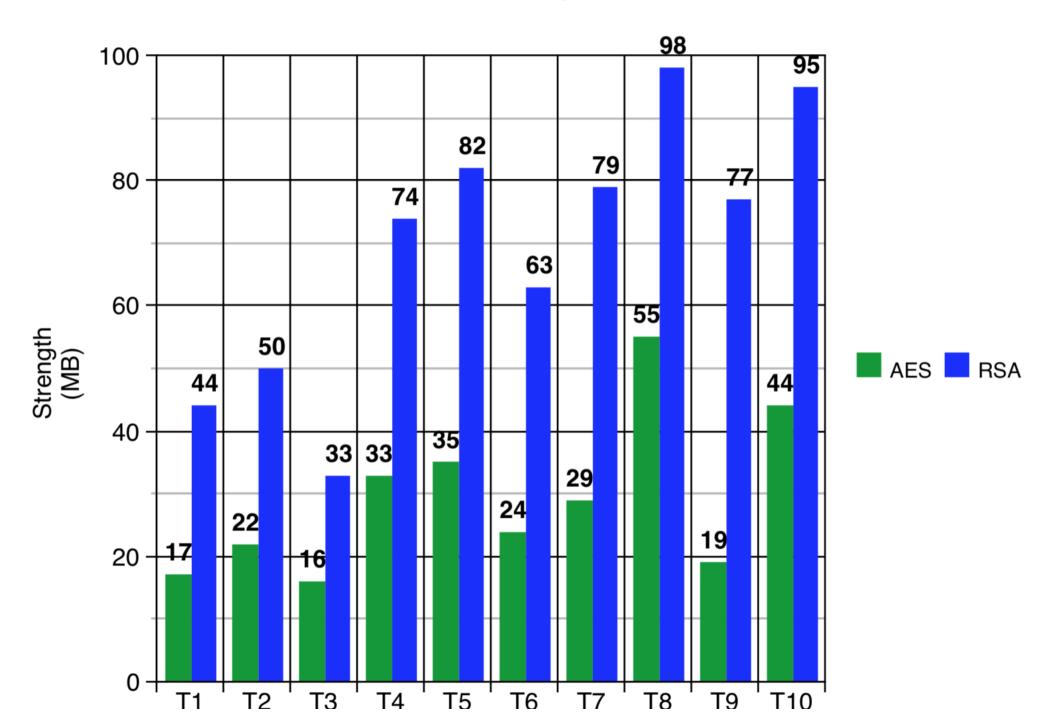
**RSA Speed**: Relatively Slow: Complex implementation does take time away from processing speed.

## Strength Test

Again I combined the results from over 100 tests into 10 since several tests were similar enough in results to condense. The bar chart does show that RSA encryption is hands down more secure based on the file size created after encryption.

**AES Encryption Stength**: Relatively Weak: Many files sizes created did not exceed 30 MB in encrypted text.

**RSA Encryption Stength**: Relatively Strong: Many files sizes created did exceeded 50 MB in encrypted text.



.. .\_ .. .. .. .. .. .. ..

Test

### Conclusion

Encryption Speed and Encryption Strength.

- Encryption Speed
- AES wins the speed test by a landslide. The additional calulations for breaking data into chunks and encrypting chunks and bytes separately leads RSA to a slower implementation.
- Encryption Stength
- RSA holds the title for strength in standard practice and in output file size. Creating files larger than those taken in creates a much more difficult decryption process and thus, stronger encryption.

### Metrics

Number of Tests Speed Simulation Pass/Fail Tests Accuracy Compare Results Examine Hypothesis **Record Conclusions** Publish

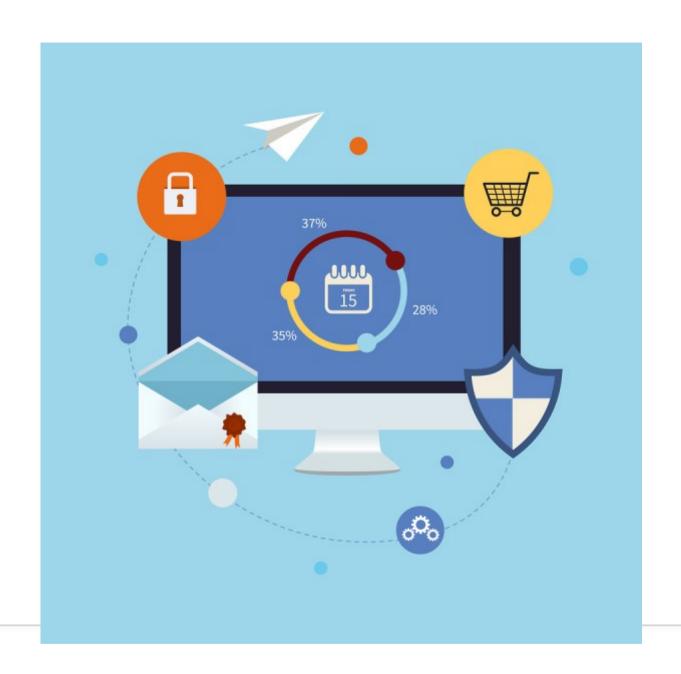
Files on Github.



```
r00t@r00t-KitPloit:~/KitPloit$ uncompyle6 -h
 uncompyle6 [OPTIONS]... [ FILE | DIR]...
uncompyle6 [--help | -h | --V | --version]
  uncompyle6
                       foo.pyc bar.pyc
                                                   # decompile foo.pyc, bar.pyc to stdout
  uncompyle6 -o . foo.pyc bar.pyc # decompile to ./foo.pyc_dis and ./bar.pyc_dis uncompyle6 -o /tmp /usr/lib/python1.5 # decompile whole library
Options:
                    output decompiled files to this path: if multiple input files are decompiled, the common prefix
  -o <path>
                    is stripped from these names and the remainder appended to
                      uncompyle6 -o /tmp bla/fasel.pyc bla/foo.pyc

-> /tmp/fasel.pyc_dis, /tmp/foo.pyc_dis

uncompyle6 -o /tmp bla/fasel.pyc bar/foo.pyc
                      -> /tmp/bla/fasel.pyc_dis, /tmp/bar/foo.pyc_dis
uncompyle6 -o /tmp /usr/lib/python1.5
-> /tmp/smtplib.pyc_dis ... /tmp/lib-tk/FixTk.pyc_dis
   --compile | -c <python-file>
                    attempts a decompilation after compiling <python-file>
                    print timestamps
  -p <integer> use <integer> number of processes
                    recurse directories looking for .pyc and .pyo files
                   use fragments deparser
  --fragments
                    compare generated source with input byte-code
  --verify-run compile generated source, run it and check exit code
  --syntax-verify compile generated source
--linemaps generated line number correspondencies between byte-code
                    and generated source output
  --encoding <encoding> use <encoding> in generated source according to pep-0263
  --help
                    show this message
 ebugging Options:
  --asm
                 -a include byte-code
                                                      (disables --verify)
  --grammar
                      show matching grammar
  --tree
                      include syntax tree (disables --verify add template rules to --tree when possible
                                                     (disables --verify)
  --tree++
Extensions of generated files:
   .pyc_dis' '.pyo_dis'
+ '_unverified'
                                successfully decompiled (and verified if --verify) successfully decompile but --verify failed
     + ' failed'
                                 decompile failed (contact author for enhancement)
 r00t@r00t-KitPloit:~/KitPloit$ ∏
    KitPloit : bash
IMPLEMENTATION
```



### CONCLUSION

# Closing Remarks. 🖫

There are only two kinds of companies:

- 1. Those that have been hacked.
- 2. Those that will be.
- ~ Robert Mueller

Best,

Brandon Rowe