Aaron Parks
CSS 527 Cryptology and Data Assurance
Autumn 2014
Final Project: Report

Part 0: System Information

This system was used for the development of "Numbers" as well as the scp trials.

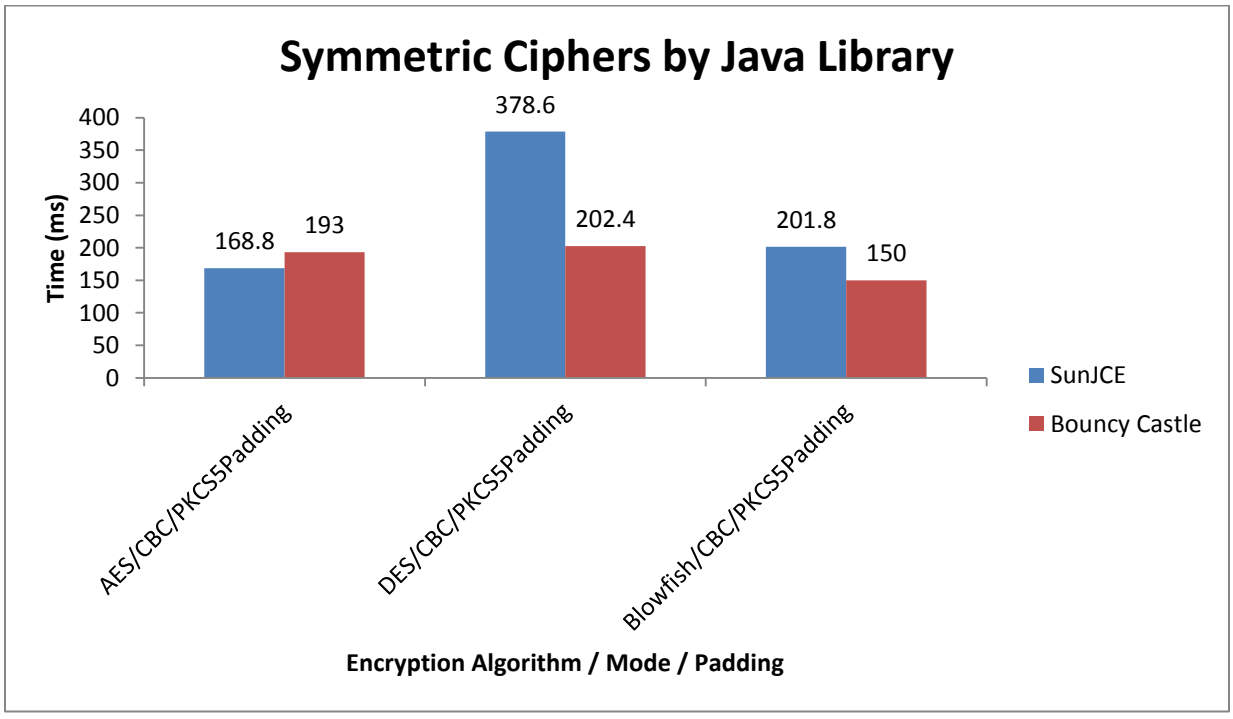
System Info	
Operating System	Windows 8.1 Professional 64-bit
CPU	4.1 Ghz, Dual-Core
RAM	16 GB
Network Connection	Wireless 802.11ac
Bandwidth	Bandwidth: 29.57 Mbps down, 6.2 Mbps up

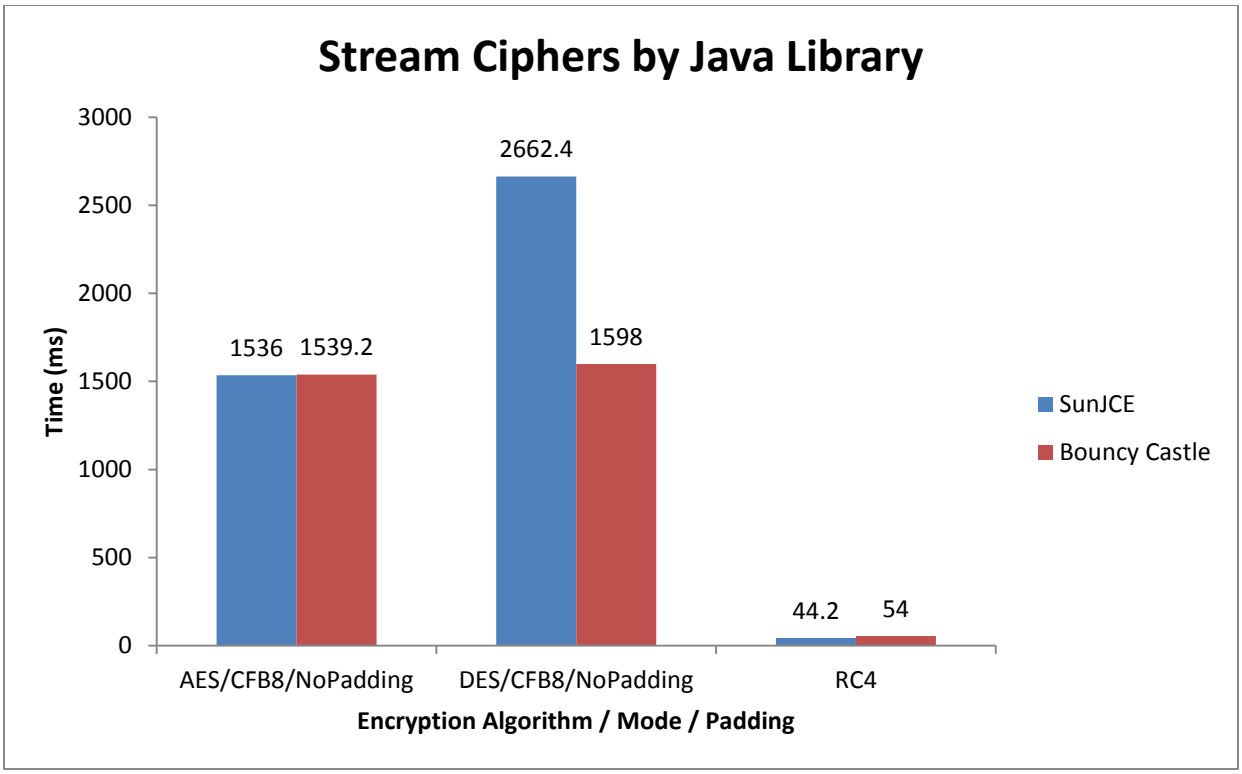
Part 1: Crypto Libraries in Java

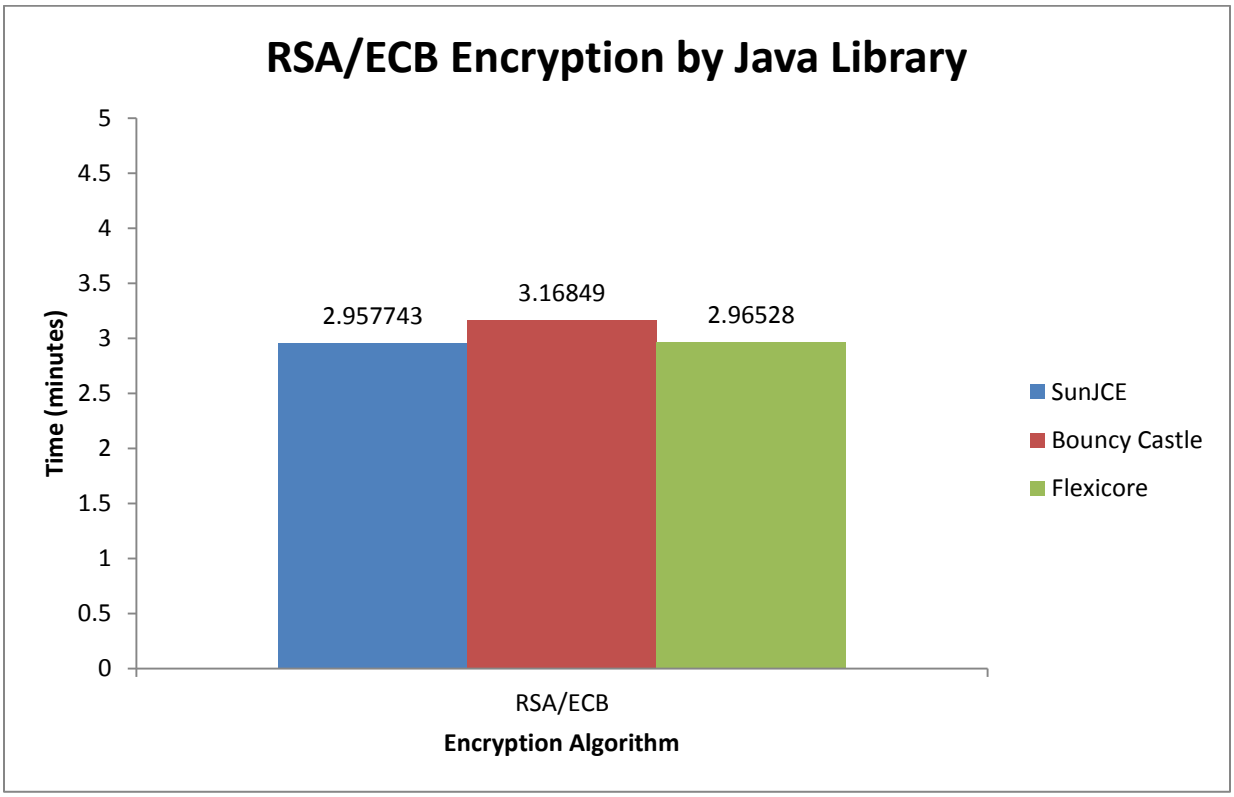
The program "Numbers" was written in Java an used the following three cryptography libraries:

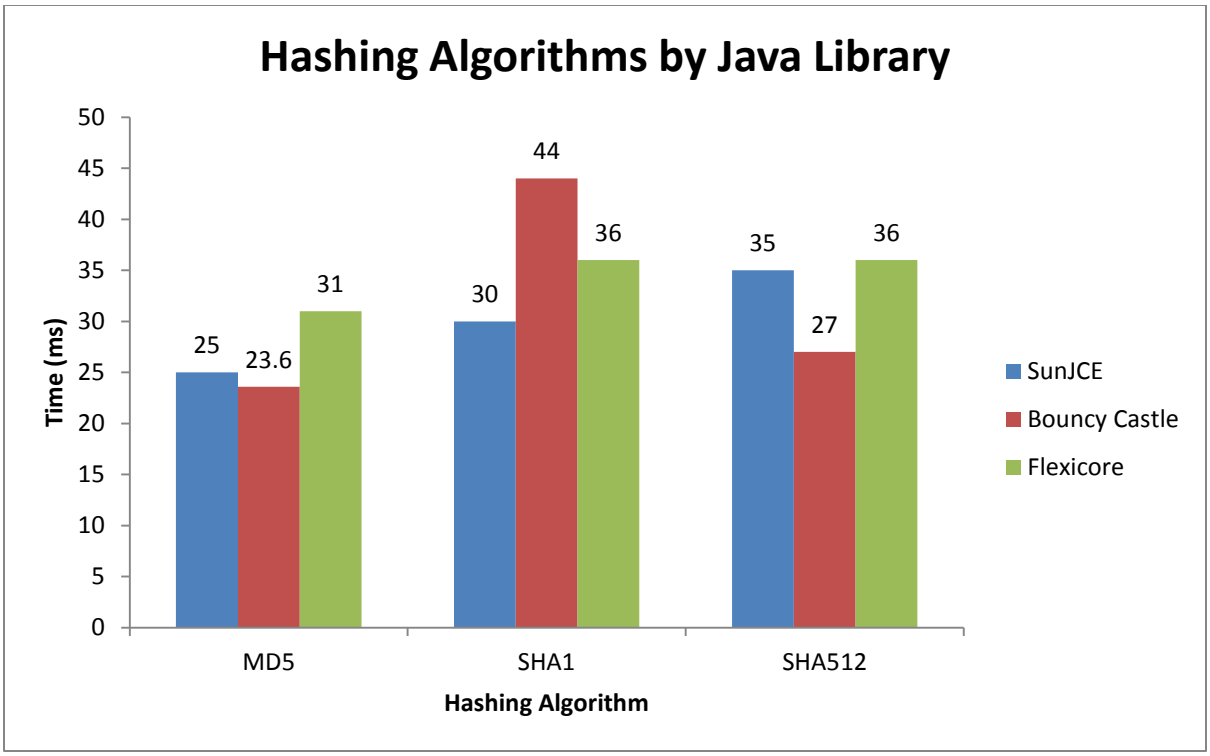
- SunJCE
- Bouncy Castle
- FlexiCore

Because of the difficulty of implementing asymmetric encryption in Java, only RSA was implemented in the project time frame. Additionally, FlexiCore did not integrate well into the existing SunJCE code for symmetric and stream encryption. The results of all the working trials are shown in the following four graphs.




Stream Ciphers by Java Library
3000
2500
2000
1500
1000
500
0
Time (ms)
1536
1539.2
2662.4
1598
44.2
54
AES/CFB8/NoPadding
DES/CFB8/NoPadding
RC4
Encryption Algorithm / Mode / Padding
SunJCE
Bouncy Castle


RSA/ECB Encryption by Java Library
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Time (minutes)
2.957743
3.16849
2.96528
RSA/ECB
Encryption Algorithm
SunJCE
Bouncy Castle
Flexicore



Discussion:

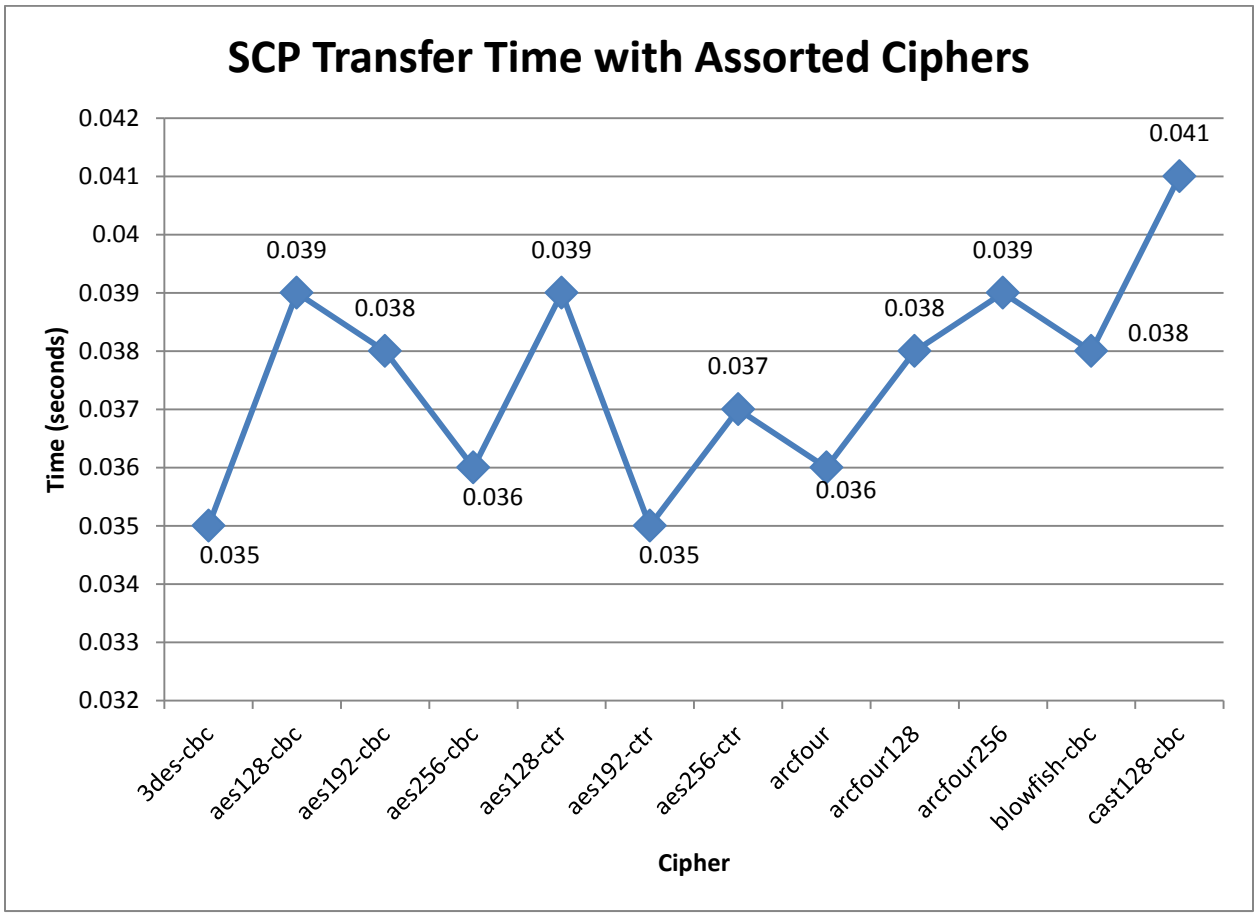
For the purpose of following comparisons, the built-in Java encryption library (SunJCE) will be the control. The performance of the other two libraries (where applicable) will be judged based upon the performance of the same task by SunJCE. For symmetric encryption, SunJCE and Bouncy Castle demonstrated relatively the same performance on two of the encryption methods: AES/CBC/PKCS5Padding (AES symmetric) and Blowfish/CBC/PKCS5Padding (Blowfish). Though the graph shows a difference of 50ms between SunJCE and Bouncy Castle for the Blowfish encryption, the difference in execution time between the two libraries is dwarfed by the exceptional performance of Bouncy Castle with DES/CBC/PKCS5Padding (DES symmetric) encryption. Bouncy Castle consistently performed DES encryption at least 150ms faster than SunJCE.

The superior performance of Bouncy Castle on DES encryption was also found when the modes on AES and DES encryption where changed to stream ciphers using CTR mode. Once again, the result of the AES/CTR8/NoPadding (AES stream) with SunJCE and Bouncy Castle were nearly the same. However, the performance of Bouncy Castle on DES/CTR8/NoPadding (DES stream) was once again, much faster than SunJCE using the same cipher. RC4 was the only non-block cipher used for these trials. SunJCE and Bouncy Castle were on par with each other using that encryption method.

For asymmetric encryption, all three libraries were able to be executed. Though the execution time for SunJCE, Bouncy Castle, and FlexiCore were roughly the same, it's interesting to note that the average time of SunJCE and FlexiCore are nearly the same (about a difference of 8ms when average execution time for both libraries was a little less than 3 minutes). Bouncy Castle performed relatively the same as the other two libraries, though the average execution time is noticeably slower than the other two.

When comparing hashing times, FlexiCore took longer than SunJCE for all three algorithms. However, it performed well against Bouncy Castle for SHA1 hashing. Save for the case of SHA1 hashing where it performed poorly compared to SunJCE and FlexiCore, Bouncy Castle was able to provide the fastest result for MD5 and SHA512 hashing.

Part 2: SCP Tests and Comparisons to Ciphers used in "Numbers"
The following table shows the results of the transfer of a 5mb file from a local system to a system at `uw1-320-lab.uwb.edu`:



Though the times shown seem to vary wildly among the different tests, the transfer times are actually between 0.035 to 0.041 seconds. The two ciphers used in this test as well as the "Numbers" program are AES128 in CBC mod and Blowfish in CBC mode.

Comparing the performance of the ciphers among the SCP transfers and the program shows some strange results. The SCP transfers all took about 40ms to encrypt, transfer, and decrypt. That seems very strange as the ciphers used were various symmetric and stream ciphers in various modes using various key sizes. I suspect that if the timing was able to be tracked to the microsecond there may have been more usable results.

Conclusion:

Overall, the best Java encryption library to use is either the built-in encryption library or Bouncy Castle. Both libraries performed well in different types of encryption so there's no need to discount either. Which library to use will be dependent on what type of encryption the user intends to implement. Both libraries had plenty of documentation and examples available. This fact alone is the main reason I manage to get both libraries working on all four categories. The other reason is that Bouncy Castle integrates well into the existing methods of SunJCE. It was as simple as adding the Bouncy Castle JAR files and adding a few lines of code to get it working. FlexiCore is the exact opposite of the two. There's very little documentation about it (unless you're looking for materials to build a deck) and it didn't integrate well with SunJCE. Sometimes it did, such as with hashing and RSA, but for symmetric and asymmetric encryption the library didn't like the key generator that SunJCE and Bouncy Castle used with no issues.