Assignment 2

Ashlesha Atrey 200203500 amatrey@ncsu.edu

Sai Kiran 200202554 ssiddin@ncsu.edu

October 2, 2017

Contents

List of Figures

List of Tables

1 Algorithms

1.1 AES 128-bit and 256-bit CTR mode

- AES algorithm was developed in 1998 by Joan Daemen and Vincent Rijmen.
- It is a symmetric key block cipher.
- Counter (CTR) mode converts the block cipher to stream cipher.
- It uses variable key length of 128, 192 and 256 bits.

1.1.1 Algorithm

Generate a 128-bit key or 256 bit key based on the application and need. We use KeyGenerator class in java to generate a key. The key generation should be random and each block of data should have a different key. SecureRandom class with NativePRNGNonBlocking is used and given as a parameter in key initialization and thus produces a secret key with the help of Secret class in Java.

After key generation, various processing is done on data and key to generate an encrypted message. This processing is done by the doFinal method of the Cipher class. There are various modes of processing. We used CTR mode in our experiment. First, we initialize the Cipher class with AES and give CTR as the parameter with no NOPADDING. With CTR mode, the number of bytes output is exactly equal to the number of bytes input, so no padding/unpadding is required. The PaddingScheme property does not apply for counter mode. CTR mode increments a counter for each subsequent block encrypted.[1]

The decrypting application would need to decrypt in the same manner. The 1st decrypt should begin with a new instance of a Cioher object so that it's counter is at the initial value of 0.

We time the encryption time and decryption time to analyze the performance of the algorithm using mean and median.

1.2 RSA 1024 and RSA 4096

- RSA was developed in 1977 by Ron Rivest, Adi Shamir and Len Adlemen.
- It is an asymmetric key cryptographic algorithm.

- The RSA uses the property that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product.

1.2.1 Algorithm

RSA is implemented by generating two pair of keys, Public key and the private key using Key pair class in java. Keys are generated pseudo randomly using SecureRandom class in java. RSA needs input in chunks of specific bytes. We implemented RSA 1024 and 4096. We use public key as an input along with the data for encryption. To decrypt, we give private key along with the ciphered text as the input and we get the original text back. We used SHA1PRNG padding in RSA. We are using PKCS1Padding as the padding in ECB mode. padding adds at least 11 bytes to the message, and the total padded message length must be equal to the modulus length, e.g. 128 bytes for a 1024-bit RSA key [2].

1.3 Hash

- We implemented three hashing algorithm HMAC MD5, HMAC SHA1 and HMAC SHA-256.

1.3.1 Algorithm

In hashing we first use secure random and key generator to generate a key. After that for encryption, we use Mac class for the specific hashing algorithm. In our experiment we are using key size of 256 bit. Hence with the key the data is encrypted.

1.4 Signature

1.4.1 Algorithm

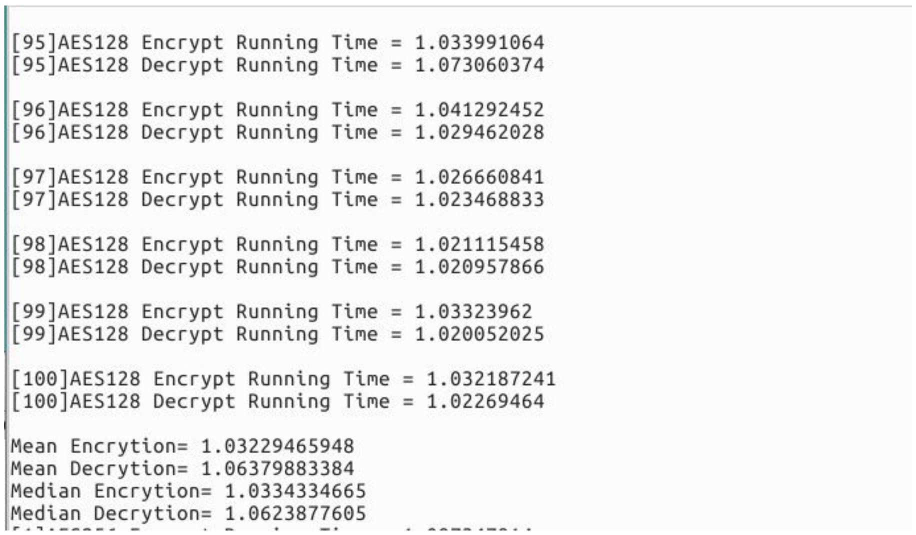
A digital signature is used to certify a document. We add a time stamp to it. So, when a document is changed or modified, then the verification fails. This process is different from the general encryption and decryption. The signer uses his/her private key for signing. The signature will then be verified using the corresponding public key. Another use of digital signature is for non-repudiation and integrity. This means that a sender cannot deny that he/she did not send the file once he/she has sent it already. In the case of RSA, when we use the signature, we do not use it directly. We use the concept of message digest to break the messages into smaller parts. We hash the message to be signed first. After this step only we apply the signature. This will not provide us a message that might lead to the original value. So, it is more secure.

2 Results

We evaluated the algorithm based on Encryption time and Decryption time. The encryption time is the time taken for encryption algorithm to create a cipher text from the original text. The decryption time is the time taken for decryption algorithm to create the original text from the cipher text.

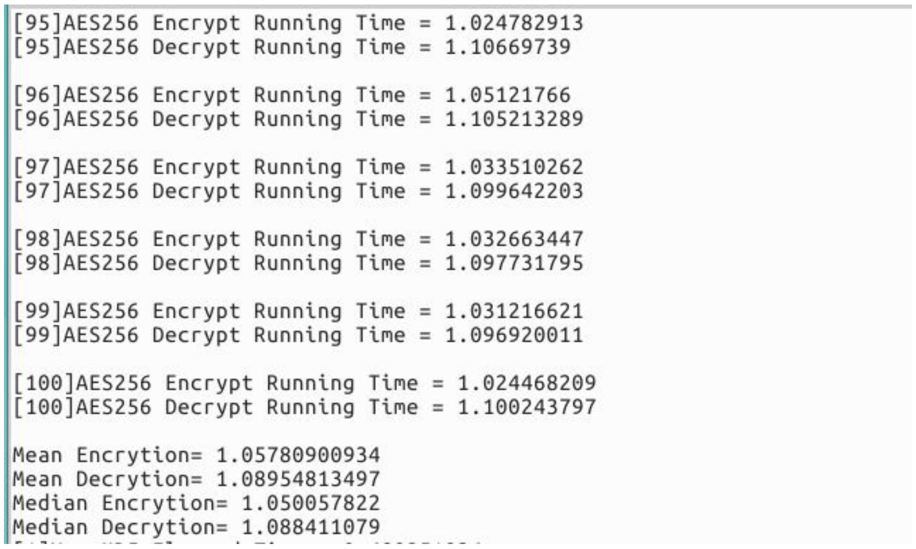
2.1 AES

AES 128 bit and AES 256 bit using CTR mode: Following screenshots shows mean and median encryption time and decryption time takes for AES 128 bit and AES 256 bit using CTR mode.



```
[95]AES128 Encrypt Running Time = 1.033991064
[95]AES128 Decrypt Running Time = 1.073060374

[96]AES128 Encrypt Running Time = 1.041292452
[96]AES128 Decrypt Running Time = 1.029462028

[97]AES128 Encrypt Running Time = 1.026660841
[97]AES128 Decrypt Running Time = 1.023468833

[98]AES128 Encrypt Running Time = 1.021115458
[98]AES128 Decrypt Running Time = 1.020957866

[99]AES128 Encrypt Running Time = 1.03323962
[99]AES128 Decrypt Running Time = 1.020052025

[100]AES128 Encrypt Running Time = 1.032187241
[100]AES128 Decrypt Running Time = 1.02269464

Mean Encrytion= 1.03229465948
Mean Decrytion= 1.06379883384
Median Encrytion= 1.0334334665
Median Decrytion= 1.0623877605
```

Figure 1: AES-128



```
[95]AES256 Encrypt Running Time = 1.024782913
[95]AES256 Decrypt Running Time = 1.10669739

[96]AES256 Encrypt Running Time = 1.05121766
[96]AES256 Decrypt Running Time = 1.105213289

[97]AES256 Encrypt Running Time = 1.033510262
[97]AES256 Decrypt Running Time = 1.099642203

[98]AES256 Encrypt Running Time = 1.032663447
[98]AES256 Decrypt Running Time = 1.097731795

[99]AES256 Encrypt Running Time = 1.031216621
[99]AES256 Decrypt Running Time = 1.096920011

[100]AES256 Encrypt Running Time = 1.024468209
[100]AES256 Decrypt Running Time = 1.100243797

Mean Encrytion= 1.05780900934
Mean Decrytion= 1.08954813497
Median Encrytion= 1.050057822
Median Decrytion= 1.088411079
```

Figure 2: AES-256

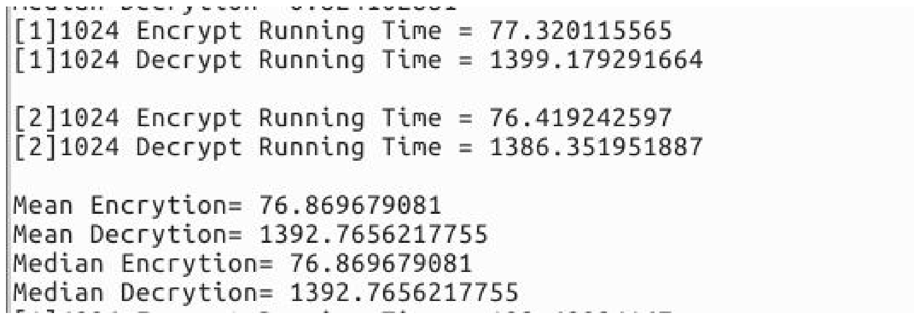
As from the test we can analyze that AES 128 takes approximately same time to encrypt and decrypt in CTR mode. This is because encryption and decryption are implemented in the same way in CTR mode. Similarly, for AES 256 bit using CTR mode we can observe that mean and median are quite similar. Further, we can notice that AES 128 and AES 256 have similar encryption and decryption time.

For a file of 100Mb, median encryption time is 1.03344 for AES-128. Hence the throughput is 96.7Mb/s. For AES-256 median encryption time is 1.05sec. Hence the throughput is 95.23 Mb/s.

For decryption of AES-128, the median time is 1.0623.Hence, the throughput is 94.1 Mb/s. For decryption of AES-256, the median time is 1.0884. Hence, the throughput is 91.8 Mb/s.

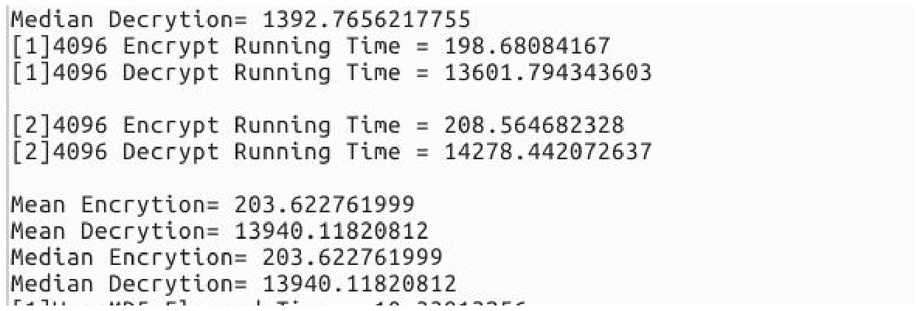
2.2 RSA 1024 and 4096 bit Algorithm

Following screenshot shows mean and median encryption time and decryption time takes for RSA 1024 bit and RSA 4096-bit algorithm.



```
[1]1024 Encrypt Running Time = 77.320115565
[1]1024 Decrypt Running Time = 1399.179291664

[2]1024 Encrypt Running Time = 76.419242597
[2]1024 Decrypt Running Time = 1386.351951887

Mean Encrytion= 76.869679081
Mean Decrytion= 1392.7656217755
Median Encrytion= 76.869679081
Median Decrytion= 1392.7656217755
```

Figure 3: RSA-1024



```
Median Decrytion= 1392.7656217755
[1]4096 Encrypt Running Time = 198.68084167
[1]4096 Decrypt Running Time = 13601.794343603

[2]4096 Encrypt Running Time = 208.564682328
[2]4096 Decrypt Running Time = 14278.442072637

Mean Encrytion= 203.622761999
Mean Decrytion= 13940.11820812
Median Encrytion= 203.622761999
Median Decrytion= 13940.11820812
```

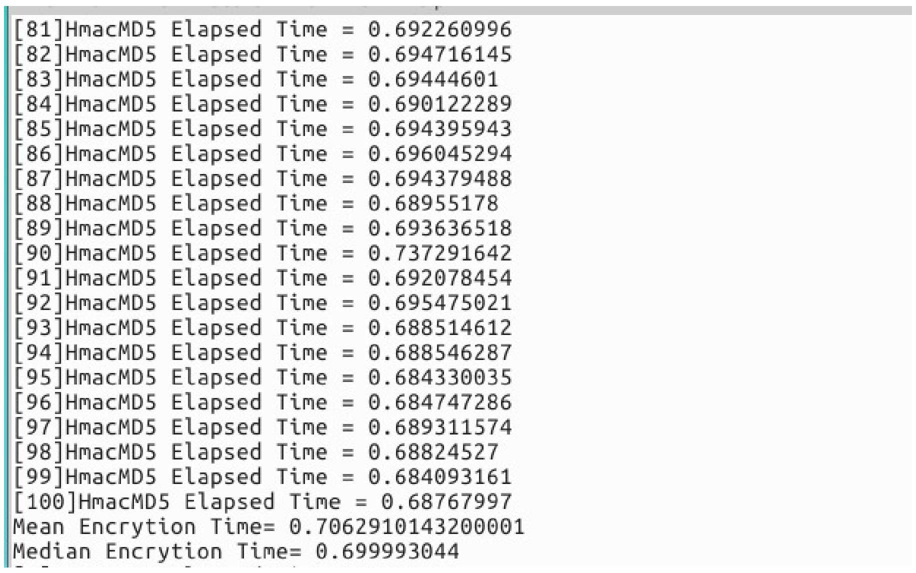
Figure 4: RSA-4096

As from the test we analyze that encryption time is very less compared to the decryption time in both RSA 1024 and RSA 4096-bit algorithm. With every doubling of the RSA key length, decryption is 6-7 times slower.[3]. Hence this statement is proved. For 100Mb file, the encryption time is 76.8 sec for RSA-1024. Hence the throughput is 1.302. The encryption time is 203.6 for RSA-4096. Hence the throughput is 0.049.

Decryption time is 1392.7 sec for RSA-1024. Hence the throughput is 0.071Mb/sec. Decryption time is 13940 sec for RSA-2048. Hence the throughput is 0.071Mb/sec.

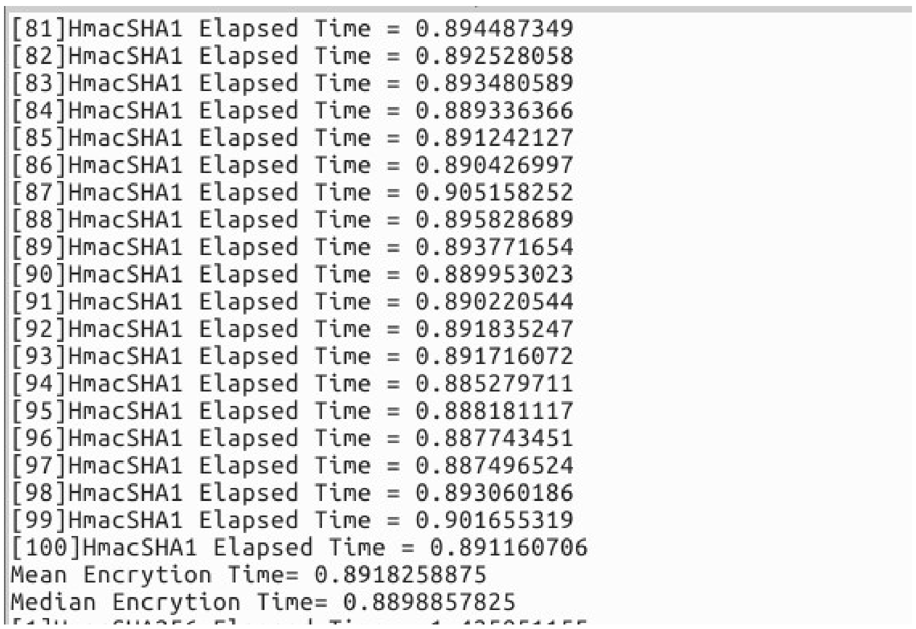
2.3 Hashing

Following screenshot shows mean and median encryption time and decryption time takes for hashing algorithms HMACMD5, HMACSHA1 and HMACSHA256.



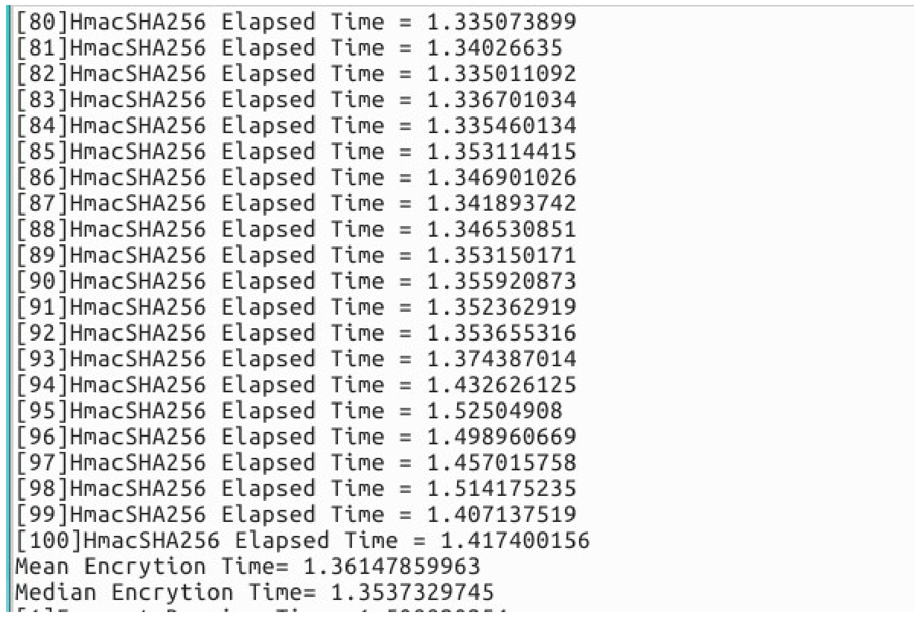
```
[81]HmacMD5 Elapsed Time = 0.692260996
[82]HmacMD5 Elapsed Time = 0.694716145
[83]HmacMD5 Elapsed Time = 0.69444601
[84]HmacMD5 Elapsed Time = 0.690122289
[85]HmacMD5 Elapsed Time = 0.694395943
[86]HmacMD5 Elapsed Time = 0.696045294
[87]HmacMD5 Elapsed Time = 0.694379488
[88]HmacMD5 Elapsed Time = 0.68955178
[89]HmacMD5 Elapsed Time = 0.693636518
[90]HmacMD5 Elapsed Time = 0.737291642
[91]HmacMD5 Elapsed Time = 0.692078454
[92]HmacMD5 Elapsed Time = 0.695475021
[93]HmacMD5 Elapsed Time = 0.688514612
[94]HmacMD5 Elapsed Time = 0.688546287
[95]HmacMD5 Elapsed Time = 0.684330035
[96]HmacMD5 Elapsed Time = 0.684747286
[97]HmacMD5 Elapsed Time = 0.689311574
[98]HmacMD5 Elapsed Time = 0.68824527
[99]HmacMD5 Elapsed Time = 0.684093161
[100]HmacMD5 Elapsed Time = 0.68767997
Mean Encrytion Time= 0.7062910143200001
Median Encrytion Time= 0.699993044
```

Figure 5: HMACMD5



```
[81]HmacSHA1 Elapsed Time = 0.894487349
[82]HmacSHA1 Elapsed Time = 0.892528058
[83]HmacSHA1 Elapsed Time = 0.893480589
[84]HmacSHA1 Elapsed Time = 0.889336366
[85]HmacSHA1 Elapsed Time = 0.891242127
[86]HmacSHA1 Elapsed Time = 0.890426997
[87]HmacSHA1 Elapsed Time = 0.905158252
[88]HmacSHA1 Elapsed Time = 0.895828689
[89]HmacSHA1 Elapsed Time = 0.893771654
[90]HmacSHA1 Elapsed Time = 0.889953023
[91]HmacSHA1 Elapsed Time = 0.890220544
[92]HmacSHA1 Elapsed Time = 0.891835247
[93]HmacSHA1 Elapsed Time = 0.891716072
[94]HmacSHA1 Elapsed Time = 0.885279711
[95]HmacSHA1 Elapsed Time = 0.888181117
[96]HmacSHA1 Elapsed Time = 0.887743451
[97]HmacSHA1 Elapsed Time = 0.887496524
[98]HmacSHA1 Elapsed Time = 0.893060186
[99]HmacSHA1 Elapsed Time = 0.901655319
[100]HmacSHA1 Elapsed Time = 0.891160706
Mean Encrytion Time= 0.8918258875
Median Encrytion Time= 0.8898857825
```

Figure 6: HMACSHA1



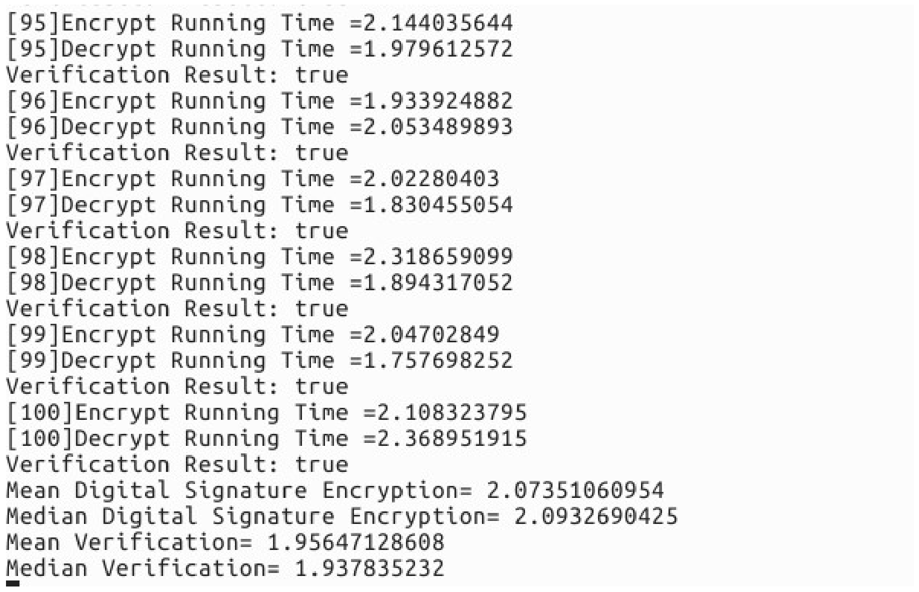
```
[80]HmacSHA256 Elapsed Time = 1.335073899
[81]HmacSHA256 Elapsed Time = 1.34026635
[82]HmacSHA256 Elapsed Time = 1.335011092
[83]HmacSHA256 Elapsed Time = 1.336701034
[84]HmacSHA256 Elapsed Time = 1.335460134
[85]HmacSHA256 Elapsed Time = 1.353114415
[86]HmacSHA256 Elapsed Time = 1.346901026
[87]HmacSHA256 Elapsed Time = 1.341893742
[88]HmacSHA256 Elapsed Time = 1.346530851
[89]HmacSHA256 Elapsed Time = 1.353150171
[90]HmacSHA256 Elapsed Time = 1.355920873
[91]HmacSHA256 Elapsed Time = 1.352362919
[92]HmacSHA256 Elapsed Time = 1.353655316
[93]HmacSHA256 Elapsed Time = 1.374387014
[94]HmacSHA256 Elapsed Time = 1.432626125
[95]HmacSHA256 Elapsed Time = 1.52504908
[96]HmacSHA256 Elapsed Time = 1.498960669
[97]HmacSHA256 Elapsed Time = 1.457015758
[98]HmacSHA256 Elapsed Time = 1.514175235
[99]HmacSHA256 Elapsed Time = 1.407137519
[100]HmacSHA256 Elapsed Time = 1.417400156
Mean Encrytion Time= 1.36147859963
Median Encrytion Time= 1.3537329745
```

Figure 7: HMACSHA256

Encryption and decryption of HMACMD5 is the fastest and HMACSHA256 takes the longest time. HMACSHA1 has encryption and decryption time in between those two.

2.4 Signature

Following screenshot shows mean and median encryption time and decryption time takes for signing and verification of the input file.



```
[95]Encrypt Running Time =2.144035644
[95]Decrypt Running Time =1.979612572
Verification Result: true
[96]Encrypt Running Time =1.933924882
[96]Decrypt Running Time =2.053489893
Verification Result: true
[97]Encrypt Running Time =2.02280403
[97]Decrypt Running Time =1.830455054
Verification Result: true
[98]Encrypt Running Time =2.318659099
[98]Decrypt Running Time =1.894317052
Verification Result: true
[99]Encrypt Running Time =2.04702849
[99]Decrypt Running Time =1.757698252
Verification Result: true
[100]Encrypt Running Time =2.108323795
[100]Decrypt Running Time =2.368951915
Verification Result: true
Mean Digital Signature Encryption= 2.07351060954
Median Digital Signature Encryption= 2.0932690425
Mean Verification= 1.95647128608
Median Verification= 1.937835232
```

Figure 8: Signature

As from the test we analyze that verification takes little less time than the signing the document.

3 Testing Platform

Testing platform is the machine on which we ran all our test. Test results may vary on different machines. We use the VCL machine to run the tests. We used an input file of 100Mb and ran several test on all the algorithms and found mean and median.

Characteristics of VCL machine:

- Architecture: x86-64
- CPU op-mode(s): 32-bit, 64-bit
- CPU(s): 2
- Model name: Intel(R) Xeon(R) CPU L5638 @ 2.00GHz
- CPU MHz: 2000.071

4 Relative performance of algorithms

4.1 AES algorithm

4.1.1 AES 128 VS 256

AES 256 is considered more secure than AES 128. This can be explained with an example of brute force attack. In the case of AES 256, the attacker must try 2 to the power256 combinations of keys to create an effective brute force attack. But in the case of AES 128, only 2 to the power 128 tries are required.

4.1.2 AES Encryption vs Decryption

The difference in time taken for execution in the process of encryption and in the case of decryption comes up because of the difference in steps followed in executing the algorithm during the encryption and decryption. In this project the CTR mode was used for implementing AES 128 as well as AES 256. The CTR mode involves the same method for encrypting and decrypting. So, there is hardly a difference between the time taken for the encryption and decryption. But AES can run on CBC mode as well. In this case, the encryption process involves encrypting the blocks sequentially one after the other. But in case of decryption it involves a parallel XOR operation where the operation happens simultaneously, thereby making the process of decrypting faster.

4.2 RSA Algorithm

4.2.1 RSA 1024 VS 4096

Just like the case of AES, an RSA 4096 takes more time to crack rather than the 1024 bit key. The 4096-bit key requires more effort to crack than the 1024 key.

4.2.2 RSA Encryption vs Decryption

In case of RSA, the time taken for encryption is lesser than time taken for decryption. The reason behind this lies in the algorithm used for these processes. In the case of encryption, the power is generally chosen to be a lesser value. This will complete the operation in a few operations of multiplication. But on the other hand, when we consider decryption, the operation is not as simple as that because it involves huge calculation of powers. Both involve modular expansion. But public encryption exponent is smaller and fixed compared to the private key decryption which takes a relatively larger time.

One of the reasons this is done is because of security reasons. We can choose a public key manually (say a smaller one), so the encryption is achieved faster. But the secret key is found with the help of this and various other parameters which increase the complexity of the calculation, thereby making it more time consuming. This also highlights the fact that a larger security key has a better security

4.3 AES vs RSA

AES and RSA have their own advantages. AES is a faster algorithm compared to RSA. This is highly evident from the time taken for encryption and decryption in the above experiments. This is because of the computational complexity involved. AES can be implemented by simple bit operations. But RSA needs complex math to be implemented. When the data is more, then the complexity becomes even higher. Thus, it would be an apt idea to use RSA for small amount of data that need to be protected and AES for larger amounts of data.

RSA algorithm uses prime numbers for its algorithm and AES uses the concept of symmetric keys. It is easier to calculate the primes faster than trying to brute force AES (If we use of quantum computers). Moreover, this is the case with AES with 128 bits. In case of 256, it becomes extremely secure and next to impossible to crack unlike RSA, which can be brute forced.

So, AES is considered a more successful algorithm in todays world and in the near future than RSA. But again it depends on the application's need.

4.4 HMAC

Firstly, HMAC is a hashing algorithm unlike AES and RSA. In the case of comparison of HMAC SHA256 and HMAC-SHA1, there is not much difference in the case of security performance. Both can resist well against a brute force attack as both of them possess a long tag. However, there has been recent events where a weakness for the SHA-1 has been exposed and the collision was computed at a speed much lesser than the estimated. But that implementation could not be done on SHA-256. Now, coming MD-5, it is much more prone to attacks(relatively) because it is very easy to find collisions in MD-5. In case of MD-5, the message digest is 128 Bits. So, for the attacks to happen we require 2 to the power 128-bit operations to get the original message. In the case of SHA, the message Digest Length is 160 bits. So, we will need 2 to the power160-bit operations. This makes it computationally hard for the attacker to get the original message in the case of SHA than MD-5. From the experiment conducted, the MD-5 was faster than SHA-1 which is followed by SHA-256. This is because it has only 64 iterations compared to SHA which has 80 iterations.

4.5 Signature

From the tests, we observe that the verification process takes lesser time than signing. The signing part is done with a more complex process which involves making message digests and then signing them. However, on the verification part, the process is relatively less complex. This accounts for the difference in speed.

5 Comparison

Here is a table presenting each operation (enc/dec/mac/sign/verify) for each relevant algorithm in a rank order by median time. All the times are in seconds.

Table 1: Encryption comparison based on ascending order of Median time

Algorithm	Encryption
AES-128	1.0334334665
AES-256	1.050057822
RSA-1024	76.86979801
RSA-4096	203.622761999

Table 2: Decryption comparison based on ascending order of Median time

Algorithm	Decryption
AES-128	1.0623877605
AES-256	1.088411079
RSA-1024	1392.7656217755
RSA-4096	13940.11820812

Table 3: MAC Encryption comparison based on ascending order of Median time

Algorithm	MAC
HMACMD-5	0.699993044
HMACSHA1	0.8898857825
HMACSHA256	1.3537329745

Table 4: Signature Encryption comparison based on ascending order of Median time

Algorithm	Time
Verification	1.937835232
Signature	2.0932690425

References

[1] *https://www.example-code.com/java/crypt2aes.ctr.asp.*

[2] *https://en.wikipedia.org/wiki/PKCS1*

[3] *http://www.javamex.com/tutorials/cryptography*

[4] *http://www.java2s.com/Code/Java/Security/Generatea1024bitRSAkeypair.htm*

[5] *http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html*