**Final Project - IT 3201**

1. Application of digital certificates requires encryption, decryption and digital signatures. To simulate the issuance of digital certificates, the following must be met.
2. A system that would encrypt and decrypt any file or data using self-defined algorithm combining RSA, shifting, transposition and OTP cryptography. The system will create a public and private key with the minimum of 64 bit. The public and the private keys should be stored in a file or anywhere to be used in encryption and decryption. (see sample below)
3. Before encryption, the system will check first the integrity of the file; this simulates the digital signature of a certificate.
4. Encrypt the data using the defined algorithm. This simulates the encryption of the data in a certificate.
5. To decrypt the file, the system should check the integrity of the file first, before decrypting this is where the digital signature is simulated, and if the file is compromised, then the decryption process cannot take place and if the file is authentic the decryption will pursue.
6. You can use any programming languages that you feel comfortable working with, including C programming (file handling is necessary). You can work by partners.
7. Document everything.

1. Requirements
2. Running System
3. Documentation of the developed system,

* formula,
* procedures,
* algorithm or flowchart,
* sample input and output
  + how a public key is created.
  + how a private key is created.
  + how a process or command is made to perform data integrity
  + then show the output
  + how a process or command is made to perform data encryption
  + then show the output
  + how a process or command is made to perform data decryption
  + then show the output

1. Sample output below:

------------------------------------------------------------------

To create Public key and store the key in a file for later use

c:\> certify createpubkey pubkey.key

------------------------------------------------------------------

To create Private Key and store the key in a file for later use

c:\> certify createprivkey privkey.key

------------------------------------------------------------------

Note: Plaintext should be inside a file.

Plaintext: Message.txt

Hello, welcome the world of encryption!

------------------------------------------------------------------

To perform data integrity:

c:\> certify datainteg message.txt sha1

Sha1 Value: SDFKNCB309234JLAKJDF0292U3987KAJSDFKAHP3987409823H

------------------------------------------------------------------

To perform data encryption

c:\> certify dataencryp message.txt pubkey.key ciphertxt.txt

message encrypted in file ciphertext.txt

lkjdfTERtye456rtTI.NCL.kjsS5675DFhafDS342FG123AF.m1JHLKnqoeu

------------------------------------------------------------------

To perform data decryption with verified data integrity authentication

c:\> certify datadecrypt ciphertext.txt privkey.key message.txt datainteg sha1

message decrypted and check integrity in file message.txt

Hello, welcome the world of encryption!

File Verification: Verified!

------------------------------------------------------------------

To perform data decryption with failed data integrity authentication

c:\> certify datadecrypt ciphertext.txt privkey.key message.txt datainteg sha1

message decrypted and check integrity in file message.txt

Hilhy, asecami tad wirlf if dncr8pt1on!

File Verification: Error! File was tampered or corrupted!

------------------------------------------------------------------

Terminologies used in this example:

* certify datainteg - command to perform data integrity
* sha 1 - hash method or any method for that matter when performing data integrity
* message.txt - plaintext or a file contains the text.
* certify createpubkey - command used to create a public key and stores it into a file
* certify createprivkey - command used to create a private key and stores it into a file
* pubkey.key - file where the result of a formula for (e)encryption was stored
* privkey.key - file where the result of a formula for (d)decryption was stored

------------------------------------------------------------------

1. Should you follow the given sample or create your own.
2. Print the output in a short size bond paper with your names, group, schedule and date submitted as header or footer of the document. Font size arial - 11pts.
3. Rubrics for checking the system:

1.0 - No errors were detected in the output; all requirements on each element were presented and indicated in the system.

2.0 - Minimal errors were detected in the output; some requirements on each element presented and indicated in the system.

3.0 - Major errors were detected in the output but the system was running; lesser requirements on each element were indicated in the system.

5.0 - No output; no system;

1. Rubrics for checking the document:

1.0 - All requirements on each element were presented and indicated in the document

2.0 - Limited requirements on each element presented and indicated in the document

3.0 - Less requirements on each element was indicated in the document

5.0 - No document submitted

1. Percentage output:
2. Document 10%
3. System - 90%
   1. self-defined Public key algorithm - 15%
   2. self-defined Private key algorithm - 15%
   3. data integrity output - 20%
   4. data encryption output - 15%
   5. data decryption output - 15%
4. System Integrity- 10%