Project 1
Due March 10th, 2005 at noon

Project description
Implement a symmetric encryption algorithm of your liking. Here are the specific requirements:

1. Decide on a set of features you would want to find in an encryption algorithm. For instance you might choose “hard to break and fast encryption/decryption”, or you might choose “easy to implement, no error propagation”, etc.
2. Use the knowledge gained so far in the course to select one encryption algorithm that satisfies your chosen set of features.
3. Write a program called supercipher that implements both the encryption and the decryption steps of the chosen algorithm. Once started, the program goes into infinite loop and in each iteration:
   a. Tells the user what encryption algorithm is being used and, if you have implemented a block cipher, what is the block size. Asks the user whether he wants to encrypt or decrypt
   b. If the user wants to encrypt:
      i. Asks for plaintext. Provide support for up to 10000 characters. Assume plaintext is inputted as a string of characters.
      ii. Asks for key length. Tell the user what key sizes are supported
      iii. Asks for key. Tell the user whether the key should be inputted as string of characters or in hexadecimal format
      iv. Encrypts the plaintext with the key and prints out the ciphertext. Depending on the encryption algorithm you used it might make more sense to print the ciphertext in hexadecimal format.
      v. Prints the time in microseconds that it took to do encryption
   c. If the user wants to decrypt:
      i. Asks for ciphertext. Tell the user whether the ciphertext should be inputted as a string of characters or as a string of hexadecimal numbers. For some encryption algorithms it will make more sense to input the ciphertext as hexadecimal numbers.
      ii. Asks for key length. Tell the user what key sizes are supported
      iii. Asks for key. Tell the user whether the key should be inputted as string of characters or in hexadecimal format
      iv. Decrypts the ciphertext with the key and prints out the plaintext as a string of characters
      v. Prints the time in microseconds that it took to do decryption
4. Modify your program supercipher into a program togglecipher where a user can specify the probability of a bit of ciphertext being toggled. Once started, the program goes into infinite loop and in each iteration:
   a. Tells the user what encryption algorithm is being used and, if you have implemented a block cipher, what is the block size.
   b. Asks the user for plaintext, key length and a key, same way as before.
c. Encrypts the plaintext with the key and prints out the ciphertext.
d. Asks the user for the probability of a bit being toggled, applies this to
   ciphertext and prints modified ciphertext.
e. Decrypts the modified ciphertext with the key and prints out the plaintext.
5. Modify your program `supercipher` into a program `losscipher` where a user can
   specify the probability of a part of ciphertext being lost. Once started, the program
   goes into infinite loop and in each iteration:
a. Tells the user what encryption algorithm is being used and, if you have
   implemented a block cipher, what is the block size.
b. Asks the user for plaintext, key length and a key, same way as before.
c. Encrypts the plaintext with the key and prints out the ciphertext.
d. Asks the user for the probability of a bit being lost, applies this to
   ciphertext to “lose” some bits and prints modified ciphertext.
e. Asks the user for the probability of a block being lost, applies this to
   ciphertext to “lose” some blocks and prints modified ciphertext. If you
   implemented a stream cipher assume a block size of 64.
f. Decrypts the modified ciphertext with the key and prints out the plaintext.
6. Provide a write-up containing four parts:
a. 1-page description of the algorithm you have implemented
   Provide a detailed description of your algorithm. Advocate your
   algorithm. Why have you chosen this one and not the others? What is it
   good for? Which applications would be likely to use this algorithm? Is it
   likely to be implemented in hardware or in software? Also discuss any
   drawbacks of the algorithm you have chosen.
   This page should contain your name somewhere on the top of the page.
b. Brute-force attack evaluation
   On a separate page, discuss the possibility of a brute-force attack on your
   algorithm. Remember that the difficulty of brute-force attack depends only
   on the key size, all the implementation details of your algorithm are
   known to the attacker. How many trials does the attacker have to make to
   break the code? Now measure the average time to decrypt a
   1000-character message using your algorithm and changing the key
   length. Assuming that the attacker will attempt to break your code by
   doing a brute-force attack and attempting to break 1000-character
   messages, plot the diagram showing:
   i. How the encryption time increases with key length
   ii. How the brute-force search time increases with key length
   iii. If you are using memory (e.g. for storing encryption tables) show
       on a separate graph how do memory requirements of your
       algorithm increase with key size
   Discuss any other attacks that might be possible against your algorithm
c. Ciphertext modification
   On a separate page discuss how your algorithm behaves if one bit of
   ciphertext is toggled. How many output bits are toggled? Is any part of the
   message retrievable or not?
d. **Ciphertext loss**
   On a separate page discuss how your algorithm behaves if one bit of ciphertext is lost. Is any part of the message retrievable or not? What if one block of ciphertext is lost (for stream ciphers assume that the block size is 64-bits)?

**Project Submission Instructions**
Create four folders: `supercipher`, `togglecipher`, `losscipher` and `writeup`. Place the source code, executables and Makefiles of programs in the corresponding folders. Place the PDF or PS file containing your writeup in the `writeup` folder. Create a folder named “YourName_project1” (naturally, replace YourName with your name, e.g. I would create a folder named JelenaMirkovic_project1) and place all four folders inside. Tar and zip this folder and send it as attachment by E-mail to Jelena at *sunshine@cis.udel.edu* and CC Ke at *kli@cis.udel.edu*.

**Some Guidelines**
Don’t choose a very difficult algorithm that you will need a month to implement. The main requirement for projects is that they work and that they are reasonably secure. If you have 1000 lines of code and the program doesn’t work then you can get maximum 10% of a project grade for the writeup and that’s it. If you really, really want to use DES or AES in your program, write a modular program that uses a simple cipher instead. Then try to implement DES or AES (or any other complex algorithm) and, if you are satisfied that it works, plug it in instead of your simple cipher.

If you really want to implement DES or AES (or any other complex algorithm) you must implement it yourself. There are some implementations online and you are allowed to look at those and copy-paste tables but you must write the rest of the code yourself.

If you are implementing stream ciphers, make sure that your cipher is truly random (i.e. implement a custom RNG, either the one we studied in the class or something else you found on the Internet. You must write the source code for this one yourself).

If you are implementing block cipher, address the “short block” problem either by padding or by ciphertext stealing.

Provide support for sentences, not only words of plaintext. This means that you should assume that inputted plaintext will have spaces inside.

Don’t forget to have **rounds** if you are implementing a simple block cipher. Don’t forget to have both substitution and permutation with block ciphers.