Computer Security Learning Laboratory: Implementation of DES and AES Algorithms using Spreadsheets

Oi-Shong Chok, Jayantha & Susantha Herath
St. Cloud State University, St. Cloud, MN 56301
choi9901@stcloudstate.edu, jherath@stcloudstate.edu, sherath@stcloudstate.edu

Abstract
Most computer science and information systems programs have two or more computer security courses but lack suitable active learning and design experience in the classroom. At the senior level computer security course focuses on the implementation details of different symmetric and asymmetric data encryption algorithms, among many other topics, using either programming languages or special hardware. Very often students have difficulty understanding those algorithms. To help students understand special features of those algorithms a computer security course has been developed with hands-on classroom activities, laboratories and web based assignments. This paper describes our experience in developing an active learning course module to help students understand DES and AES algorithms using Microsoft Office Excel XP learned in their computer literacy class. The user document with snapshots gives details for a user to input and use these DES and AES learning tools. Testing and security issues have been discussed and several improvements have been suggested.

Introduction
During last fifteen years, we have been experimenting with methods to improve the quality of teaching computer security courses for undergraduate computer science and information systems students. The goal has been and continues to be to help them become good information assurance and security experts in a relatively short period of time with both theoretical understanding and practical skills so that they can enter and make valuable contributions to the profession. Traditionally, computer security subject matter has been presented to a less than enthusiastic student body in a relatively passive classroom environment. In general, this chalk-talk instructional process consists of multiple copying stages: the instructor first copies notes from a textbook to his note book, then the instructor copies those notes onto the blackboard, thereafter the students copy notes into their note books. Moreover, each instructor allocates considerable amount of his/her time to prepare or update the same course material in each offering. In addition, there is both local and national need for high-quality trained labor with the ability to stay current with the technological advances in information assurance and security.

Computer security serves as a gateway for learning more advanced topics in information assurance and security courses offered in upper level undergraduate computer science and engineering courses. It is hypothesized that the learning rate can be increased if both the instructor and the student are active at the same time. Thus the performance of the students can be improved by converting the traditional passive classroom into an active hands-on learning environment [1][3]. Designing a course with learning-by-doing modules and making it available for other instructors on-line reduces the course preparation time for instructors and reduces multiple copying steps in the learning

process. Making those modules available on PDAs will help traditional undergraduate students as well as the adult learners.

Goals and Objectives

The main objective of this project is to develop computer security course modules for introductory level undergraduate students and the faculty. These active learning modules are central to achieve the following goals:

- To provide the students an engaging learning environment with necessary tools and training to become proficient in the computer security subject matter in a relatively short period of time.
- To provide implementation details of encryption and decryption algorithms using excel with hands-on skills, integration, team-work and hence to enhance the quality of the graduates.
- To focus on design variations to illustrate the principles of computer security.
- To provide the faculty and students modifiable on-line courseware with state-of-the-art hardware and software practice.

This paper describes the active learning module developed to implement advanced symmetric cryptographic algorithms, observe the inner working of these security algorithms and ways to improve the performance at many levels of the design. Following sections outline the details of implementing DES and AES using Microsoft Office Excel XP, goals achieved, difficulties encountered, future work and summary.

DES Algorithm Overview

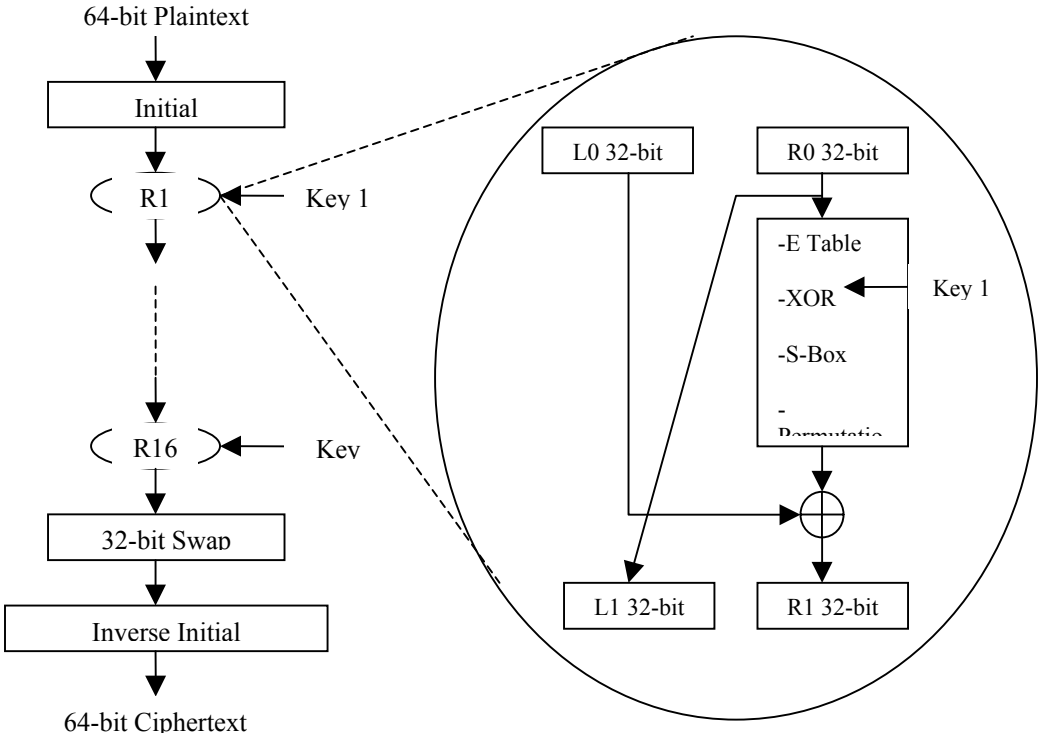


Figure 1: General overview of encryption for DES algorithm

The National Bureau of Standards has adopted DES in 1977 and now become Federal Information Processing Standard 46 (FIPS PUB 46) by National Institute of Standards and Technology (NIST). DES has been widely used for many years before it replaced by Advanced Encryption Standard (AES). Figure 1 illustrates the overview of the DES algorithm for encryption.

In DES, the fixed length input is 64-bit plaintext. The first operation in the DES encryption is to perform initial permutation on 64-bit plaintext before going into 16 rounds of the same function. In each round, it is involved both substitution and permutation. After 16 rounds, the left and right halves of the output will be swapped to perform the inverse initial permutation to produce 64-bit of ciphertext. [2]

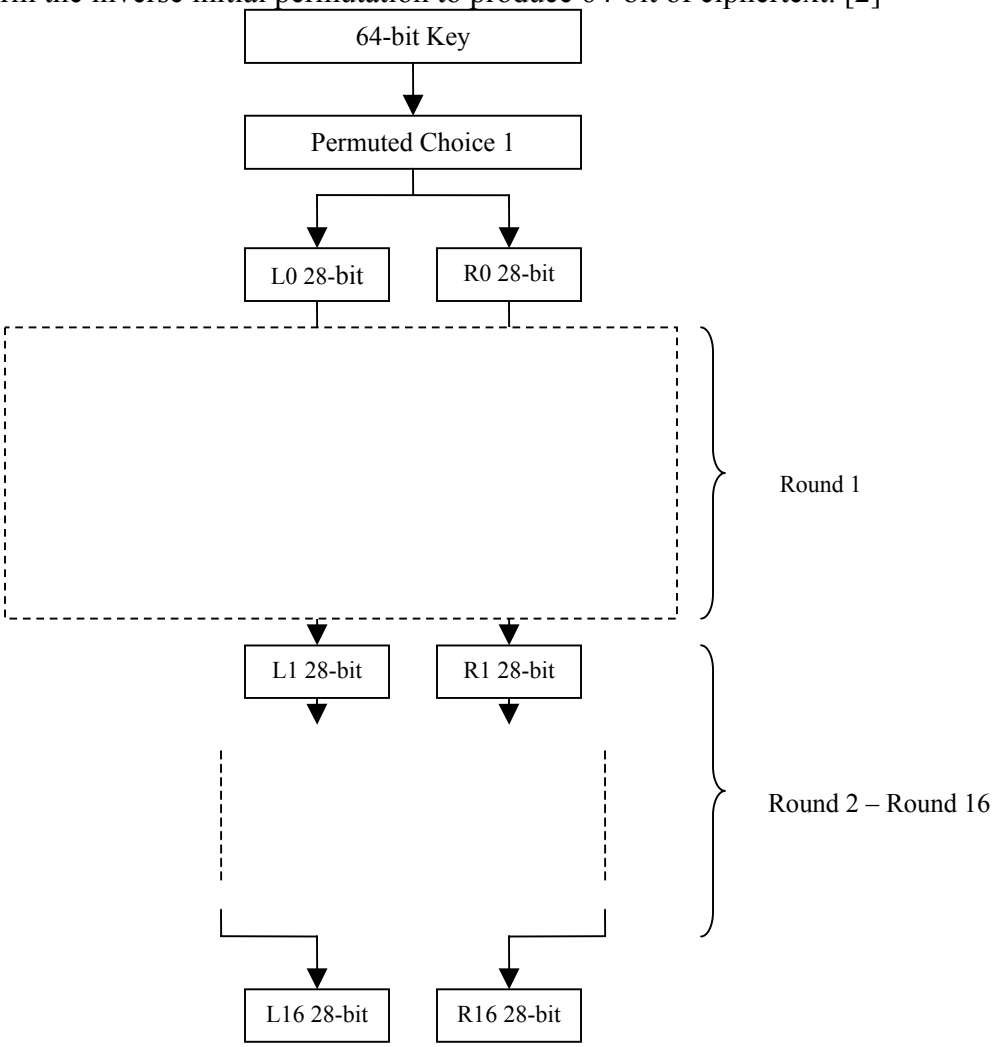


Figure 2: Key generation

During key generation, the first operation is permuted choice 1 and the output of this will be 54-bit key. There are 16 rounds in generating 16 keys for each round of the encryption of decryption function. The output of permuted choice 1 will be divided into 2 halves,

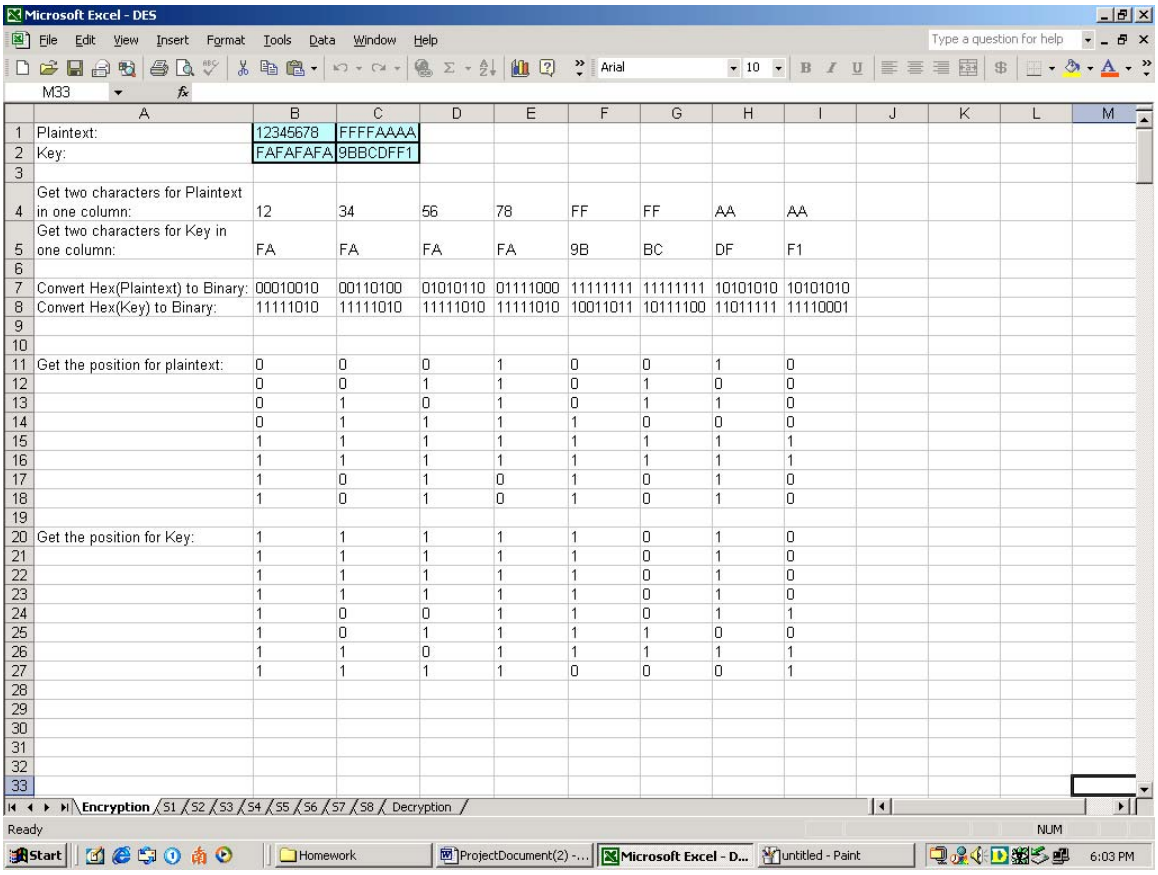
perform a left shift and thereafter pass through permuted choice 2 to produce the output of 48-bit key. Each key will perform XOR in each round of DES function.

For the decryption algorithm of DES, it uses the same algorithm as encryption. The only difference is use the reverse sub key after the key generation.

Tables such as Initial Permutation, Inverse Initial Permutation, Expansion Permutation, Permutation Function, S-Boxes, Permuted Choice 1 and 2 and Schedule of left shift are in [2].

User Document

Excel XP implementation of DES is available at [9]. When running this program, first open the worksheet file, *DES.xls*. Figure 3 depicts the snapshot of the DES program when it is opened. There are 10 worksheets in this file named as *Encryption, S1, S2, S3, S4, S5, S6, S7, S8* and *Decryption*. Figure 4 shows these 10 sheets tab in the DES worksheet.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Plaintext:	12345678	FFFFAAAA										
2	Key:	FAFAFAFA	9BBCDFF1										
3													
4	Get two characters for Plaintext in one column:	12	34	56	78	FF	FF	AA	AA				
5	Get two characters for Key in one column:	FA	FA	FA	FA	9B	BC	DF	F1				
6													
7	Convert Hex(Plaintext) to Binary:	00010010	00110100	01010110	01111000	11111111	11111111	10101010	10101010				
8	Convert Hex(Key) to Binary:	11111010	11111010	11111010	11111010	10011011	10111100	11011111	11110001				
9													
10													
11	Get the position for plaintext:	0	0	0	1	0	0	1	0				
12		0	0	1	1	0	1	0	0				
13		0	1	0	1	0	1	1	0				
14		0	1	1	1	1	0	0	0				
15		1	1	1	1	1	1	1	1				
16		1	1	1	1	1	1	1	1				
17		1	0	1	0	1	0	1	0				
18		1	0	1	0	1	0	1	0				
19													
20	Get the position for Key:	1	1	1	1	1	0	1	0				
21		1	1	1	1	1	0	1	0				
22		1	1	1	1	1	0	1	0				
23		1	1	1	1	1	0	1	0				
24		1	0	0	1	1	0	1	1				
25		1	0	1	1	1	1	0	0				
26		1	1	0	1	1	1	1	1				
27		1	1	1	1	0	0	0	1				
28													
29													
30													
31													
32													
33													

Figure 3: The outlook of the DES.xls file when it is opened

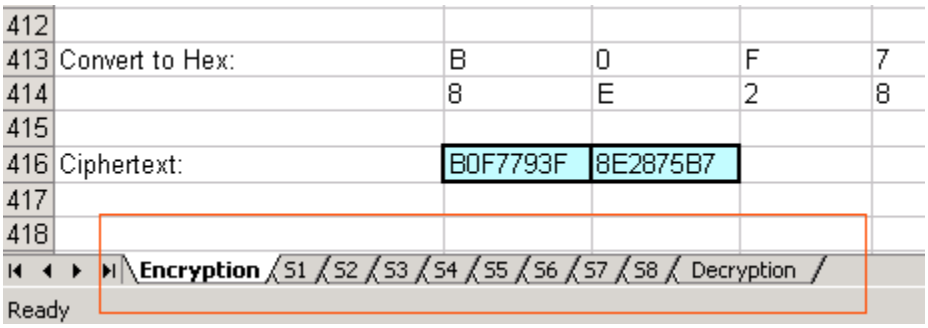


Figure 4: Ten sheets in DES worksheet

The function of *Encryption* sheet is to encrypt the 64-bit plaintext with 52-bits key into 64-bit ciphertext. The function of the *Decryption* sheet is to decrypt the 64-bit cipher text with the same 52-bit key that used in encryption into the 64-bit plaintext. *S1, S2, S3, S4, S5, S6, S7* and *S8* sheets contain S-box values which are needed in both encryption and decryption algorithms.

All the cells in *Encryption* sheet are protected except for the cells B1, B2, C1 and C2. These cells are not protected for the purpose of entering plaintext input and key input. Only 8 characters can be input in each cell. When selecting any of B1, B2, C1 or C2 cell, there is a message describing the kind of the input needed to be entered in the selected cell. Error message will appear if the length of the input string is not equal to 8. B1, B2, C1 and C2 cells cannot be empty. This program only will work properly if the input string is in hexadecimal. Figure 5 shows the location of the plaintext and key are entered.

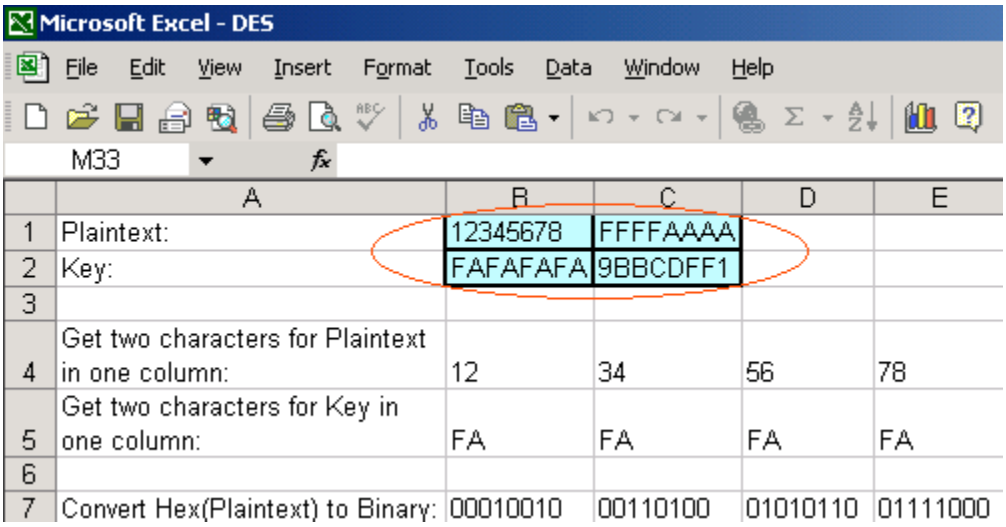


Figure 5: Location of input plaintext and key with blue color highlighted in *Encryption* Sheet

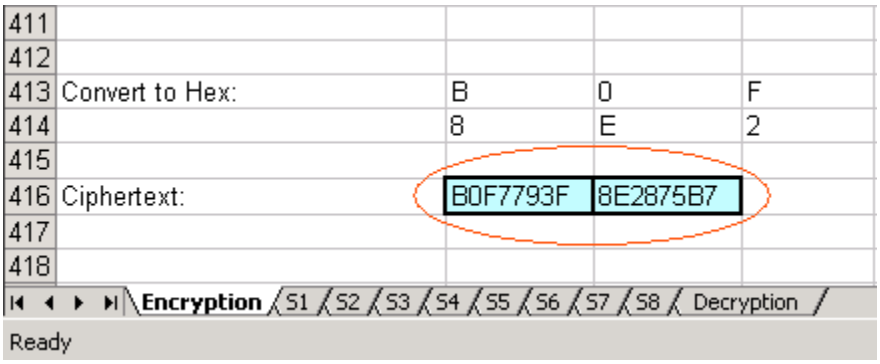


Figure 6: Location of Ciphertext after encryption done in *Encryption* Sheet

As soon as the plaintext and key are entered into the *Encryption* sheet, other cells will change the values according to the input and generate the cipher text in the same sheet in cells B416 and C416. Figure 6 shows the location of the ciphertext in the *Encryption* sheet.

The generated cipher text can be decrypted in *Decryption* sheet. No input is necessary in the *Decryption* sheet since the input of the ciphertext is referring to the ciphertext in the *Encryption* sheet as shown in Figure 7. In this Decryption sheet, Figure 8, plaintext will be produced in cell B416 and cell C416.

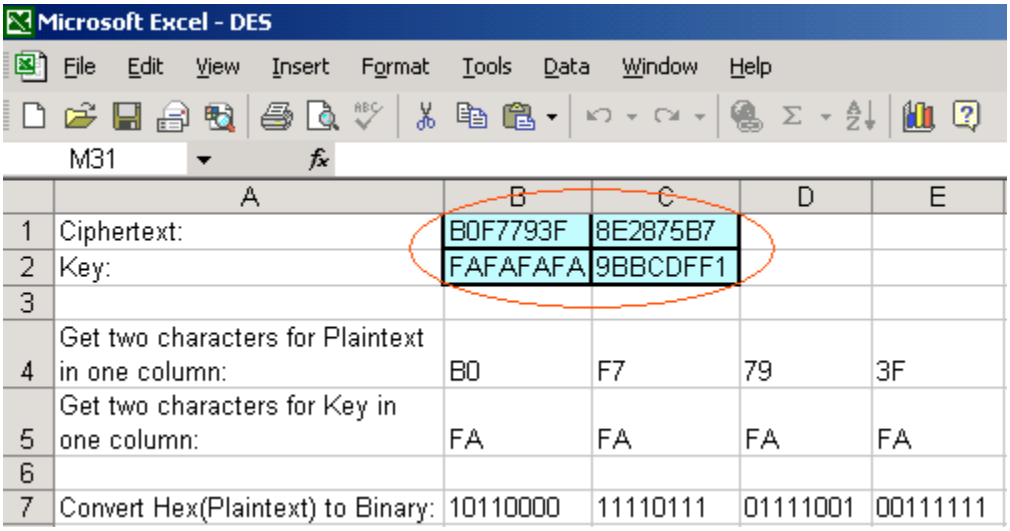


Figure 7: Location of Ciphertext and key that need to be decrypted in *Decryption* Sheet

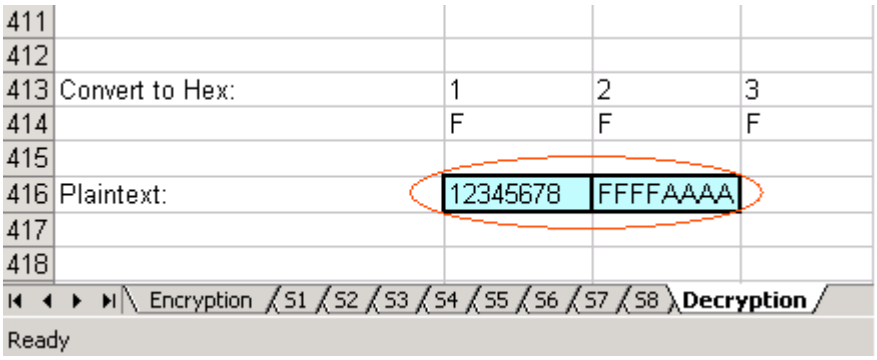


Figure 8: Output Plaintext of the Decryption in *Decryption* Sheet

Functions used for implementation of DES

Some of the functions needed for implementation of DES algorithm in Microsoft Office Excel are not found unless otherwise the Analysis Tool Pack is installed. This Analysis Tool Pack provides functions and interfaces for financial and scientific data analysis. Therefore, before the implementation of DES, check whether the Analysis Tool Pack has been installed or not. If not, go to the menu *Tools* and click on *Add-Ins*...to select Analysis Tool Pack and install it. Functions that are needed to implement DES algorithm are:

- LEFT(text, num_chars)
- RIGHT(text, num_chars)
- LEN(text)
- BIN2HEX(number, places)
- BIN2DEC(number)
- HEX2BIN(number, places)
- DEC2BIN(number, places)
- CONCATENATE(text1, text2…)
- INDEX(array, row_num, column_num)
- VALUE(text)
- IF(logical_test, value_if_true, value_if_false)

All these functions are very helpful especially converting hexadecimal to binary. There are more functions that can be found in *Worksheet functions listed by category* through the Microsoft Excel Help. However, the listed functions above are sufficient for implementing DES algorithm in Microsoft Office Excel XP.

Design choices for Implementing DES algorithm

There are many design choices to implement the DES algorithm. An efficient way to implement DES is to have different sheets with different functions. Each sheet in a worksheet is similar to a function in C++. For example, there are encryption and decryption functions which are mainly used to encrypt a given plaintext or decrypt a given ciphertext. After recognizing the functions that are needed, constant values such as

S-boxes can be stored in different sheet in a worksheet. All the sheets in a worksheet can be inserted and renamed.

Good design techniques are needed in implementation to avoid unnecessary repeated work. For example, in Microsoft Office Excel XP, one formula such as *=C2* has been entered in cell B2. In order to get the same result in other column such as cell C2 has formula *=D2*, it is very easy in Microsoft Office Excel XP by clicking on the cell and dragging the mouse arrow across the destination cell. This will result every single cell selected to have the same formula or function with the change of the cell number. Hence, for implementing DES algorithm, it is good to design that all the operations that performed in each round should be in the same column. Now a click and drag of the first round column until 10th round column will give the same formula and functions. It will save the time in doing the repeated work and avoiding mistakes which will save the time in looking or searching for the mistakes in a formula or functions.

It is always good in practice to put comments while implementing any program. Column A has been reserved to enter comments and explanations. In addition to column A, any necessary comments that needed also can be written down in any cell. It is good to leave a space in between any different operation. There are 3 main parts in the *Encryption* sheet which are:

1st part: - Convert Hexadecimal to Binary for both plaintext and key, rearrange each bit in each cell for both plaintext and key, perform initial permutation for plaintext and perform permuted choice 1 for key.
2nd part: - Calculate 16 sub keys and perform 16 round functions for encryption.
3rd part: - 32-bit swap for left and right halves, perform inverse of permutation and convert back to Hexadecimal from binary.

The main reason to divide *Encryption* into 3 parts is because 1st and 3rd parts actually are pre and post calculation for the encryption. They are never repeated in the whole encryption process. However, in the 2nd part, each operation is repeated in 16 rounds. In fact, it is the *for* loop in any high level programming language such as in C++. Yet, it is necessary to check other rounds after performing the click and drag. The reason is some of the variables are increasing with the formula or function in other rounds which are not necessarily needed. For instance, the cell B284 has formula *=DEC2BIN(INDEX('S1'!A1:'S1'!P4, B266+1, B267+1),4)*. When dragging the cell from B284 to C284, the variables *'S1'!A1: 'S1'!P4* will increase to *'S1'!A2:'S1'!P5* in cell C284. This will return a wrong array in another sheet such as S1. In the whole program, *'S1'!A1: 'S1'!P4* is the only needed formula. Therefore, rechecking the formulas is necessary and important.

The decryption algorithm, implemented in *Decryption* sheet and has the same structure as *Encryption* sheet, consists of 3 main parts as well. The only difference is using inverse 16 sub key in each round.

Testing

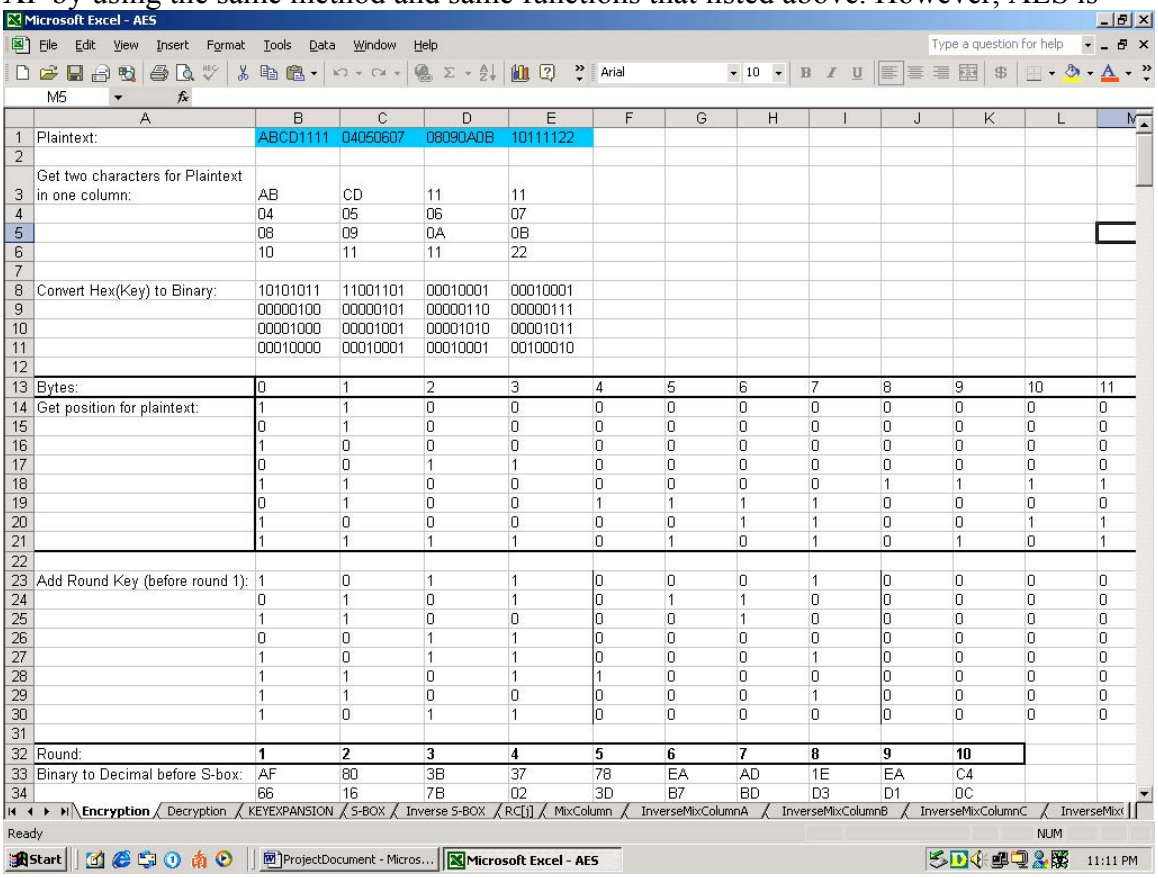
After finishing the implementation, it is important to have a set of test cases such as what is the output of each round for the encryption and decryption. It is very useful to debug in spreadsheet since there are many cells with formulas. Without a good test case will need more time to recheck every single cell. Ready test cases are available in [6-7].

Security Issues

Every single cell in the spreadsheet can be modified easily. Sometimes the value or formula can be changed easily when accidentally click on another cell. It is necessary to prevent this happening. In Microsoft Office Excel XP, spreadsheet can be protected with a password. In order to protect the spreadsheet, go to the *tool* in the menu and then click on the *protection*. There are four different ways in protecting the spreadsheet. In this DES implementation, the input cells used to enter plaintext and key are not protected. Therefore, choosing *Allow Users to edit Ranges ...* can let you to choose the cell that is not needed to be protected. Otherwise, all the other cells will be protected. It is essential to have this worksheet protection since all the cells can easily be changed affecting the result. An error message will be prompted if someone double clicks on the protected cell.

AES Implementation

Advanced Encryption Standard, AES, has been implemented in Microsoft Office Excel XP by using the same method and same functions that listed above. However, AES is



	A	B	C	D	E	F	G	H	I	J	K	L
1	Plaintext:	ABCD1111	04050607	08090A0B	10111122							
2												
3	Get two characters for Plaintext in one column:	AB	CD	11	11							
4		04	05	06	07							
5		08	09	0A	0B							
6		10	11	11	22							
7												
8	Convert Hex(Key) to Binary:	10101011	11001101	00010001	00010001							
9		00000100	00000101	00000110	00000111							
10		00001000	00001001	00001010	00001011							
11		00010000	00010001	00010001	00100010							
12												
13	Bytes:	0	1	2	3	4	5	6	7	8	9	10	11
14	Get position for plaintext:	1	1	0	0	0	0	0	0	0	0	0	0
15		0	1	0	0	0	0	0	0	0	0	0	0
16		1	0	0	0	0	0	0	0	0	0	0	0
17		0	0	1	1	0	0	0	0	0	0	0	0
18		1	1	0	0	0	0	0	0	1	1	1	1
19		0	1	0	0	1	1	1	1	0	0	0	0
20		1	0	0	0	0	0	1	1	0	0	1	1
21		1	1	1	1	0	1	0	1	0	1	0	1
22													
23	Add Round Key (before round 1):	1	0	1	1	0	0	0	1	0	0	0	0
24		0	1	0	1	0	1	1	0	0	0	0	0
25		1	1	0	0	0	0	1	0	0	0	0	0
26		0	0	1	1	0	0	0	0	0	0	0	0
27		1	0	1	1	0	0	0	1	0	0	0	0
28		1	1	0	1	1	0	0	0	0	0	0	0
29		1	1	0	0	0	0	0	1	0	0	0	0
30		1	0	1	1	0	0	0	0	0	0	0	0
31													
32	Round:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
33	Binary to Decimal before S-box:	AF	80	3B	37	78	EA	AD	1E	EA	C4		
34		66	16	7B	02	3D	B7	BD	D3	D1	0C		

Figure 9: The outlook of the AES.xls file when it is opened

more complicated than DES and need many calculations especially for the mix column and inverse mix column. In AES program that has been implemented, there are 11 sheets in one worksheet such as Encryption, Decryption, KEYEXPANSION, S-BOX, Inverse S-BOX, RC[j], MixColumn, InverseMixColumnA, InverseMixColumnB, InverseMixColumnC and InverseMixColumnD. Figure 9 illustrates the opened AES.xls file.

For AES, the size of the input plaintext is 128 bits. However, the size of the key can be 128 bit, 192 bit or 256 bit and each expanded key size is corresponding to the key size which is 176 bytes, 208 bytes or 240 bytes. Number of rounds also depends on the size of the key chosen and this can be 10, 12 and 14 rounds. Yet, the round key size is 128 bit for all variations of key size [2]. For this implementation, 128 bit of key has been chosen and key is expanded through the *KEYEXPANSION* sheet. The input key is entered in the *KEYEXPANSION* sheet. Figure 10 illustrates the Key expansion snapshot.

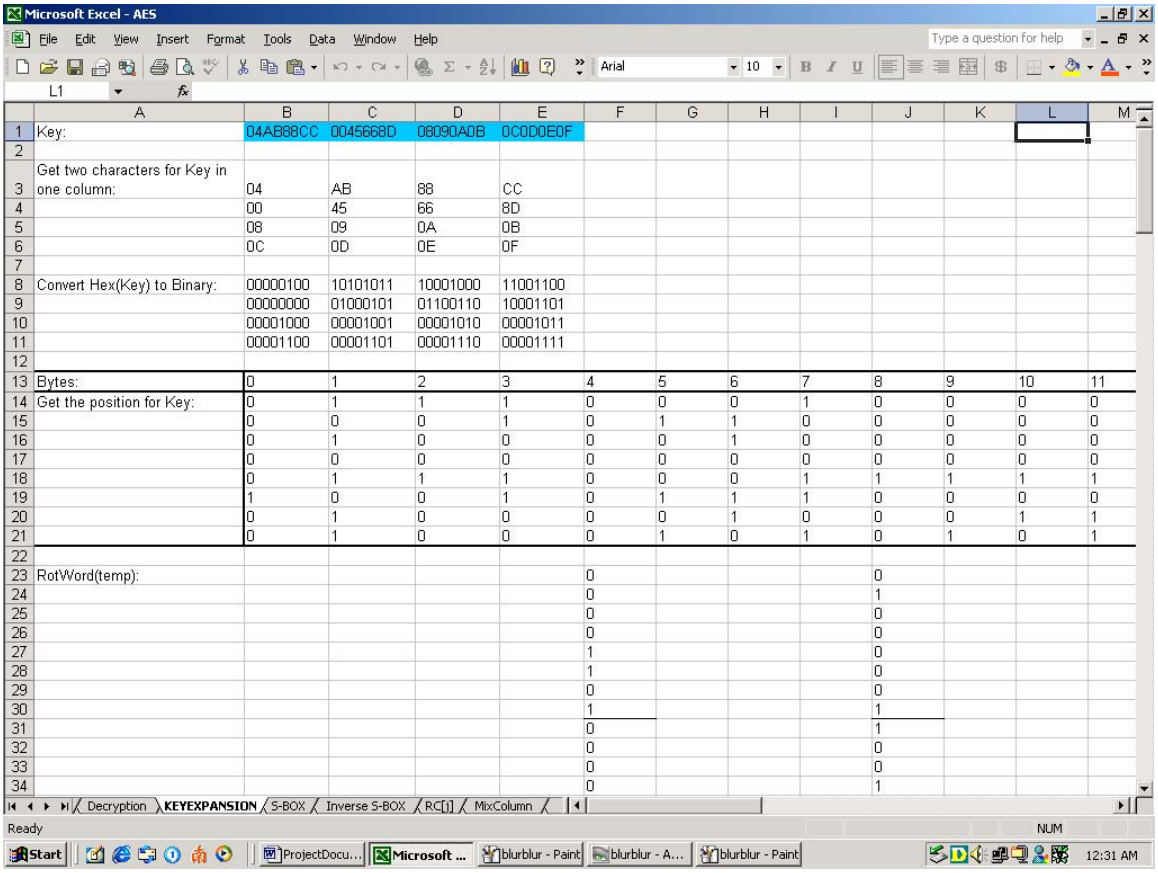


Figure 10: Key Expansion

There are four different stages in AES algorithm which are Substitute bytes, Shift rows, Mix Columns and Add round keys. In Substitute stage, 8 S-boxes are used for substitution for each byte. Shift rows is a simple permutation and Mix Columns is another permutation by using arithmetic over GF(2^8). Add round keys is a simple bitwise XOR with expanded key.

There are four different sheets for inverse mix column calculation namely InverseMixColumnA, InverseMixColumnB, InverseMixColumnC and InverseMixColumnD. Each sheet will generate corresponding column of the output for the matrix multiplication. For example, InverseMixColumnA will generate the first column of the output for the matrix multiplication which is the first 4 bytes for 16 rounds. The reason for choosing this method is because the formula for getting the output for each column is the same. Therefore, these four sheets have the same formula except for the input key for calculation. This is another advantage in doing this for saving time and work. However, for the Mix Column in encryption algorithm, there is only one sheet for this it is because there are not many steps in calculation compared to Inverse Mix Column. Test cases used for this implementation are in [8]. Details of AES algorithm can be found in [2].

Information Security Symposium

In addition to developing active learning tools [9-11] we organize local computer security symposiums annually to stimulate our students, computer science, information systems and engineering faculty in five neighboring states as well as businesses and industry. Last two symposiums were well attended by students, faculty and industry representatives. Invited speakers delivered lectures based on their work. These symposiums helped our efforts to develop a curriculum emphasizing secure storages, forensics, network-database security that presents an integrated view of hardware, software and other important security issues to the undergraduate students [12].

Conclusion

There is both local and national need for high-quality trained information system security professionals with the ability to learn in a short period of time and stay current with the information technology advances. Colleges can help to meet this demand by redesigning their information security courses, integrating security aspects in every possible course and allowing more students to succeed at the entry level. Successful application of active learning with rapidly changing technologies in the learning process is a way to remove the difficulties at entry level. Such changes will help the students to improve their skills. Consequently, both the public and private sectors will benefit with the greater number of highly-skilled trained professionals. There is a big demand in information security courses with learning-that-lasts modules throughout the country. To supply the demand we have been developing information security courses integrated with hands-on classroom activities and web based tools. Our course materials are available to others for their classroom use.

This paper discussed the implementation of DES and AES algorithms using Excel XP as a teaching-that-lasts-long module. This active learning module helps the students to understand the symmetric key cryptographic algorithms quickly enough to prepare for the design. Knowing each function makes the implementation easy and save time. Test cases are essential part of the implementation especially in debugging the program. Overall, the implementation of DES is easier than the implementation of AES in spreadsheet and not all the algorithm is suitable to be implemented in spreadsheets. However, implementing

DES algorithm and AES algorithm are new experiences and challenging for the students do.

Future Work

This module can be expanded more and make user friendly by creating a user interface to prompt for the input instead of just inserting the string into spreadsheet. Furthermore, the program can be expanded to take care of some invalid inputs such as the input has to be hexadecimal and no space is allowed.

References

1. Information Security Course Based on Applications of Management Techniques in Digital Systems for Business Colleges, Herath et al, The 2002 International Conference on Security and Management, (SAM'02), Las Vegas, USA, 23-25, June 2002
2. Stallings, W., Cryptography and Network Security, Third Edition. Upper Saddle River, NJ: Prentice Hall, 2003.
3. Integration of computer security laboratories into computer architecture courses to enhance undergraduate curriculum, Herath et al., Proceedings of Workshop on Computer Architecture Education, June 9-11, 2003, San Diego, CA http://www4.ncsu.edu/~efg/wcae2003.html
4. Data Encryption Standard (DES) http://www.itl.nist.gov/fipspubs/fip46-2.htm
5. What is DES? http://www.thenextwave.com/page19.html
6. DES, an example http://www.mapleapps.com/powertools/cryptography/HTML/DES-Example.html
7. DES Key Expansion http://www.mapleapps.com/powertools/cryptography/HTML/DES-KeyExpansion.html
8. AES Encryption http://www.mapleapps.com/powertools/cryptography/HTML/AES-Encryption.html
9. http://web.stcloudstate.edu/sherath/SecurityLabs
10. Computer Architecture I http://web.stcloudstate.edu/jherath/CompArch-1
11. Computer Architecture II http://web.stcloudstate.edu/jherath/CompArch-2
12. http://web.stcloudstate.edu/sherath/osss2004