UOW MALAYSIA
KDU PENANG
UNIVERSITY COLLEGE

PART OF THE UNIVERSITY
OF WOLLONGONG AUSTRALIA
GLOBAL NETWORK

School of Engineering, Computing & Built Environment

Department of Computing

Diploma in Computer Studies & Diploma in Information Technology

COMPUTING MATHEMATICS (DCM1124)

September 2020 Semester

[LECTURER'S NAME: Dr. Ang Sau Loong]

[Assignment 1]

[STUDENT NUMBER & STUDENT NAME]

0205228	KHOO SOON FATT
0204677	LIM ZHE YUAN
0205300	LIM HUI PENG
0205096	THOR WEN ZHENG

DUE DATE : [30th October 2020]

TOTAL MARKS : [/ 15%]

DCM 1124 Computing Mathematics
Assignment 1 (15%)

Assignment 1: Group work (3-4 students in a group)

Due date for submission	Week 7
Marks	100% (Weighted marks: 15%)

Learning outcomes

CLO1	Evaluate the basic computer architecture and information representation using the Arabic system, binary, octal and hexadecimal.

Question 1:

Task 1 (Group Work):

Encryption is a process that encodes a message or file so that it can be only be read by certain people. Encryption uses an algorithm to scramble, or encrypt, data and then uses a key for the receiving party to unscramble, or decrypt, the information. Given that you are the chief intelligent officer who urgently need to convey a secretive message to your subordinates without being detected/tracked.

i) Encrypt the following sentence into Hexadecimals format. You are required to show the conversion process(es) using a table of conversion or a conversion tool.

The important key is hiding under the mat of my house. Please pick it up by tomorrow morning before it is too late. (30 marks)

ii) Comment on the safety of the Hexadecimal encryption. Justify your statements with solid reasons and appropriate examples. (20 marks)

Task 2 (Individual Work):

By using your creativity and innovative, find a proper way to improve the encryption method demonstrated in part (i) so that the sentence becomes more difficult to be decrypted. Write a short report to explain on the proposed method. You are requested to show the steps of the encryption and decryption.

TABLE OF CONTENTS

Answer

Task 1 (i)

Original message:

The important key is hiding under the mat of my house. Please pick it up by tomorrow morning before it is too late.

Encrypted message:

54686520696D706F7274616E74206B6579206973206869646696E6720756E6465722074686520 6D61
74206F66206D7920686F7573652E20506C65617365207069636B20697420757020627920746F6D6F
72726F77206D6F726E696E67206265666F726520697420697320746F6F206C6174652E

The way hexadecimal encryption is applied is that each character from the original message corresponds to their own hexadecimal code in the ASCII table. The first step in hexadecimal encryption is to identify the character to be encrypted, then look for the current character in the ASCII table, and refer to its hexadecimal character code. Once the hexadecimal code is correctly identified, the original character is substituted with the hexadecimal code. This process is repeated for every character in the original message until every character is encrypted into hexadecimal codes.

Below is the ASCII table that was used to encrypt the message into hexadecimal codes:

Decimal	Octal	Hex	Binary	Value	Description
000	000	00	0000 0000	NUL	"null" character
001	001	01	0000 0001	SOH	start of header
002	002	02	0000 0010	STX	start of text
003	003	03	0000 0011	ETX	end of text
004	004	04	0000 0100	EOT	end of transmission
005	005	05	0000 0101	ENQ	enquiry
006	006	06	0000 0110	ACK	acknowledgment
007	007	07	0000 0111	BEL	bell
008	010	08	0000 1000	BS	backspace
009	011	09	0000 1001	HT	horizontal tab
010	012	0A	0000 1010	LF	line feed
011	013	0B	0000 1011	VT	vertical tab
012	014	0C	0000 1100	FF	form feed
013	015	0D	0000 1101	CR	carriage return
014	016	0E	0000 1110	SO	shift out
015	017	0F	0000 1111	SI	shift in
016	020	10	0001 0000	DLE	data link escape
017	021	11	0001 0001	DC1	device control 1 (XON)
018	022	12	0001 0010	DC2	device control 2

019	023	13	0001 0011	DC3	device control 3 (XOFF)
020	024	14	0001 0100	DC4	device control 4
021	025	15	0001 0101	NAK	negative acknowledgeme nt
022	026	16	0001 0110	SYN	synchronous idle
023	027	17	0001 0111	ETB	end of transmission block
024	030	18	0001 1000	CAN	cancel
025	031	19	0001 1001	EM	end of medium
026	032	1A	0001 1010	SUB	substitute
027	033	1B	0001 1011	ESC	escape
028	034	1C	0001 1100	FS	file separator
029	035	1D	0001 1101	GS	group separator
030	036	1E	0001 1110	RS	request to send/record separator
031	037	1F	0001 1111	US	unit separator
032	040	20	0010 0000	SP	space
033	041	21	0010 0001	!	exclamation mark
034	042	22	0010 0010	"	double quote
035	043	23	0010 0011	#	number sign
036	044	24	0010 0100	$	dollar sign
037	045	25	0010 0101	%	percent
038	046	26	0010 0110	&	ampersand
039	047	27	0010 0111	'	single quote
040	050	28	0010 1000	(	left/opening parenthesis
041	051	29	0010 1001	)	right/closing parenthesis
042	052	2A	0010 1010	*	asterisk
043	053	2B	0010 1011	+	plus
044	054	2C	0010 1100	,	comma
045	055	2D	0010 1101	-	minus or dash
046	056	2E	0010 1110	.	dot
047	057	2F	0010 1111	/	forward slash
048	060	30	0011 0000	0	
049	061	31	0011 0001	1	
050	062	32	0011 0010	2	
051	063	33	0011 0011	3	
052	064	34	0011 0100	4	
053	065	35	0011 0101	5	
054	066	36	0011 0110	6	
055	067	37	0011 0111	7	
056	070	38	0011 1000	8	

057	071	39	0011 1001	9	
058	072	3A	0011 1010	:	colon
059	073	3B	0011 1011	;	semi-colon
060	074	3C	0011 1100	<	less than
061	075	3D	0011 1101	=	equal sign
062	076	3E	0011 1110	>	greater than
063	077	3F	0011 1111	?	question mark
064	100	40	0100 0000	@	"at" symbol
065	101	41	0100 0001	A	
066	102	42	0100 0010	B	
067	103	43	0100 0011	C	
068	104	44	0100 0100	D	
069	105	45	0100 0101	E	
070	106	46	0100 0110	F	
071	107	47	0100 0111	G	
072	110	48	0100 1000	H	
073	111	49	0100 1001	I	
074	112	4A	0100 1010	J	
075	113	4B	0100 1011	K	
076	114	4C	0100 1100	L	
077	115	4D	0100 1101	M	
078	116	4E	0100 1110	N	
079	117	4F	0100 1111	O	
080	120	50	0101 0000	P	
081	121	51	0101 0001	Q	
082	122	52	0101 0010	R	
083	123	53	0101 0011	S	
084	124	54	0101 0100	T	
085	125	55	0101 0101	U	
086	126	56	0101 0110	V	
087	127	57	0101 0111	W	
088	130	58	0101 1000	X	
089	131	59	0101 1001	Y	
090	132	5A	0101 1010	Z	
091	133	5B	0101 1011	[	left/opening bracket
092	134	5C	0101 1100	\	back slash
093	135	5D	0101 1101	]	right/closing bracket
094	136	5E	0101 1110	^	caret/circumflex
095	137	5F	0101 1111	_	underscore
096	140	60	0110 0000	`	
097	141	61	0110 0001	a	

098	142	62	0110 0010	b	
099	143	63	0110 0011	c	
100	144	64	0110 0100	d	
101	145	65	0110 0101	e	
102	146	66	0110 0110	f	
103	147	67	0110 0111	g	
104	150	68	0110 1000	h	
105	151	69	0110 1001	i	
106	152	6A	0110 1010	j	
107	153	6B	0110 1011	k	
108	154	6C	0110 1100	l	
109	155	6D	0110 1101	m	
110	156	6E	0110 1110	n	
111	157	6F	0110 1111	o	
112	160	70	0111 0000	p	
113	161	71	0111 0001	q	
114	162	72	0111 0010	r	
115	163	73	0111 0011	s	
116	164	74	0111 0100	t	
117	165	75	0111 0101	u	
118	166	76	0111 0110	v	
119	167	77	0111 0111	w	
120	170	78	0111 1000	x	
121	171	79	0111 1001	y	
122	172	7A	0111 1010	z	
123	173	7B	0111 1011	{	left/opening brace
124	174	7C	0111 1100	|	vertical bar
125	175	7D	0111 1101	}	right/closing brace
126	176	7E	0111 1110	~	tilde
127	177	7F	0111 1111	DEL	delete

Task 1 (ii)

Hexadecimal encryption is insecure.

One of the reasons why hexadecimal encryption is not safe is that it is a straightforward, one-step process. Using hexadecimal encryption, all one needs to do is identify each hexadecimal character code and refer to the ASCII table, which is publicly available to everyone on the Internet, to determine its ASCII character. Then, they repeat this straightforward and almost effortless process for each and every hexadecimal code until the whole message is fully decrypted. They could also just use an online conversion tool to instantly decrypt the whole message. Thus, it does not take much time or effort to decrypt the encrypted message. Generally, when encrypting confidential information, we want it to take as much time and effort as possible for the decryption process. The main purpose of that is to discourage any potential attackers from trying to decrypt the message by making them think that it is not worth their time and effort. Hence, the more time and effort it takes, the more secure the encryption method will be. Imagine trying to brute force every single value of each character in the original message that it can possibly have, but the intruder still ends up having multiple versions of the answer and all of them are possibly the actual message. They have to spend time and effort figuring out which version of the answer makes the most sense, knowing fully that if they make even one small mistake, the information could be completely misread and the operation will end in failure.

Another reason is that the hexadecimal encryption method itself is easily identifiable. Naturally, when we encrypt confidential information, we want to use an encryption method that is not easy to identify. Forcing any potential attacker to spend extra time just to identify the encryption method used is a huge plus for the sender of the message. In a way, this adds to the difficulty level of the encryption method that is used and could help discourage potential attackers from attempting to decrypt confidential information. The hexadecimal encryption method, however, is very easy to identify at a glance, as stated before. Anyone with basic mathematical knowledge of number bases can easily identify the alphanumeric hexadecimal codes all throughout the message, which are obvious clues that the message was encrypted using the hexadecimal encryption method. If attackers can identify the encryption method, they can of course, research its process and how it works in order to decrypt the message.

Since hexadecimal encryption is considered an easier encryption approach, the chances are higher for intruders to try this method first as a mean of decryption compared to other decryption methods. Hexadecimal encryption is one of the few fundamental concepts that will be covered in basic cryptology. We, as first year undergraduates, were also taught about the concept of number bases early on so we can understand hexadecimal encryption. What stops the outside attackers from thinking that the encryption might be hexadecimal encryption and try to decrypt an encrypted message using it? Besides, intruders will try to work their way up, starting with an easier decryption method to try decrypting the message and progressively increase the complexity of a decryption method if the previous decryption method does not work. Logically, if they attempt to do it that way, they can save a lot of time and energy trying to decrypt the code if the encryption method used is very simple. Therefore, if they tried hexadecimal decryption ahead of any other decryption methods available out there in the world, they would decrypt the code sooner than expected. This also means that hexadecimal encryption doesn’t provide even the slightest amount of protection, making it one of the worst encryption methods to be used for encrypting confidential information.

Hexadecimal encryption is also unsafe because it does not utilize encryption keys to perform mathematical or logical operations. An encryption key is a string of bits used to determine the output of a cryptographic encryption algorithm that scrambles or alters the form of data. Usually, the pattern of the encrypted data appears to be vague and random because the key is applied in the encryption algorithm mathematically or logically. The end results are randomized, and each character are different with one another depending on the equation used. Some secure encryption algorithms use a one-time pad so that the key used to encrypt a message is at least the same size as the message, generated truly randomly and kept in secrecy. Only the sender and the receiver will be able to decrypt the message using the randomly generated key that acts like their password. However, hexadecimal encryption does not use any key to perform any logical or mathematical operations to encrypt a message. It only uses a straightforward substitution method to represent data in another form. The alphanumeric pattern of hexadecimal encryption is already a huge giveaway, but if the outside attackers wanted more solid clues about the usage of hexadecimal encryption, they can also just convert the encrypted message into other formats like decimals, octal and binary and try to decrypt them using their corresponding values in the ASCII table. If their decryption output matches hexadecimal's decryption output, then it is certain that the decrypted message is the original message that they are supposed to decrypt for. The values of each character in the original message are not actually randomized and can be easily decrypted if the attackers are knowledgeable about number bases.

Task 2

Applying Simpler Model of AES to Improve Hexadecimal Encryption

by LIM ZHE YUAN

My idea on improving the encryption method used in part (i), which is hexadecimal encryption, is based on a better encryption method available today. It is my simplified version of the Advanced Encryption Standard (AES) using 128-bit key, or alternatively known as the 'Rijndael Cipher' (Paar and Pelzl, 2009, 89).

Using hexadecimal encryption, the original text of the encrypted code will still be readable after decrypting the code as each hexadecimal value do not move from their original position. To be frank, hexadecimal codes can be decrypted very easily by using any online converter available in the web. Therefore, to solve this issue, my idea is to use byte permutation to increase the difficulty of decrypting the code, alongside other substitution techniques. Byte permutation is a process of rearranging the bytes in a message so that each byte that represent a printable character are scrambled randomly at different locations (Paar and Pelzl, 2009, 90).

Specifically, my idea is an encryption process that repeats for 10 rounds. To start off, the original message is split into 128-bit blocks. In each block, each hexadecimal pair that form a byte are filled inside a 4 by 4 table in an orderly manner to represent 1 character. In each round of encryption, the encryption process will start by substituting each byte with a substitute value using a substitution box called the Rijndael forward S-box. Then, to implement the main solution of byte permutation, each byte in the 4 by 4 block are shifted first by rows and then by columns following these set of rules: For the nth row, the values of the row are shifted n-1 to the left. After the row shifting process is done, for the nth column, the values of the column are shifted n-1 downwards. Before the end of an encryption round, a secret and randomly-generated key that is also the same size of the 128-bit block will also be used to perform bitwise XOR operations to each byte in the block to produce randomized results. This also adds decryption difficulty for outside attackers. XOR operation is an operation whereas if two Boolean values are the same, it returns false and if two Boolean values are different, it returns false (Solomon and Kim, 2011, 447). To perform the operation, the generated key is also split into a 4 by 4 format so that each block field of the key corresponds to the block field of the code. After 10 rounds of encryption, each 128-bit block is concatenated back to form a strongly encrypted message (Paar and Pelzl, 2009, 99-106).

Obviously, to decrypt the encrypted message, we must do the exact opposite of the encryption process. The decryption process also repeats for 10 rounds. But for each round, the round starts by inversing the XOR operation of round key, followed by inversing the column shifting process, then inversing the row shifting process, and lastly inversing the byte substitution process. The decryption process uses the same 4*4 block method like in the encryption process. Firstly, the XOR operation of round key is inversed to find the original byte that is XORed with the round key that produced the byte we will be inverse XORing. This process is done by XORing the round key with the encrypted code again byte by byte. This works as XOR operations are bidirectional and allow byte values to be reversible, meaning the original byte can be obtained by XORing its corresponding encrypted byte again (Salomon, 2007, 166-167). After that, we will inverse shift the columns to determine the original position of the bytes vertically before the shift in the encryption process. Similar to the encryption process, for the nth column, the bytes in the column are shifted n-1 upwards instead of downwards. Then, we will proceed to shift the rows inversely to determine the original position of bytes horizontally before the shift in the encryption process. Similar to the encryption process, for the nth row, the bytes in the row are shifted n-1 to the right, instead of shifting them to the left. Finally, we will inverse the byte substitution process by using another substitution box that is the opposite of the Rijndael forward S-box, called the Rijndael inverse S-box. As you would've guessed, this substitution box substitutes each byte that has gone through encryption back to their original value, that is before they are substituted with another value using Rijndael forward S-box. After 10 rounds of the whole decryption process, the original message will be revealed after joining each block of deciphered message back together (Paar and Pelzl, 2009, 110-114).

I consider this encryption method better than hexadecimal encryption because it uses a lot of substitution and permutation methods to distort the values and make it so that it looks nothing like it originally was before. The large number of rounds used for the encryption process also confuse outside attackers and consumes a lot of their time and effort when trying to crack the code. As we said before, the best way to proof that an encryption is secure is to find out if it will demotivate outside attackers on trying to decrypt the code. They will feel exactly that way when they will still need to work with a bunch of dummy values even after attempting so many times. Another interesting fact is that the usage of XOR operations produce results that does not provide any clues of the original byte values or the round key itself. This is because there is always a 50% chance for a bit in the original byte value or the round key to be either 0 or 1. For example, let's say we XOR a bit from the original byte value and the corresponding bit of the same position from the round key together that outputs to 1. Outside attackers can only imagine that either both input bits are 0 or both are 1 for the XOR operation to output 1. This increases the number of attempts needed for the outside attackers to be able to crack the encrypted code twofold, leaving them begging for mercy.

Utilizing Addition and Subtraction to Increase the Security of Hexadecimal Encryption

by: THOR WEN ZHENG

Introduction

My idea for improving hexadecimal encryption is to add additional steps into the encryption process to make the message more difficult to decrypt, instead of just directly converting each character into hexadecimal form based on the ASCII table. For the additional steps, we will be referring to the decimal character code of each character in the message based on the ASCII table. After we identify the decimal character code, we determine whether the character code is an even number or odd number. For even numbers, we will add x to the decimal value of the character code, where x is a secret encryption key which is a secret number specified by the sender of the message, and it is only known by the sender and the receiver of the message. On the other hand, for odd numbers, we will subtract x from the decimal value of the character code. Finally, we convert the decimal character code into the hexadecimal equivalent by manual conversion or by referring to the ASCII table.

Essentially, we are performing a combination of addition and subtraction to the even and odd decimal character codes respectively, before converting them to hexadecimal codes. This increases the difficulty to decrypt the message because the characters in the message are not just directly substituted with their corresponding hexadecimal codes based on the ASCII table. If an attacker tries to decrypt the message by directly converting every hexadecimal code into characters, what they will get is a messy sentence with scrambled letters that does not make sense. They will have to spend extra time and effort to figure out the correct way to decrypt the message.

Encryption

In the encryption process, the first step is to identify the current character in the message to be encrypted. Next, refer to the ASCII table to look for the character in the range of printable characters and determine its decimal character code. Then, determine whether the decimal character code is an even number or an odd number. The next step involves addition and subtraction using the secret number which is the encryption key specified by the sender of the message and is only known by the sender and the receiver. If the decimal character code is an even number, add the value of the secret number to the decimal character code. If it is an odd number, subtract the decimal character code by the value of the secret number. Then, the final step is to substitute the decimal character code with its corresponding hexadecimal code based on the ASCII table. This process is repeated for every single character in the message until every one of them is encrypted.

To make the steps clearer, the encryption process is demonstrated with an example. In this example, the word to be encrypted is “The” and the secret number that is used is 5. The steps to encrypt this word are as follows:

1. The first character to be encrypted is the capital letter “T”. Based on the ASCII table, the decimal character code of “T” is 84, and 84 is an even number. So, apply addition by adding 5 to 84, the result is 89. Next, convert 89 to its hexadecimal equivalent by doing manual conversion or refer to the ASCII table, so 89 will be converted to 59.
2. The next character to encrypt is the lowercase letter “h”, its decimal character code is 104, which is an even number. Hence, the same process done for the letter “T” can be repeated for the letter “h”, and “h” will be substituted with 6D.
3. The final character to be encrypted is the letter “e”, its decimal character code is 101, which is an odd number. So, apply subtraction by subtracting 5 from 101, the result is 96. Next, convert 96 to its hexadecimal equivalent, so 96 will be converted to 60.
4. The final encrypted version of the word “The” is “596D60”.

Since the decimal character codes of the ASCII printable characters range from 32 to 126, the process is slightly different if the character code goes outside of this range after addition or subtraction. If the character code undergoes subtraction and results in a number less than 32, the subtraction should continue from the end of the range, which is 126, before the number goes below 32. For example, if the character to be encrypted is "!", its decimal character code is 33. This is an odd number so it will undergo subtraction. Assuming that the secret number that is being used is 5, 5 is subtracted from the character code, the resulting number would be 28. However, 28 is outside the range of the character codes of the ASCII printable characters. Thus, the maximum number it can be subtracted by before going below 32 is 1. After it is subtracted by 1, the resulting number is 32 and it still needs to be subtracted by 4. Hence, when 32 is subtracted by 1 again, the resulting number will be 126. Then, continue subtracting the number by the remaining value to be subtracted, which is 3. The final decimal character code should be 123.

If the character code undergoes addition and results in a number greater than 126 the addition should continue from the start of the range, which is 32, before the number goes beyond 126. For example, if the character to be encrypted is "}", its decimal character code is 125. This is an even number so it will undergo addition. In the event that 5 is added to its character code, the resulting number would be 130. However, 130 is outside the range of the character codes of the ASCII printable characters. Thus, the maximum number that can be added to 125 is 1, and the resulting number is 126. Next, add 1 to 126, the resulting number should be 32. The remaining value to be added is 3, so add 3 to 32 and the final decimal character code will be 35.

Furthermore, using a larger-sized encryption key would also be able to increase the security of the encryption method. In this case, a larger-sized encryption key means a larger secret number used in the encryption. By using a larger secret number, the encrypted characters would be farther away from their original character in the ASCII table. This should make the key as well as the encryption method less obvious and less easy to guess.

Decryption

The decryption process will simply be a reversed process of the encryption. However, the decryption process will have added difficulty as the receiver of the message would not know which characters initially had even or odd decimal character codes, or whether addition or subtraction was applied before encryption. The receiver would first have to identify the first hexadecimal code, then refer to the ASCII table or do manual conversion to determine the decimal code. Next, perform both addition and subtraction for the identified decimal code using the secret number, and subsequently convert both resulting decimal numbers into characters based on the ASCII table. The receiver should then record the 2 possible characters after decryption of the first hexadecimal code, then repeat the decryption process for the subsequent hexadecimal codes and record the possible characters along the way. At a point where the receiver feels that he or she has acquired enough characters, or after converting all hexadecimal codes into possible characters, the receiver should use the recorded possible characters to piece together actual, proper and intelligible words to form the full decrypted message.

To make the decryption process clearer, it is demonstrated with an example. The example used is the same as the one used in the encryption process, the message to decrypt is “596D60” and the secret number is 5. The steps to decrypt this message are as follows:

1. The first hexadecimal code to decrypt is 59. First, convert 59 to its decimal equivalent which is 89.
2. Next, from the receiver’s perspective, it is not known whether the decimal character code of the original character is even or odd. Thus, both addition and subtraction need to be performed to determine the possible characters. The 2 resulting decimal numbers are 84 and 94, which means the 2 possible characters are “T” and “^”.
3. The same process done in steps 1 and 2 is repeated for the remaining hexadecimal codes, which are “6D” and “60”. The possible characters for “6D” are “h” and “r”, while the possible characters for “60” are “[” and “e”.
4. Finally, the pairs of possible characters acquired are:
   - T or ^
   - h or r
   - [or e

   Select one character from each pair to form a proper word. Now, by combining the characters together, it is possible to determine that the word is “The”.

The example used decrypts a very short and simple word, which is why it does not take much time or effort to decrypt. However, to decrypt an entire encrypted message is a different story. The encrypted message may have several sentences, which may contain many words, including simple and complicated words. The longer the message, the longer the time and the more effort required to decrypt the whole message. Furthermore, an attacker may not necessarily be able to figure out the exact encryption method and the encryption key that was used.

Conclusion

To summarize, by applying the concept of even and odd numbers, addition, and subtraction in the process of hexadecimal encryption, we are able to increase the difficulty of decrypting the encrypted message. To further prove that statement, let’s use some example situations. If we use the original straightforward hexadecimal encryption a message, the attacker may be able to identify the hexadecimal codes at a glance. The first method the attacker would probably try is to simply convert the hexadecimal codes into characters based on the ASCII table. The attacker could also just copy the encrypted message and paste it into an online conversion tool to decrypt the message. Just like that, he easily decrypted the message within a short time and with little to no effort.

However, if we apply additional steps into the hexadecimal encryption process, we can at least complicate the decryption process to some degree. Using the idea that I proposed, an attacker would get a messy message with random scrambled letters if he or she simply converted the hexadecimal codes into characters based on the ASCII table. The attacker would then have to spend extra time and effort to figure out what exact steps were added to complicate the decryption process. This would hopefully be able to discourage the attacker from trying to decrypt the message as it may not be worth it.

In conclusion, most attackers with the intent of stealing or peeking on private and confidential information would most likely be able to identify hexadecimal codes effortlessly. Therefore, if we really want to use hexadecimal encryption, we should at least add some additional steps in the encryption process. Of course, it is highly probable that an attacker will figure out the steps that I suggested to add in the encryption process. But by adding those steps, the decryption process will be slightly more complicated so that it is not too easy and straightforward for an attacker to decrypt the message.

<u>Session key (Content encryption key, CEK)to improve Hexadecimal Encryption</u>

by: LIM HUI PENG

Introduction

Based on Task 1 part (ii), the information show that Hexadecimal encryption method are not secure because it can use the tools that available on web to convert encrypt data easily. Therefore, the encryption sentence on Task 1 part (i) can be improve like the information that show at below:

Sentence part (i)

The important key is hiding under the mat of my house. Please pick it up by tomorrow morning before it is too late.

Improved message:

```
TishP%enq ASinkm5ip./o$srIHt06aglnWqt*) UrkXTe!Ey91 Cvi-ts!= @bhM4ikXd9^i%>nJOgnd
Llu<,nwMd{ee&TrF3 ^\t6qh:}eX` #:mlNa/?t6! (6oNWfk& xBm%~y0z *\hqCo7^u.Ys:/eim.-
/J2Pvul7;eRga\|s0wecK .=pAqiQ>c</k8z ?!i#@tHj 4fuzEphb >/bOEy*& -
|t=io79mWYo(.r#3r\\o!~w>G xPm%?o*Zr:pnoOi#`n0sg.; Bmb/De+\f|roApr~)e!L #^iuUt9j C?izQs,;
6/tNRo0Ko\& #%l:`ajit{pe}T.?2
```

Explanation on Proposed Method

The idea that use to improve the encryption sentence above is add two useless letter behind every important letter, such as " The " in the sentence will be write in " TishP%enq ". The last number letter of the sentence is the hidden key for encryption. For example, the last letter of the sentence "2" play as hidden key. It means two letters behind every important letter is useless and no need to encrypt. The recipient should know the meaning or hint of the last letter in advance. Even the attacker use the ASCII code correctly, they still get the improved sentence after decrypt because they don't know the meaning of "2" or notice that "2" is the important key of the sentence. Then they will think the sentence does not have meanings or the decrypt method that they use is incorrect. Hence, they will try other decrypt method such as Binary, Octal and Decimal but still get the wrong code so they cannot recover the original message. For more information, the steps of encryption and decryption will be show at below.

Steps of encryption

Based on the information above, the idea of improved sentence is add up two useless letter behind every important letter and the letter "2" must be written at the end of the sentence. The first step is using Hexadecimal encryption to encrypt the whole message. All SPACE in the sentence need to be encrypted.

Improved sentence:

TishP%enq ASinkm5ip./o$srIHt06aglnWqt*) UrkXTe!Ey91 Cvi-ts!= @bhM4ikXd9^i%>nJOgnd Llu<,nwMd{ee&TrF3 ^\t6qh:}eX` #:mlNa/?t6! (6oNWfk& xBm%~y0z *\hqCo7^u.Ys:/eim.-/ J2Pvul7;eRga\|s0wecK .=pAqiQ>c</k8z ?!i#@tHj 4fuzEphb >/bOEy*& - |t=io79mWYo(.r#3r\\o!~w>G xPm%?o*Zr:pnoOi#`n0sg.; Bmb/De+\f|roApr~)e!L #^iuUt9j C?izQs,; 6/tNRo0Ko\& #%l:`ajit{pe}T.?2

Encrypted message:

546973685025656E71204153696E6B6D3569702E2F6F247372494874303661676C6E5771742A292
055726B585465214579393120437669 2D7473213D204062684D34696B5864395E69253E6E4A4F67
6E64204C6C753C2C6E774D647B65652654724633205E5C743671683A7D65586020233A6D6C4E
612F3F743621202836 6F4E57666B262078426D257E79307A202A5C6871436F375E752E59733A2F
65696D2E2D2F204A325076756C373B655267615C7C73307765634B202E3D70417169513E633C2
F6B387A203F2169234074486A203466757A45706862203E2F624F45792A26202D7C743D696F373
96D57596F282E722333725C5C6F217E773E472078506D253F6F2A5A723A706E6F4F6923606E30
73672E3B20426D622F44652B5C667C726F4170727E2965214C20235E69755574396A20433F697A
51732C3B20362F744E526F304B6F5C262023256C3A60616A69747B70657D542E3F32

Steps of decryption

First step, as the recipient know the method that use to encrypted so they start to decrypt the last code which is "32" and they will get the number "2" (use hexadecimal encryption). After they get the number "2", they will start to keep the first code (important code) and throw away the two codes behind (useless codes), repeat the method until the end of the sentence. The important code will be show in red colour and useless codes is in blue colour on above. They will get the encrypted message after take out the useless codes. At last, decrypt the message using the ASCII table that show in Task 1 (i).

Encrypted message after take out useless codes:
546865206 96D706F7274616E74206B657920697320686964696E6720756E646572207468652 06D61
74206F66206D7920686F7573652E20506C65617365207069636B2069742075702062792074 6F6D6F
72726F77206D6F726E696E67206265666F72652069742069732074 6F6F206C6174652E

ASCII Table (The codes used in message):

HEX	Symbol
54	T
50	P
61	a
62	b
63	c
64	d
65	e
66	f
67	g
68	h
69	i
6B	k
6C	l

HEX	Symbol
6D	m
6E	n
6F	o
70	p
72	r
73	s
74	t
75	u
77	w
79	y
2E	.
20	

Decrypted message:
The important key is hiding under the mat of my house. Please pick it up by tomorrow morning before it is too late.

“The Simpler They Are The Harder They Crack”

By Using The Caesar Cipher Encryption To Increase the Security of Hexadecimal Encryption

By : KHOO SOON FATT

Introduction

My idea for improving hexadecimal encryption is to rearrange every hexadecimal value from the encryption and not just that, I also would use a random generated hexadecimal value and added to the hexadecimal encryption to make the message more difficult to decrypt, instead of just converting each other from the ASCII. It is somewhat similar to byte permutation where you have to shift the encryption.

Basically, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. But in this case I’m gonna use it to replaced a hexadecimal position by another hexadecimal position some fixed number of position down below.

Encryption and Decryption

By using the Caesar cipher is too easy to be decrypt, so in order to make it harder, First we will convert the original text :

“ The important key is hiding under the mat of my house. Please pick it up by tomorrow morning before it is too late ”

Then convert into hexadecimal encryption. Like so :

54 68 65 20 69 6d 70 6f 72 74 61 6e 74 20 6b 65 79 20 69 73 20 68 69 64 69 6e 67 20 75 6e 64 65 72
20 74 68 65 20 6d 61 74 20 6f 66 20 6d 79 20 68 6f 75 73 65 2e 20 50 6c 65 61 73 65 20 70 69 63 6b
20 69 74 20 75 70 20 62 79 20 74 6f 6d 6f 72 72 6f 77 20 6d 6f 72 6e 69 6e 67 20 62 65 66 6f 72 65
20 69 74 20 69 73 20 74 6f 6f 20 6c 61 74 65

Then take the second position of the hexadecimal and placed it in the back

54 68 65 20 69 6d 70 6f 72 74 61 6e 74 20 6b 65 79 20 69 73 20 68 69 64 69 6e 67 20 75 6e 64 65 72
20 74 68 65 20 6d 61 74 20 6f 66 20 6d 79 20 68 6f 75 73 65 2e 20 50 6c 65 61 73 65 20 70 69 63 6b
20 69 74 20 75 70 20 62 79 20 74 6f 6d 6f 72 72 6f 77 20 6d 6f 72 6e 69 6e 67 20 62 65 66 6f 72 65
20 69 74 20 69 73 20 74 6f 6f 20 6c 61 74 65

So the result would end up like this :

56 62 66 76 77 66 72 66 72 67 26 66 66 62 76 66 72 76 62 66 72 66 26 72 66 77 62 25 66 67 62 76 66 26 72 77 26 72 76 66 77 67 26 67 66 66 26 66 67 62 67 26 72 76 62 66 76 48 50 9d 0f 24 1e 40 b5 90 93 08 94 9e 70 5e 45 20 48 50 d1 40 f6 0d 90 8f 53 5e 00 c5 13 50 09 3b 09 40 50 02 90 4f df 22 f7 0d f2 e9 e7 02 56 f2 50 94 09 30 4f f0 c1 45

And do this about 4 time then to add a bit more layer of security I would generate a random number key code and input in the hexadecimal encryption. The random number means nothing and it will be different then the rest and only the person have the key code can subtract the random numbers. For examples :

56 88 62 66 88 76 77 88 66 72 6f 66 72 67 26 66 66 62 76 88 66 72 76 62 66 72 66 26 6f 72 66 77 6f 62 25 66 67 62 6f 76 66 26 72 77 26 72 88 76 66 77 67 26 67 66 66 88 26 66 67 62 67 26 72 76 62 66 76 48 50 9d 0f 24 88 1e 40 b5 90 93 08 94 9e 88 70 5e 45 20 48 50 d1 40 f6 0d 90 8f 53 5e 00 88 c5 13 50 09 3b 09 40 50 02 88 90 4f df 22 f7 6f 0d f2 e9 e7 02 88 56 f2 50 94 09 30 4f f0 c1 45

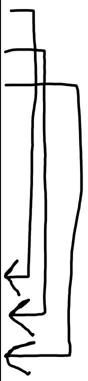
Key : 88, 6f, etc

Finally we will use the Caesar cipher to further encrypt the message. In this case we want the cypher to be replace in the eight fixed position down below.

For examples :

ASCII Table (The codes used in message):

HEX	Symbol
41	A
42	B
43	C
44	D
45	E
46	F
47	G
48	H
49	I
4A	J
4B	K
4C	L
4D	M



HEX	Symbol
4E	N
4F	O
50	P
51	Q
52	R
53	S
54	T
55	U
56	V
57	W
58	X
59	Y

In the end you will get a complete randomized number and they can’t crack the code without subtracting the key code.

Citation and Referencing

Miller, R. and Kasparian, R. (2006) *Java for Artists: The Art, Philosophy, and Science of Object-oriented Programming*. Virginia: Pulp Free Press.

Bauer, F.L. (2000) *Decrypted Secrets*, 2nd edition. Berlin: Springer.

Kahn, D. (1996) *The Codebreakers: The Story of Secret Writing*, revised edition. New York: Simon and Schuster.

Salomon, D. (2007) *Variable-length Codes for Data Compression*. London: Springer Science & Business Media.

Solomon, G. M. and Kim, D. (2011) *Fundamentals of Communications and Networking*. Sudbury: Jones & Bartlett Publishers.

Paar, C. and Pelzl, J (2009) *Understanding Cryptography*. Berlin: Springer Science & Business Media.

Bhanu, N. and Chaitanya, N.V. (2019) Aes Modes of Operations. *International Journal of Innovative Technology and Exploring Engineering*, 8(9) 2213-2217. Available from https://www.ijitee.org/wp-content/uploads/papers/v8i9/I8127078919.pdf [accessed 25 October 2020].

NIST (2001) *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. FIPS 197. Springfield, Virginia: NIST. Available from https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf [accessed 25 October 2020].

Rijmenants, D. (2018) *The Complete Guide to Secure Communications with the One Time Pad Cipher*. Available from http://users.telenet.be/d.rijmenants/papers/one_time_pad.pdf [accessed 28 October 2020].

Techopedia (2017) *Encryption Key*. Techopedia. Available from https://www.techopedia.com/definition/25403/encryption-key [accessed 28 October 2020].

Cloudflare (2020) *What is a cryptographic key?*. Cloudflare. Available from https://www.cloudflare.com/learning/ssl/what-is-a-cryptographic-key/ [accessed 28 October 2020].

O'Dwyer, M. (2019) *Double Encryption: Is It More Secure Or Dangerous?* [blog]. 8 July. Available from https://blog.ipswitch.com/double-encryption-is-it-more-secure-or-dangerous [accessed 23 October 2020].

Wagner, L. (2020) *Reasons Why XOR is Important in Cryptography* [blog]. 20 January. Available from https://hackernoon.com/reasons-why-xor-is-important-in-cryptography-6tcn32yx [accessed 23 October 2020].

Shad sluiter (2019) *How does AES encryption work? Advanced Encryption Standard* [video]. Available from https://www.youtube.com/watch?v=lnKPoWZnNNM [accessed 24 October 2020].

Savarese, C. and Hart, B. (1999) *The Caesar Cipher*. Available from http://www.cs.trincoll.edu/~crypto/historical/caesar.html [accessed 29 October 2020].

Marking Rubric

[CBC3033 Business Statistic] MARKING RUBRIC ASSIGNMENT [1]

Report Writing (15%)

LEARNING OUTCOME	MARKING CRITERIA	SCALE					
		Fail (0-49)	**3rd Class (50-59)**	**2nd Lower Class (60-69)**	**2nd Upper Class (70-79)**	**1st Class (80-100)**	**YOUR MARKS/COMMENTS**
CLO1: Evaluate the basic computer architecture and information representation using the Arabic system, binary, octal and hexadecimal.	**Task 1: Group Work i)Quality of explanation on the conversion. (40%)**	Fails to provide a satisfactory level of explanation on the methods.	Explanation level is satisfactory but lack of facts, table and figures to support the explanation on the conversion methods.	Explanation level is somehow good with some facts, tables and figures to support the explanation on the conversion methods.	Explanation is fairly well performed in describing the methods with strong support ample support facts, tables and figures.	The explanation is comprehensive and cover all processes taken in the conversion method. Good description with solid facts, tables and figures.	
	ii) Comments on the safety of conversion with justifications. (20%)	Comments provided has little or nothing to do with the question.	Weak comments which is not clear to the main topic, but lack of justifications.	Good comments to the main topics with some justifications.	Most of the descriptions comments are logic and can be accepted based on the justifications given.	Very outstanding comments on the safety of conversion with strong facts and justifications.	
	Task 2: Group Work The proposed method to improve safety of conversion (40%)	The proposed method does not meet the criteria for the assignment.	The proposed method requires clarification and/or are off-topic or have marginal relevance to the assignment.	The proposed method is stated clearly and is related to the topic, with only adequate explanations.	The proposed matter is presented clearly and is related to the topic, with only minor errors.	The proposed is clear, practical and relevant, with no errors – very professional.	
	Citation and Referencing (10%)	Missing or no citation and major flaws on the format. Reference section is missing.	Very minimal amount of cited works, with incorrect format. Improper reference section.	Adequate amount cited works, both text and visual, are done in the correct format. Inconsistencies evident. Reference section with minor flaws.	All, both text and visual, are done with minimal errors on the format. Reference section is in minimal error.	All cited works, both text and visual, are done in the correct format with no errors. Reference section properly formatted.	
	Total (100%)						