Lab #6

Out : November 6, 1997; 2:30 pm Due : November 13, 1997, 12:30 pm

Calculator Help: The MarkCalc can calculate modular multiplicative inverses. To calculate the multiplicative inverse of d, put "1/d" in the Command field and put the appropriate modulus in the Mod field.

Experiment 1:

The Pie of the Month Club uses the exponentiation cypher to encrypt each month's recipe so that those who are not in the club will not be able to learn the super-tasty secrets they have to offer. Being a fan of pie, you sign on. Each month a different pie recipe is sent out encrypted with the exponentiation cypher. The modulus the Pie Club uses is 477500508660178935571, which is prime. In this experiment you will receive just the name of the pie for this month.

When you open this experiment, you will see one calculator and three cells. One cell contains the cyphertext (a single block). One cell contains your key. The third cell is a hand-in transcript. You are also provided with a calculator. Receive the key and the cyphertext, and then decrypt the message. Send the plaintext to the hand-in transcript, and submit. Remember that in order to decrypt, you raise the cyphertext to the power of the mod $\phi(m)$ multiplicative inverse of the key.

In Brief							
We give you:	A cell containing the key, a cell containing the block of						
	cyphertext to decrypt, a handin cell, and a calculator for						
	you						
Your task:	Decrypt the cypertext						
Hand in:	Submit electronically when the handin cell contains the						
	$\operatorname{plaintext}$						

Experiment 2:

The Dog of the Month Club sees the success that the Pie of the Month Club had using the exponentiation cypher and decide to adopt their encryption scheme. They send out tips on breeding and care of various expensive breeds of dog. In this experiment, you will be receiving a message encrypted using ECB mode. For this problem the modulus is 88245335319593, a prime.

When you open this experiment, you will see a cell containing the key, a transcript containing the cyphertext, and a handin transcript. You are also provided with a calculator. You will decrypt the message and send it to the handin transcript. Submit electronically when you have successfully done so.



Figure 1: Experiment 1

	C	alcu	lator			
	•					
7		8	9	Ŧ	test	
4		5	6	<u> </u>	composite	
\vdash		2	3	L*	=	
ge	I (-			2	<u> </u>
H	rand primegen if		if f	alse		
F	send receive New Ri		Row			
Do	it S	Star	t Ma	any S	Stop	Reset
	Name Command Mod					



	Calcu	ilator			[
0					Key 27223		
7	8	9	+	test			
4	5	6	-	composite			
1	2	3	*	= ±	Cypnertext 93727 82049 12343		
gcd	0	^		2 5	Reset Clear Num New		
ra	rand primegen if true			if true			
sk	skip panic if false		if false				
se	end receive New Row			New Row			
Do it Start Many Stop Reset					Handin		
Name Command Mod Reset Clear Symbol New							

Figure 3: Experiment 3

In Brief			
We give you:	A cell containing the key, a transcript containing the blocks		
	of cyphertext, a handin transcript, and a calculator for you		
Your task:	Decrypt the message (ECB mode)		
Hand in:	Submit the plaintext electronically		

Experiment 3:

The Gadget of the Month Club has also decided to begin using the exponentiation cypher to mail product announcements to their members. However, they have chosen to use cypher block chaining (CBC mode) with no I.V. and a modulus of 88245335319593, a prime.

When you open this experiment, you will see a cell containing the key, a transcript containing the cyphertext, and a handin transcript. You are also provided with a calculator. You will decrypt the message and send it to the handin transcript. Submit electronically when you have successfully done so. Remember that this message was encrypted using CBC mode.

In Brief				
We give you:	A cell containing the key, a transcript containing the blocks			
	of cyphertext, a handin transcript, and a calculator for you			
Your task:	Decrypt the message (CBC mode with no I.V.)			
Hand in:	Submit the plaintext electronically			

Experiment 4:

In this experiment, you will assume the role of Alice and perform a series of tasks using Kerberos. Specifically, you will retrieve "email" from a calculator acting as the mail server and then send that email text to a calculator acting as the print server. In each of these exchanges, you will use Kerberos tickets to authenticate yourself to the servers and arrange for a session key with which the actual data (in this case, Alice's email) will be encrypted. Figure 4 illustrates this protocol.

The Kerberos server will be represented by a calculator. Tickets will consist of a series of 10-digit blocks, and encryption will be done using the exponentiation cypher in CBC mode with no I.V. To illustrate, a calculator Alice obtains a service ticket using the following procedure:

- Send your name ("Alice") to the Kerberos server.
- Send the name of the service you wish to use ("Email" or "Print").
- Receive four blocks encrypted with the expo cyper using your key in CBC mode. The first block will decrypt to yield the session key you will use for communicating with the service you wish to use. The remaining three blocks, once decrypted, are the service ticket.
- Send the three blocks that comprise the service ticket to the service you wish to use. If this procedure has been performed correctly, you will now be authenticated to that service and share a session key with it.

For this experiment, you will be given a set of calculators representing the Kerberos server, the Email server, and the Print server. The sequence of actions you will will need to perform is thus:

- Obtain a ticket for the mail server from the Kerberos server.
- Give that ticket to the mail server.
- Receive mail from the mail server encrypted with the session key.
- Decrypt the mail, which is a single block of text.
- Obtain a ticket for the print server from the Kerberos server.
- Give that ticket to the print server.
- Encrypt the mail you received earlier using the session key for use with the print server.
- Send the resulting cyphertext to the print server.

CS007

Alice

Kerberos

Email Server

Alice wants to use the Email server. She sends a ticket-request to Kerberos, consisting of her name and the name of the server she needs to use.

"Alice", "Email"

Kerberos responds by preparing a ticket for Alice to use with the Email server.

First, Kerberos randomly chooses a session key Kae, a key for Alice and the Email server to use.

Next, Kerberos constructs a threepart message: Kae, "Alice", expir where expir is the expiration time for the ticket.

Next, Kerberos encrypts this message using the Email server's key. The resulting cyphertext is called the 'ticket'.

Kerberos then prepares a four-part message. The first part is the session key Kae. The remaining three parts are the blocks of cyphertext that comprise the ticket.

Kerberos encrypts this four-part message using Alice's key (typically her password), and sends the result to Alice.

{Kae,ticket1,ticket2,ticket3}_Alice's key

Alice decrypts the message, obtaining the session key Kae, and the three-part ticket.

 \leq

She sends the ticket to the server.

ticket1, ticket2, ticket3

The Email server decrypts the ticket using her key. She obtains the session key Kae, the name of the person requesting service, and the expiration time. She checks the expiration time against the current time, making sure the ticket is still valid. She now can communicate securely with the party requesting service, by encrypting all communication with the session key.

For example, she might send the party her email, encrypted with Kae.

{Alice's email}

Figure 4: Alice interacting with a single server via Kerberos

Fall 1997

CS007

When you have successfully accomplished this, the cell labeled *Printer Output* will contain, as plaintext, your email message.

In Brief						
We give you:	A collection of calculators designed to emulate the Kerberos					
	server, an email server, and a print server					
Your task:	Perform the transactions necessary to retrieve Alice's email					
	and send it to the printer					
Hand in:	Submit once the <i>Printer output</i> cell contains the email					
	plaintext					