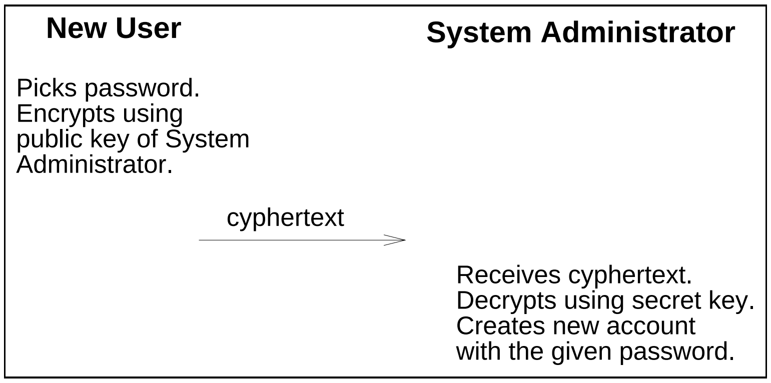
CS007 **Fall 1997**

Homework #9

Out : December 4, 2:30 pm 1997
Due : December 11, 12:30 pm 1997

1. The naive way to use RSA encryption is to directly encrypt the plaintext using RSA, i.e. raise the plaintext to the power of the recipient's public exponent (often 3) modulo the recipient's public modulus. (A better way is to choose a big random key K, encrypt K directly using RSA, and send the encryption of K, then encrypt the real plaintext using a single-key cryptosystem like DES or RC4, and send the resulting cyphertext.)

   Consider the following proposal: whenever a new user needs an account, she selects her password, encrypts it using the public key of the system administrator, and sends the cyphertext to the system administrator via email.


New User
System Administrator
Picks password. Encrypts using public key of System Administrator.
cyphertext
Receives cyphertext. Decrypts using secret key. Creates new account with the given password.

   (a) **Show that if naive RSA encryption is used, Eve can benefit from a dictionary attack.**

   (b) **Why does the better RSA encryption method render the dictionary attack impossible?**

2. Just as we previously used a challenge-response protocol based on ordinary, symmetric-key cryptography (in IFF, Identification Friend or Foe), we can use a challenge-response protocol based on public-key cryptography, RSA in particular. The goal is to make it possible for a party to know who she's communicating with. Suppose Carole is communicating with someone who claims to be Dave. Carole knows Dave's public key (she has previously obtained a certificate for Dave). Carole sends a random challenge R to the other party, who responds by sending back Dave's signature for the value R. Carole can use Dave's public key to check that the signature for R is valid; since only Dave possesses the secret key that would enable someone to sign as Dave, Carole is convinced that she is talking to Dave.

   Carole chooses R according to the uniform distribution on mod-m numbers, where m is Dave's public modulus. We assume m is a huge number (too big to factor), so there are so many possibilities for R that it is extremely unlikely Dave has ever previously calculated the signature for the particular R chosen. It is also extremely unlikely that R is a a number (e.g. 1) for which anyone can compute the signature without knowing the secret key.

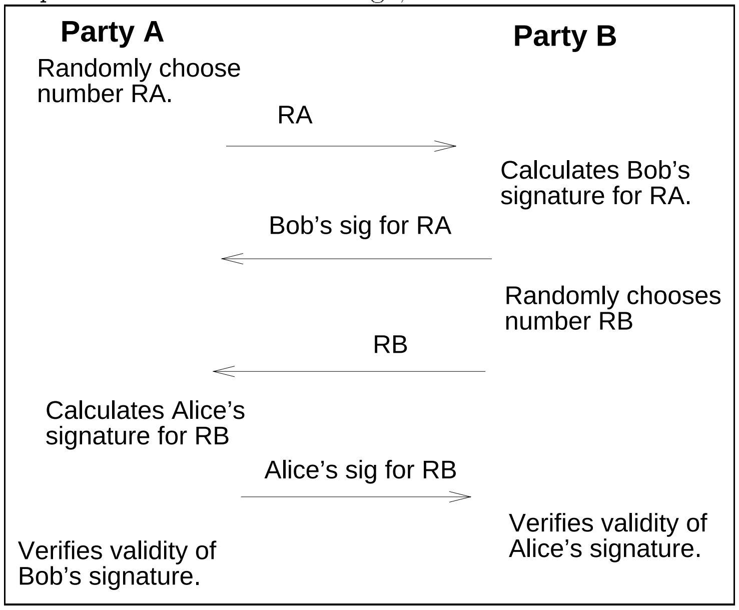
Consider the following protocol to enable two parties to mutually identify each other. Party A is claiming to be Alice, and party B is claiming to be Bob.

(a) Party A chooses a random challenge R_A and sends it to Party B.

(b) Party B responds by sending Bob's signature for the value R_A, and by sending a random challenge R_B of his own.

(c) Party A responds by sending Alice's signature for the value R_B.

Assume that the public moduli used are huge, so it is too difficult for Eve to factor them.


Party A
Party B
Randomly choose number RA.
RA
Calculates Bob's signature for RA.
Bob's sig for RA
Randomly chooses number RB
RB
Calculates Alice's signature for RB
Alice's sig for RB
Verifies validity of Alice's signature.
Verifies validity of Bob's signature.

Recalling your solutions for Lab 7, Experiments 4 and 5, find a similar way for Eve to convince Bob and Alice that they are talking to each other when in fact they are each talking to Eve.

3. We describe a protocol proposed in 1989 for enabling people to communicate privately in a radio network. The protocol uses a trusted center (in this respect it is like Kerberos) to handle the go-between. In this protocol, we ignore issues of authentication and identification: proving who you are is not necessary.

   This protocol requires that only the trusted center have an RSA public-key/private-key pair. All other parties know the public key of the trusted center, and none of them knows the corresponding secret key.

Here is how the protocol goes. Suppose party A wants to communicate privately with party B.

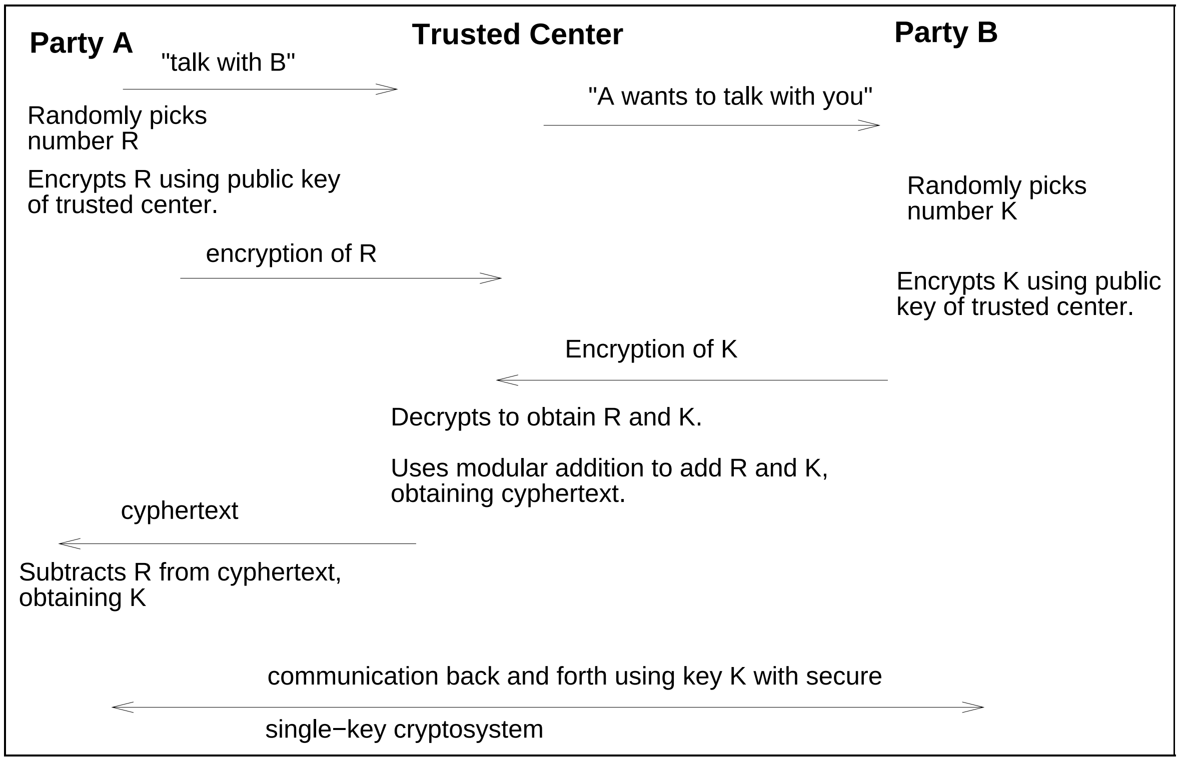
(a) Party A chooses a big random key R, encrypts R using the trusted center's public key, and sends the cyphertext to the trusted center.

(b) The trusted center informs party B of the request.

(c) Party B chooses a big random session key K, encrypts K using the trusted center's public key, and sends the cyphertext to the trusted center.

(d) The trusted center decrypts the cyphertext from A, obtaining the key R, and decrypts the cyphertext from B, obtaining the key K. It uses the one-time pad to encrypt K using key R (i.e. adds K and R using modular arithmetic), and sends the cyphertext to Party A.

(e) Party A decrypts the cyphertext, obtaining K. Now Party A and Party B have a common key, K, and so they encrypt all their messages to one another using an ordinary (single-key) cryptosystem.

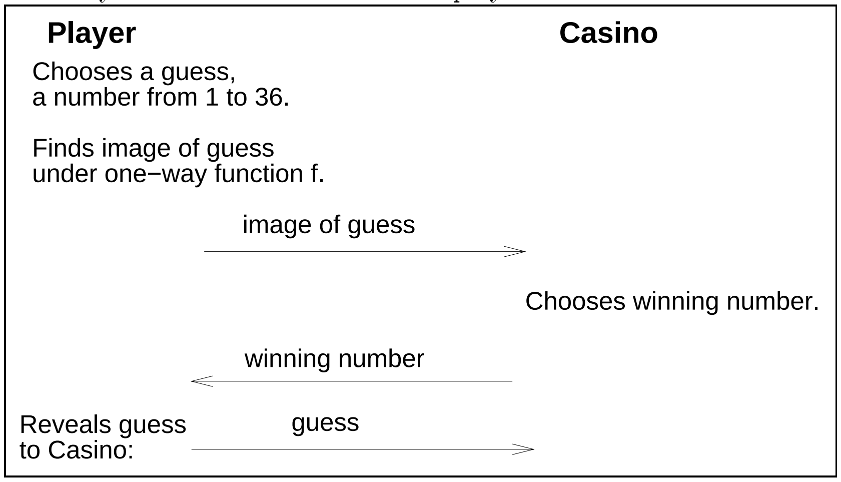

Party A
Trusted Center
Party B
"talk with B"
"A wants to talk with you"
Randomly picks number R
Encrypts R using public key of trusted center.
Randomly picks number K
encryption of R
Encrypts K using public key of trusted center.
Encryption of K
Decrypts to obtain R and K.
Uses modular addition to add R and K, obtaining cyphertext.
cyphertext
Subtracts R from cyphertext, obtaining K
communication back and forth using key K with secure single−key cryptosystem

Assume the trusted center behaves honestly. This protocol *seems* quite secure, but has a rather striking weakness. Two parties C and D working together and eavesdropping on all communication can arrange to decrypt the communication between A and B.

Your job is to figure out how. *Hint 1:* **After eavesdropping on the communication between A, B, and the trusted center, C requests to communicate with D.** *Hint 2:* **The idea we used in blinded signatures helps C and D to avoid getting caught.**

4. Consider an Internet roulette game. You've seen the idea in lab assignments. A player places a bet (perhaps by sending his credit card via SSL), selects one of the numbers 1 through 36, and sends the chosen number to the Internet Casino. The Casino selects one of the numbers (presumably at random!) and then tells the player whether he has won or not. Obviously there is a security problem if the player has to send his chosen number in plaintext form: the Casino can simply avoid that number!

   Here is an apparently better way to implement the roulette game. There is a system-wide one-way function f. Instead of sending his guess, the player uses his guess as the input to f, and calculates the corresponding output and sends it to the Casino. Once the Casino sends the winning number to the player, he tells them his guess. The Casino can then verify for itself that the player is not lying, because it can itself find the image of the alleged guess under f, and verify that it matches what the player sent earlier.



   (a) **Describe the biggest security flaw with this system.**

   (b) *Extra credit:* **Suppose f is an invertible function whose domain is the numbers 1 through 10^{20}. Suggest a way to repair the system so it is secure.**