

# Incorporating the Software GAP into Teaching Abstract Algebra PREP Workshop 2006 Project: Collins and Hoffman

This project explores the RSA public key cipher system. We begin with a few GAP commands which will prove useful. The lists `Primes` and `Primes2` in GAP contain prime numbers of reasonable size for this project. We have already seen the command `mod` which behaves exactly as we expect. For large numbers, `mod` is not efficient. The command `PowerMod` takes advantage of reducing along the way to calculate  $a^r \bmod n$ . Notice the time GAP takes for the following computations.

```
gap> 53^123456 mod 67;  
64  
gap> PowerMod(53,123456,67);  
64
```

We use the functions `encode` and `decode` to change text strings into numbers and numbers into text strings. These functions will ensure that we are all using the same encoding scheme for our message. The input for `encode` is your message as a string and an integer for your block length. The command `decode` takes a list of number blocks and block length as its input.

```
gap> message:=encode("Hello World!",3);  
[ 441, 522, 222, 510, 592, 528, 221, 466 ]  
gap> decode(message,3);  
"Hello World!"
```

These functions are included at the end.

## 1. HOW RSA WORKS

The following algorithms describe the methods we need for sending and receiving messages.

### Constructing your public key.

- (1) Pick 2 large primes  $p$  and  $q$  from the lists `Primes` or `Primes2` and compute  $n = pq$ .
- (2) Compute  $m := \text{Lcm}(p-1, q-1)$ .
- (3) Pick an integer  $r$  relatively prime to  $m$ . Hint: Use `Factors(m)`; to get the prime factorization of  $m$ .
- (4) Find an integer  $s$  such that  $rs \bmod m = 1$ . Hint: Use `Gcdex(r,m)`.
- (5) The integers  $n$  and  $r$  are your public key.

### Sending a message.

- (1) Convert your message to digits. The command `encode( message, blocksize)` will do this for you. `blocksize` must be less than the number of digits in  $n$ .
- (2) For each block  $M_i$  of your encoded message, compute and send  $R_i = M_i^r \bmod n$ .

### Receiving a message.

- (1) For each block  $R_i$  of the received message, compute  $R_i^s \pmod n$ . See discussion on Fermat's Little Theorem to see why this works to decrypt the message (should this be given as a question to the students?).
- (2) Convert your decrypted blocks back into a human readable string using the command `decode( code, blocksize)`.

## 2. PROJECT

- (1) Construct a public key and distribute to class.
- (2) Encrypt a message and send it to a friend.
- (3) Decrypt a message from someone else.
- (4) Recall that Fermat's Little Theorem says that  $a^p \equiv a \pmod p$  for  $p$  a prime not dividing  $a$ . A corollary of Fermat's Little Theorem states that if  $p$  and  $q$  are primes and  $a$  is an integer such that  $a^p \equiv a \pmod q$  and  $a^q \equiv a \pmod p$  then  $a^{pq} \equiv a \pmod{pq}$ . Explain how this result is used in the RSA cipher system.
- (5) Try to decrypt a message not meant for you.
- (6) (advanced) Devise a method to send a signed message with the same public key information.

## 3. NEW FUNCTIONS

```

encode:=function(message,blocksize)
local letters,digits,code,i,out,len;
letters=[" ","a","b","c","d","e","f","g","h","i","j","k",
"l","m","n","o","p","q","r","s","t","u","v","w","x","y",
"z","A","B","C","D","E","F","G","H","I","J","K","L","M",
"N","O","P","Q","R","S","T","U","V","W","X","Y","Z",
",",".", "?", "!", "'"];
code:=ShallowCopy(message);
for i in [1..Length(letters)] do
    code:=ReplacedString(code,letters[i],String(i+9));
od;
len:=Length(code);
out:=[];
for i in [1..(len-(len mod blocksize))/blocksize] do
    out[i]:=Int(code{[(i-1)*blocksize+1..i*blocksize]});
od;
if (len mod blocksize) > 0 then
    out[Length(out)+1]:=Int(code{[len-(len mod blocksize)+1..len]});
fi;
return out;
end;

decode:=function(code,blocksize)
local letters,message,i,out,temp;
letters=[" ","a","b","c","d","e","f","g","h","i","j","k",
"l","m","n","o","p","q","r","s","t","u","v","w","x","y",
"z","A","B","C","D","E","F","G","H","I","J","K","L","M",
"N","O","P","Q","R","S","T","U","V","W","X","Y","Z",
",",".", "?", "!", "'"];

```

```
message:=List(code,x->String(x));
for i in [1..Length(message)-1] do
  if Length(message[i])<blocksize then
    message[i]:=Concatenation("0",message[i]);
  fi;
od;
temp:=JoinStringsWithSeparator(message,"");
if (Length(temp) mod 2) = 1 then
  message[Length(message)]:=Concatenation("0",message[Length(message)]);
  temp:=JoinStringsWithSeparator(message,"");
fi;
out:=[];
for i in [1..Length(temp)/2] do
  out[i]:=letters[Int(temp{[2*i-1..2*i]})-9];
od;
out:=JoinStringsWithSeparator(out,"");
return out;
end;
```