The RSA Encryption Algorithm

Quick Version

Chris Grier

grier@uiuc.edu

You Do Not Need To Know This

- The SIGMIL handout was to be an interesting problem to solve, not a test.
- All of the information you need to solve the puzzle is Google-able.
- The code you need to solve the puzzle is also found on the web in various places.

What is it?

- Created by Rivest, Shamir and Adleman at MIT
- A Public Key Cryptosystem
- An encryption method that is used very often, especially Internet communications. Sometimes its not used correctly.

Private and Public Keys

- General Calculations
  - Generate 2 primes, p, q
  - $n = pq$, and $\phi(n) = (p-1)(q-1)$
- Public Key pair (n, e)
  - n is called the modulus
  - e is chosen (at random with some specific conditions)
- Private Key pair (n, d)
  - same n as the public key.
  - $de \equiv 1 \pmod{\phi(n)}$

Encryption & Decryption

Standard terminology: m is the message, c is the ciphertext.

- Encryption is easy
- $c = m^e \pmod{n}$
- Decrpytion is easy
- $m = c^d \pmod{n}$

Using that stuff

- Decrypting messages is pretty easy from the above formula.
- Determining the decryption key d is equivalent to factoring n. Determining the original message might be easier than finding the key to decrypt it.
- The msg from the website was encrypted and uuencoded.
- The RSA decryption and uudecoding can be done in 3 lines of perl which don't really fit well on this page.

```
#!/usr/bin/perl -s-- -export-a-crypto-system-sig -RSA-in-3-lines-P
($k,$n)=@ARGV;$m=unpack(H.$w,$m."\0"x$w),$_=`echo "16do$w 2+4Oi0$d
Sa2/d0<X+d*La1=z\U$n%0]SX$k"[$m*]\EszlXx++p|dc`,s/^.|\W//g,print p
,$_)while read(STDIN,$m,($w=2*$d-1+length($n||die"$0 [-d] k n\n")&
```

Finding d

- Convert from hex to decimal -
  echo 'obase=10; ibase=16; ADDA866025386AB4C71A1C4E53353D19' | bc
  231091089446260678243487602452544699673
- To find d, given only the public key (n, e), we need to factor n. For a small n like in the puzzle, the Eliptic Curve (EC) method works very well. There's an applet that will factor for you at: http://www.alpertron.com.ar/ECM.HTM
- Now we have p, q, subtract one from each and multiply to find $\phi(n)$. Then find $e^{-1} \bmod \phi(n)$

$(e)^{-1} \bmod \phi(n)$

```
import java.math.BigInteger;
public class invert
{
    public static final BigInteger ONE = new BigInteger("1");
    public static void main(String [] args)
    {
        BigInteger e = new BigInteger("4263582709");
        BigInteger q = new BigInteger("857781177494920426 9");
        BigInteger p = new BigInteger("2694056427318016591 7");

        q = q.subtract(ONE);
        p = p.subtract(ONE);

        BigInteger phi = p.multiply(q);

        System.out.println(phi);
        System.out.println(e.modInverse(phi));
    }
}
```

No More Programming

- uudecode -o quad_day.decoded quad_day
- need $n = ADDA866025386AB4C71A1C4E53353D19$
- need $d = 6B1A17B887D1A0AF0A12CA4B4B01553D$
- cat quad_day.decoded | perl rsa.pl -d 6B1A17B887D1A0AF0A12CA4B4B01553D ADDA866025386AB4C71A1C4E53353D19
- Do it.

Math

- Requirement $D(E(m)) = m$!

$$\begin{aligned} D(E(m)) &= (m^e)^d \\ &= m^{ed} \\ &= m^{\phi(n)k} \end{aligned}$$

- I was going to prove that determining d is the same as factoring n. This is the underlying fact that makes the RSA encryption algorithm secure. Security here is just computational difficulty.