95-702 Distributed Systems

Lecture 12: RSA

Plan for today:

- Introduce RSA and a toy example using small numbers.
  This is from Introduction to Algorithms by Cormen, Leiserson and Rivest

- Describe an interesting cryptographic protocol and its limitations.
  This is from Applied Cryptography by Bruce Schneier.

- Show how RSA cryptography can be done in Java.
  See the Java Cryptograhy API.

Purpose of RSA

Privacy: to send encrypted messages over an insecure channel.

Authentication: To digitally sign messages.

RSA was not the first public key approach.
Public key cryptography was first introduced by Diffie and Hellman in 1976.

RSA was developed by Rivest, Shamir, and Aldeman in 1977.
It's probably safe to call public key cryptography revolutionary.

The Cast of Characters

- Eve - tries to view messages she should not be viewing.
- Mallory - tries to manipulate messages and be disruptive .
- Bob and Alice - try to communicate over insecure channels.

The RSA Key Generation (1)

1. Select at random two large prime numbers p and q. These numbers would normally be about 500 digits in length.
2. Compute n by the equation $n = p \times q$.
3. Compute $\phi(n) = (p - 1) \times (q - 1)$
4. Select a small odd integer e that is relatively prime to $\phi(n)$

The RSA Key Generation (2)

5. Compute d as the multiplicative inverse of e modulo $\phi(n)$. A theorem in number theory asserts that d exists and is uniquely defined.
6. Publish the pair P = (e,n) as the RSA public key.
7. Keep secret the pair S = (d,n) as the RSA secret key.

RSA Encryption and Decryption

8. To encrypt a message M compute
   $C = M^e \pmod{n}$

9. To decrypt a message C compute
   $M = C^d \pmod{n}$

Toy Example: Key Selection(1)

1. Select at random two large prime numbers p and q. These numbers would normally be about 500 digits in length.

   $p = 3 \quad q = 11$

2. Compute n by the equation $n = p \times q$.

   $n = 33$

3. Compute $\phi(n) = (p - 1) \times (q - 1)$

   $\phi(n) = (2) \times (10) = 20$

Toy Example: Key Selection(2)

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20$

4. Select a small odd integer e that is relatively prime to $\phi(n)$

   $e = 3$

Toy Example: Key Selection(3)

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3$

5. Compute d as the multiplicative inverse of e, modulo $\phi(n)$. A theorem in number theory asserts that d exists and is uniquely defined (since e and $\phi(n)$ are relatively prime).

We need a d so that $ed \bmod \phi = 1$

Let's try 1.
$3 \times 1 \bmod 20 = 3 \bmod 20 = 3$. Nope.

Toy Example: Key Selection(4)

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3$

We need a d so that $ed \bmod \phi = 1$

Let's try 2.

$3 \times 2 \bmod 20 = 6 \bmod 20 = 6$. Nope.

Let's try 7.

$3 \times 7 \bmod 20 = 21 \bmod 20 = 1$. We found it!

This approach is too slow. A fast approach exists.

Toy Example: Publish The Public Key

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3 \quad d = 7$

6. Publish the pair P = (e,n) as the RSA public key.

"Hey everyone, my key pair is 3 and 33"

7. Keep secret the pair S = (d,n) as the RSA secret key.

"I'm not telling anyone about 7 and 33!!"

Toy Example: Message encoding phase

Bob's public keys are e = 3 n = 33

Alice wants to send the letter 'd' to Bob.
Suppose that we have a public code where
'a' = 0 'b' = 1 'c' = 2 'd' = 3 and so on…

Alice's software knows that

8. To encrypt a message M compute
   $C = M^e \pmod n$ $= 3^3 \bmod 33 = 27 \bmod 33 = 27$

Toy Example: Message decoding phase

Bob's private keys are d = 7 n = 33

Bob receives a 27 from Alice:

9. To decrypt a message C compute

$M = C^d \pmod{n}$

$= 27^7 \bmod 33 = 10460353203 \bmod 33$

$= 3$ (which is 'd')

An Example - Secure Voting

We want to think about:

Business Requirements
Cryptographic protocols
Threat models

Goals Of Secure Voting

- Only Authorized Voters Can Vote
- No one can vote more than once
- No one can determine for whom anyone else voted
- No one can duplicate anyone else's vote
- No one can change anyone else's vote without being discovered
- Every voter can make sure that his vote has been taken into account in the final tabulation.

First Attempt

- Each voter encrypts his vote with the public key of a Central Tabulating Facility (CTF)
- Each voter send his vote in to the CTF
- The CTF decrypts the votes, tabulates them, and makes the results public
- What are some problems with this protocol?

Second Attempt

- Each voter signs his vote with his private key
- Each voter encrypts his signed vote with the CTF's public key
- Each voter send his vote to the CTF
- The CTF decrypts the votes, checks the signature, tabulates the votes and makes the results public
- What are some problems with this protocol?

How do we do RSA in Java?

RSA In Java(1)

- How do I create RSA keys?

  Use the Biginteger class and do your own calculations or

  Use Java's keytool:

  keytool -genkey -alias mjm -keyalg RSA -keystore mjmkeystore

RSA In Java(2)

- How do I read the RSA keys from a keystore?

```
String keyFileName = "coolkeys";
String alias = "mjm";
char[] passWord = "sesame".toCharArray();
FileInputStream fis = new FileInputStream(keyFileName);
KeyStore keyStore = KeyStore.getInstance("JKS");
System.out.println("Load key store with file name and password");
keyStore.load(fis, passWord);
```

RSA In Java(3)

- How do I decrypt encrypted data with the private key?

```
RSAPrivateKey RSAKey =
(RSAPrivateKey)keyStore.getKey(alias,passWord);
Cipher RSACipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
RSACipher.init(Cipher.DECRYPT_MODE, RSAKey);
byte decryptedKeyBytes[] = RSACipher.doFinal(encryptedBlowFishKey);
```

RSA In Java(4)

- How do I generate a certificate?

Use the keytool and the keystore:

keytool -export -alias mjm -keystore mjmkeystore –file cool.cer

RSA In Java(5)

- How do I read the public key from the certificate?

```
CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
FileInputStream fis = new FileInputStream("cool.cer");
Certificate cert = certFactory.generateCertificate(fis);
fis.close();
PublicKey pub = cert.getPublicKey();
```

RSA In Java(6)

- How do I encrypt with the public key?

```
Cipher cipherPub = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipherPub.init(Cipher.ENCRYPT_MODE, pub);
byte encryptedBlowFish[] = cipherPub.doFinal(blowFishKeyBytes);
```