Cryptographic Protocols

Message Transmission from Bob to Alice

COMP 460 Midterm

1. Symmetric Encryption/Decryption of an arbitrary size message using AES. Here the key is transmitted as such but the message is encrypted with the symmetric key. Bob sends the encrypted message to Alice with the key. The key is sent in the open and can be compromised by an interceptor. (CryptoUtils.java)
2. Signature verification of sender Bob: Bob uses private key to sign the message. Bob sends the signature B, his public key, and the message to Alice. Alice constructs a signature object A using the message and Bob's public key. Then runs the verification method comparing A and B. If verification succeeds, then only Bob could have sent the message. Here the message is sent in the open (unencrypted). (PKTest.java)
3. Encryption of Symmetric Key: In (1) you do want to avoid sending the key out in the open, so here is what can be done. Use AES to generate a 128 bit (16 bytes) secret key. Then you use RSA and Cipher to encrypt the secret key. There are limits to the size of the secret key that you can encrypt with RSA. 1024 bit modulus RSA can encrypt up to 117 bytes, 2048 up to 245 bytes and 4096 up to 501 bytes. In particular, Bob uses Alice's public key and Cipher to encrypt secret key and transmits the encrypted symmetric key to Alice. Alice decrypts the key using her private key. In this protocol we ensure only Alice can decrypt the encrypted symmetric key. (KeyEncryption.java)
4. You can combine 1 and 3 where Bob can send an encrypted message of any size to Alice using symmetric encryption, where the symmetric key can be encrypted (and sent) using Alice's public key (say with RSA 1024 bit modulus). The encrypted symmetric key will have a size 128 bytes (1024 bits). Here you want two separate Java class files one corresponding to Bob's and the other corresponding to Alice's actions.
5. You can take it a higher level of secure protocol, where you combine sender authentication with 1 and 3. How does this work? Alice uses RSA 1024 bit modulus to generate the key pair. Bob uses Alice's public key to encrypt the symmetric key S (using AES 128 bits), to result in A. Bob generates a stronger RSA key pair using 2048 bit modulus. Bob uses his private key to encrypt A that produces a 256 byte double encrypted key, B. Bob encrypts arbitrary length message with the secret key S (method 1 above) resulting in M. Bob sends B and encrypted message M to Alice. Alice first uses Bob's public key to decrypt (authenticating that Bob was the sender), and then her own private key to get back S. Alice then uses S to decrypt message (from method 1). Here again create two separate Java files one for each player Bob and Alice are needed.

Secure as this is, still suffers from a malicious attacker who can mess up the bytes transmitted, even if the attacker could not decipher the transmission. Neither Bob nor Alice can totally prevent the attacker from messing up the bytes. But at least Alice should have a way to find out if someone did mess up the bytes. This is the improvement in the next protocol where you ensure message integrity.

6. Here you ensure message integrity on top of protocol 5. In this, you create a message digest of what Bob would send to Alice by way of protocol 5. Bob does a cryptographically secure hash of (B+M) and sends B, M and Hash(B+M) over to Alice. Note that Hash(B+M) is accomplished by Java's MessageDigest class using SHA256 hash. Use of SHA256 is illustrated in ReadFileInByteArrayWithFileInputStream.java . (Sorry for the terrible long file name). What Alice does is first checks whether Hash of received (B+M) is equal to Hash(B+M) sent by Bob. Then proceeds to unravel B and M using protocol described in 5. Also, two files.

7. Oh, I just thought of this diabolical next improvement -- are you ready?? Since we learned a bit of network communication as a prelude to learning peer to peer algorithms in this course, you can set up Bob's code on one vm and Alice's code on another vm and build a secure TCP/IP message protocol using 6 above and plain old TCP/IP. This step 7, I am going to spare you from the midterm, but perhaps we will consider for the finals. Don't hate me just yet.

   Midterm work: Your goal is to get to 6 via protocols 4 and 5. In class we covered until protocol 3. So you have all the tools needed. Submit code files for 4,5, and 6 on Sakai. thanks