Reti di Calcolatori

Esercitazione su:

Crittografia & Bet Server

Si vuole implementare un server che gestisce scommesse sulle corse dei cavalli.
Il server riceve le scommesse da parte di utenti client che indicano il numero del cavallo (da1a12) su cui vogliono puntare.
Le scommesse sono ricevute, tramite il protocollo TCP, sulla porta 8001 del server.
Il client effettua una scommessa inviando al server la stringa “<num_cavallo> <puntata>”, in cui <num_cavallo> è il numero del cavallo, e <puntata> è l’ammontare in euro della scommessa.
Il server accetta le scommesse fino all’ora <ora_limite> . A tal fine un thread del server controlla periodicamente, es. ogni minuto, l’ora del sistema. Quando l’ora corrente raggiunge l’ora <ora_limite>, il server deve chiudere l’accettazione delle scommesse.
Il server memorizza in un’opportuna struttura dati le scommesse pervenute.
Scaduto il tempo utile, il server verifica il numero del cavallo vincente (a tal proposito si può utilizzare un metodo che estrae un numero casuale da 1 a 12), e successivamente comunica a tutti gli scommettitori l’esito positivo o negativo della loro scommessa.
Tale comunicazione avviene tramite l’invio, sulla porta TCP 8002 di ogni client, della stringa “Somma vinta=<somma>” o “Scommessa persa”, a seconda se il client aveva indovinato o no il numero del cavallo vincente. <somma> è l’ammontare della eventuale somma vinta, corrispondente alla scommessa effettuata dal client moltiplicata per 12.

Crittografia & Bet Server

Preparata la scommessa, per garantire la segretezza dei dati, il client invia tali dati al server in forma criptata. In modo che solo il server li possa decifrare.

Al termine della corsa, il server comunica a tutti gli scommettitori l'esito positivo o negativo della loro scommessa. Per garantire l'affidabilità dei dati, il server cripta l'elenco delle vincite in modo che i client decifrandolo ne verifichino la provenienza.

Implementare nel client e nel server i meccanismi necessari alla segretezza e all'affidabilità dei dati inviati. Basandoli su una comune infrastruttura a chiave pubblica (PKI).

Soluzione teorica:

- Il server genera una coppia di chiavi RSA, ed attiva un thread “SendKey” che sta in ascolto sulla porta 8443. SendKey ha il solo compito di fornire la “chiave pubblica” del Bet Server a tutti i client che ne fanno richiesta.

- Il client, che vuole effettuare una scommessa, recupera dal SendKey la chiave pubblica e la usa per criptare la scommessa prima di spedirla (Segretezza).

- Il server riceve la scommessa criptata e la decifra con la sua chiave privata.

- Al termine della corsa, il server genera l’elenco delle vincite e prima di inviarlo agli scommettitori lo cripta con la sua chiave privata.

- I client, che erano rimasti in attesa del risultato, decifrano l’elenco delle vincite con la chiave pubblica del Bet Server recuperata in precedenza (Affidabilità).

Soluzione implementata:

Vedi file:

- BetClient.java
- BetServer.java
- Scommessa.java

Problematiche:

```
// Quando si trasmettono chiavi o messaggi criptati si deve lavorare a
// basso livello, leggendo o scrivendo direttamente i “byte” sugli Stream
Socket  socket = new Socket(……..);
InputStream  in  =  socket.getInputStream();
OutputStream  out  =  socket.getOutputStream();

// Invio della chiave pubblica
out.write(  keyPair.getPublic().getEncoded()  );

// Ricezione della chiave pubblica
byte[] publicKeyBytes= new byte[1000];
int numByte = in.read( publicKeyBytes );
publicKeyBytes = Arrays.copyOf( publicKeyBytes, numByte );
```

Problematiche:

```
// Rigenerazione della chiave pubblica
X509EncodedKeySpec ks = new X509EncodedKeySpec(publicKeyBytes);
KeyFactory kf = KeyFactory.getInstance("RSA");
PublicKey publicKey = kf.generatePublic(ks);

// Invio di testo cifrato
out.write( cipher.doFinal( “Testo originale da cifrare”.getBytes() ) );

// Ridezione di testo cifrato
byte[] testoCifrato = new byte[1000];
int numByte  =  in.read( testoCifrato );
testoCifrato = Arrays.copyOf( testoCifrato , numByte );
byte[] testoDecifrato = decipher.doFinal(scommessaBytes);
String testoOriginale = new String( testoDecifrato );
```