Bingo segur

Treball de fi de carrera

Alumne: Jose Luis Villasante Ramón
Enginyeria Tècnica en Informàtica de Sistemes

Consultor: Antoni Martínez Ballestè
Data de lliurament: Juny 2006

Agraïments

Aquest treball està dedicat a la meva esposa Merche i als meus fills Paula i Jaume per compartir els bon moments i ser capaços de aguantar-me en els moments dolents, donant-me el seu suport.

Vull agrair també l'aportació humana i de coneixement rebuda de totes aquelles persones amb les que he tingut el plaer de compartir aquest camí: consultors, companys,.. moltes gràcies a tots!.

Resum:

Una important línia d'investigació en Criptografia és la Computació Multipart Segura (CMS), que cerca mètodes per a realitzar el càlcul d'una funció $f(x_1, x_2, ..., x_n)$ a partir de les 'n' entrades individuals, amb el propòsit que aquestes entrades romanguin secretes i el resultat final sigui calculat a partir de totes i cadascuna d'elles.

En aquest TFC es proposa un protocol de CMS aplicat a un Joc de Bingo, en que les entrades són nombres generats pels jugadors i el resultat cercat és un nombre aleatori per al joc.

Aquest protocol s'implementa en una aplicació Java iniciada per un Servidor en la que varis jugadors poden connectar-se a una partida.

Paraules clau: Computació Multipart Segura, compromís de bit, esquemes de llindar, esquemes (m,n), esquema de compartició de secrets de Shamir, certificat digital, signatura digital

Índex de continguts

Estat de l'art 5

Especificació del problema 8

Esquema criptogràfic de la implementació 9
- Generació de nombres aleatoris de forma segura 9
- Elecció i repartiment de cartrons de joc 15
- Comprovació i verificació de premis 17

Disseny de l'aplicació 18
- Guió del servidor 18
- Guió del jugador 20
- Guió del protocol de càlcul segur 21
- Estructura de l'aplicació 23

Manual del usuari 28
- Instal·lació i execució 28
- Desinstal·lació de l'aplicació 31

Proves de funcionament efectuades 32

Conclusions 33

Bibliografia 34

Recursos de programari emprats 34

Annex A. Recerca de llibreries criptogràfiques 35

Annex B. Algorisme per a generar un cartró de Bingo 36

Glossari 38

Estat de l'art

El Treball de Fi de Carrera (TFC) proposat s'integra dins d'una important línia d'investigació dins de l'àrea de Criptografia: la computació multipart segura (CMS), plantejada en forma de problema en que 'n' participants, amb comunicació entre sí i en situació de desconfiança mútua, desitgen calcular de forma distribuïda el resultat d'una funció a partir de les 'n' entrades individuals, amb dos requisits importants:

- cal tenir seguretat que el resultat calculat s'obtingui amb participació de totes les entrades individuals de cadascuna de les parts, que sigui correcte i no manipulat per cap d'elles ni per part de tercers aliens.
- les dades aportades per cada part individual han de mantenir-se tan privades com sigui possible. Es a dir, pel fet de participar en el càlcul cap de les parts hauria d'adquirir un coneixement que vagi mes enllà del resultat computat de la funció.

Podria argumentar-se que seria factible aconseguir aquesta funcionalitat únicament amb l'existència d'una tercera part, de confiança, que recopilés les dades de forma segura i fes el càlcul a partir d'elles, enviant el resultat obtingut a cadascuna de les parts igualment de forma segura. Es a dir, existiria un tercer incorruptible e infal·lible que seria l'encarregat de dur a terme aquestes tasques.

Malauradament, en la nostra realitat l'actitud predominant i d'altre banda inevitable, és la desconfiança mútua, a la que s'afegeix la inseguretat en les comunicacions. Aquestes disfuncions són les que donen sentit a l'àrea d'investigació al voltant de la computació multipart segura.

Hi ha nombroses i importants aplicacions d'aquesta àrea de coneixement. A títol d'exemple podem assenyalar:

- eleccions electròniques. La funció calculada seria en aquest cas la suma dels vots individuals per a cada opció (recompte). És clar que cap votant hauria de poder conèixer el sentit del vot de qualsevol altre participant en els comicis.
- subhastes electròniques. En aquest cas la privacitat requeriria que només fos coneguda la puja guanyadora, mentre que per assegurar la correcció caldria que la puja guanyadora fos efectivament la més alta.
- preservació de la intimitat en processos de mineria de dades. Per exemple, en recerca mèdica pot ser imprescindible la compartició de dades de diferents malalts per a obtenir avanços en una investigació, però el sentit comú i la legislació requereixen una especial cura en el manteniment de la intimitat d'aquestes. Computar conjuntament i de forma segura aquestes dades possibilita avanços en coneixements mèdics.

Usualment la computació multipart segura d'una funció pren forma de protocol, en el sentit que està composat per una successió de rondes en la que cadascuna de les parts du a terme unes determinades accions, envia entrades o rep algun resultat de la computació.

Els aspectes bàsics que ha de considerar un protocol de CMS són:

- l'entorn de comunicacions en el que funcionarà.
- l'estructura de possibles adversaris.
- la definició de seguretat que ha de complir

Respecte a l'entorn de comunicacions podem distingir dues possibilitats:

- el model de canals segurs, en el que qualsevol parell de participants disposa d'un canal de comunicació no accessible per cap altre participant.
- el model criptogràfic, en el que qualsevol part –adversari o no- te lliure accés a qualsevol missatge de qualsevol participant.

En quant als adversaris, el model habitual és considerar una entitat que pren el control total de l'ordinador del participant(s) 'corromput(s)' i que per tant te accés a totes les dades del protocol generades fins a aquell moment per aquest. Es distingeixen dos possibles graus de corrupció:

- corrupció passiva, on l'adversari accedeix a la informació dels participants corromputs, però aquests continuen executant correctament el protocol. L'objectiu de l'adversari és recopilar informació que hagués hagut de romandre privada. Es parla també d'adversaris semi-honestos.
- corrupció activa, on els participants corromputs poden desviar-se arbitràriament de les especificacions del protocol, amb l'objectiu d'influenciar en el seu resultat final –i potser recopilar informació privada-. En aquest cas, es parla d'adversaris 'maliciosos'.

Ambdós tipus de corrupció poden presentar-se de forma estàtica –si el conjunt de participants corruptes és escollit o es forma abans de començar el protocol i es manté invariable fins a la seva fi- o adaptativa –en cas que l'adversari pugui corrompre a qualsevol participant en qualsevol moment del protocol -.

El tercer aspecte a considerar: la seguretat, potser és el component mes important en el sentit que la seva consecució és la màxima aspiració d'un protocol de CMS, però a la vegada és el component més eteri doncs és força difícil de definir.

Una aproximació clàssica consisteix en definir una llista de requeriments que es preten assolir amb el protocol: les entrades han de mantenir-se en secret per a la resta de participants, el resultat de la funció computada ha de ser correcte, etc.

No obstant, es difícil estar plenament segur de que la llista es complerta. Per exemple en el cas de les subhastes 'en línia', un protocol que aconsegueix mantenir les entrades –pujes- en secret pot ser incorrecte si les entrades dels adversaris són dependents de les entrades dels participants honestos, doncs tot i no conèixer amb exactitud l'aportació dels participants honestos és suficient amb que la puja dels 'corromputs' sigui superior per guanyar la subhasta de manera fraudulenta.

És degut a això que la seguretat sovint es formalitza mitjançant la contraposició de dos móns, el mon ideal on s'assumeix l'existència d'un tercer incorruptible, que sempre duu a terme el protocol d'acord amb les especificacions desitjades, recollint les entrades, computant el resultat correcte i distribuint-lo oportunament a tots els participants, i el mon real, amb desconfiança mútua entre els participants i on són perfectament possibles la seva corrupció i els atacs al protocol.

Des d'aquesta perspectiva, la idea és que un protocol és bo si el seu resultat en el mon real no pot distingir-se del que s'obtindria en el mon ideal i d'igual forma, un atac al protocol serà admissible en el mon real si també ho és en el mon ideal.

Finalment indicar que una important eina de treball per a l'elaboració de protocols CMS són els esquemes de compartició de secrets, o esquemes de llindar (m,n) en els que un secret és repartit entre n participants, de forma que només pot recuperar-se si es reuneixen m o més fragments. Si és possible aconseguir que les operacions requerides sobre els valors secrets puguin realitzar-se sobre els fragments, de forma que a partir d'ells puguem recompondre el resultat de la funció, haurem construït la base d'un protocol CMS.

Especificació del problema

El problema plantejat en aquest TFC consisteix en la confecció d'un joc de bingo segur per Internet, en un entorn de desconfiança mútua, en que el nombre corresponent a cada jugada sigui generat de forma segura entre tots els jugadors mitjançant una funció calculada de forma segura a partir de les entrades individuals, sense la intervenció d'una tercera part mes enllà d'un paper de distribució de cartrons i missatges.

De forma esquemàtica, el joc de Bingo hauria d'assolir:

- inici del joc
    - els jugadors efectuen peticions de connexió i el servidor les gestiona, produint-se un intercanvi de claus públiques (el servidor i cada jugador hauran de disposar d'un parell de claus privada-pública).
    - el servidor genera els cartrons i els reparteix. Cada jugador hauria de veure només els números del seu cartró.
- desenvolupament del joc
    - cada jugador genera una llavor pròpia, que juntament amb la generada per la resta de jugadors servirà pel càlcul conjunt del valor pseudoaleatori. Cap dels jugadors hauria de poder conèixer altres valors abans d'enviar el seu propi, ja que altrament podria alterar el resultat, i fins i tot, idealment aquestes llavors haurien de restar 'secretes' per a qualsevol jugador diferent del que l'hagi generat. D'altre banda, tot jugador hauria de tenir la seguretat de que els valors rebuts per a dur a terme el càlcul han sigut emesos pels jugadors que teòricament ho han fet.
    - Un cop 'oberts' els valors, tots els jugadors fan el càlcul corresponent per a obtenir el valor aleatori desitjat. Evidentment, tots els jugadors han d'haver rebut idèntics resultats de la funció i han de calcular el número pseudoaleatori de la mateixa forma. D'altre banda, els jugadors han de tenir seguretat de que el seu valor ha contribuït al resultat final.
    - si un jugador 'canta' línia o bingo, hauria d'establir-se un mecanisme per tal que tots els jugadors poguessin comprovar la seva correcció.

Esquema criptogràfic de la implementació.

Els requisits de seguretat exposats en l'especificació del problema condueixen a tres punts fonamentals

- generació de números aleatoris de forma segura.
- creació i repartiment de cartons, assegurant autenticitat e integritat.
- comprovació de premis i verificació per tots els jugadors.

Generació de nombres aleatoris de forma segura

D'una banda, i sota la suposició de l'existència d'una infrastructura de clau pública mitjançant la que qualsevol participant pot comprovar la identitat de la resta aconseguim acomplir els tres serveis bàsics de la Criptografia en quant a seguretat:

- autenticitat, assegurant que el remitent d'un missatge és qui realment diu ser.
- integritat, impedint que el missatge sigui manipulat un cop ha sortit del remitent.
- no repudi, evitant que el remitent del missatge pugui rebutjar la seva autoria.

Un cop comprovat que el certificat és vàlid i de confiança, podem assolir els tres aspectes mitjançant l'ús de la tècnica de la signatura digital.

Per a l'elecció conjunta del número de cada jugada tractarem de construir un protocol de Computació Multipart Segura (CMS), de forma que 'n' participants en condicions de desconfiança mútua puguin computar el resultat d'una funció, a partir de les seves entrades individuals.

Per a definir-ho el més acuradament possible, ens caldrà primer estudiar per al nostre cas particular els tres aspectes bàsics que tot protocol de CMS hauria de considerar:

- l'entorn de comunicacions en el que actuarà. En el TFC proposat, es demana expressament que el joc es pugui desenvolupar en la xarxa Internet. Ens trobem doncs, davant un model de comunicacions amb canals insegurs i en el que cadascun dels participants pot accedir de forma mes o menys senzilla als missatges intercanviats per a la resta de participants.

  L'objectiu serà crear canals segurs entre els diversos participants simulant connexions no interceptables i punt a punt entre cada parell de jugadors. Una forma d'aconseguir-ho és fent us del xifratge de clau pública, de forma que els missatges enviats a un participant es xifren amb la seva clau pública, i són desxifrats pel destinatari amb la seva clau privada.

  En l'aplicació a desenvolupar, el xifratge de clau pública es limitarà a xifrar aquells missatges en que sigui aconsellable mantenir la confidencialitat enfront de la resta de participants, com per exemple el repartiment de cartrons de joc.

  Això requereix una infrastructura de clau pública, en la que cada participant pugui assegurar-se de la identitat de la resta i que en el nostre cas, per a simplificar, donarem per feta, en el sentit que donarem confiança als certificats presentats per jugadors i servidor a l'inici del joc, entenent que

han sigut emesos per una CA (Certificate Authority) de confiança pels participants, que són vàlids en el temps i que no han estat revocats.

De forma addicional, ens interessarà garantir l'origen i la integritat del missatge i el seu origen. Per això farem ús abastament de les tècniques de signatura digital.

- l'estructura de possibles adversaris. Com quasi sempre es fa en Criptografia, considerarem el pitjor dels casos: adversaris maliciosos –no tenen perquè seguir el protocol i de fet procuraran eludir-ho en el seu benefici- i adaptatius, en el sentit que poden ser corromputs en qualsevol moment del joc i no només a l'inici.

  D'aquesta forma el protocol que hem de construir haurà d'obligar als jugadors deshonestos a seguir-lo, detectant qualsevol possible intent de desviació. En la tria d'un número aleatori amb intervenció de tots els jugadors el comportament maliciós que hem d'evitar és que els participants puguin crear les seves entrades per a aconseguir o influir un determinat resultat de la funció i d'aquesta forma eliminar l'aleatorietat en la tria del número corresponent a cada jugada.

  Suposarem que llevat problemes de comunicacions, un cop iniciat el protocol -i fins i tot podríem dir el joc- el jugador el finalitzarà, es a dir, el comportament maliciós no induirà a l'avortament voluntari del protocol. Aquesta suposició que pot ser massa forta en altres aplicacions de protocols de CMS no ho és en el nostre cas, ja que si un jugador deixa de participar perd la possibilitat d'aconseguir el premi.

- la definició de seguretat que ha de complir. Podríem dir que en el nostre cas serà necessari obtenir:
  - la participació de tots els jugadors en el càlcul del nombre aleatori de cada jugada
  - la correcció en el resultat final –nombre aleatori- en funció de les entrades aportades per cadascun dels jugadors.
  - la independència de les entrades, en el sentit que cap jugador deshonest pugui calcular la seva entrada per a la computació de la funció segons les entrades d'un o més jugadors honestos.
  - el secret de les entrades aportades per a cada jugador. Tot i que sembla que complint-se la independència esmentada en el paràgraf anterior aleshores el secret de les entrades no sembla un requisit fonamental per a la tria del número aleatori, òbviament si les entrades són conegudes només pel jugador que les aporta ajuda de forma clara a la consecució d'aquesta independència. D'altre banda, el secret de les entrades sol ser un requisit bàsic en la majoria de situacions en les que un protocol de CMS és aplicable, i el seu estudi és útil i interessant.

Un cop delimitats els aspectes bàsics als que el protocol haurà de donar resposta, passarem a la seva construcció.

Com primera decisió hem de triar quina funció voldrem calcular mitjançant el protocol CMS. La funció més senzilla d'implementar és la suma de les entrades individuals de cadascun dels jugadors.

En una jugada cada participant aportarà com entrada un número enter positiu i aleatori, diferent en cada jugada i en principi de qualsevol mida però preferiblement gran. El protocol conduirà al càlcul de la suma de les entrades individuals i aquesta suma constituirà la llavor per a generar el número aleatori, d'aquella jugada concreta.

El requisit d'independència de les entrades serà possible si cap jugador pot efectuar càlculs abans que tots els altres estiguin en disposició de fer-ho. Per aconseguir-ho, podem fer ús de la tècnica criptogràfica de 'compromís de bit', de forma que cada jugador es compromès a un valor per a la seva entrada al protocol i tots els valors són descoberts en el mateix moment.

Un possible esquema per aconseguir que la tria del número sigui aleatòria consisteix en que cada jugador xifri el seu valor amb una 'clau de sessió' única per a aquella jugada i que faci arribar el missatge xifrat a la resta dels jugadors. Només quan un jugador ha rebut tots els compromisos, envia a la resta de jugadors la clau que permet obrir-los. Així, si l'esquema de xifratge és prou bo serà gairebé impossible trobar un número que proporcioni el mateix compromís i per tant, ens garanteix la independència de les entrades respecte de la resta, així com la correcció del resultat, ja que tots els jugadors obtenen el mateix número com a resultat de la suma. Necessàriament, aquesta clau de sessió diferent per a cada jugada haurà de ser simètrica, es a dir, la mateixa clau xifrarà i desxifrarà.

Fins ara hem abordat la independència de les entrades, però per a calcular la llavor conjunta els jugadors han de fer la suma de totes les entrades, que per tant deixen de ser secretes, incomplint un dels requisits de seguretat que havíem esmentat.

Per aconseguir el secret de les entrades podem fer ús d'un esquema de compartició de secrets. Un esquema criptogràfic relativament senzill i que ens permet assolir aquesta fita és l'esquema de compartició de secrets proposat per Shamir basat en la interpolació polinòmica.

En aquest esquema cada jugador, selecciona un polinomi de grau n-1, on n representa el número de jugadors o participants, de forma que el seu valor secret 'k' sigui el resultat de l'avaluació del polinomi en el punt i=0. Per tant, el valor secret 'k' serà el terme independent del polinomi. La resta de coeficients es seleccionaran de forma aleatòria entre números enters, positius, negatius i fins i tot encara que no aconsellable, zero.

$$s_i = k + a_1 \cdot i + a_2 \cdot i^2 + \ldots + a_{n-1} \cdot i^{n-1}$$

Aleshores, per a cadascun dels altres participants, el jugador avalua el polinomi en un punt diferent i envia al jugador en qüestió el parell de valors (i, s_i).

Cada jugador podria escollir de forma arbitrària els punts on avaluar el seu polinomi, però per comoditat i facilitat d'implementació, nosaltres assignarem un índex 'i' a cada jugador (i=1, 2, ..., n) i assimilarem aquest índex al punt on

s'avaluarà el polinomi en cada cas: per al jugador 1, el parell (1, s_1), per al jugador 2 el parell (2, s_2) i així successivament. Sempre que un jugador 'i' hagi d'avaluar el seu polinomi per un jugador 'j' ho farà en el punt i=j.

Clarament, per a qualsevol jugador és necessari recopilar els 'n' parells (i, s_i) per a recuperar el valor secret 'k' corresponent al jugador que l'ha creat mitjançant el plantejament del sistema d'equacions corresponent de 'n' equacions amb 'n' incògnites (k, a_1, a_2, ..., a_{n-1}) del que només ens interessa el valor de 'k'.

No obstant, i donat que només ens cal conèixer el valor del terme independent –valor de 'k'- podem aplicar la interpolació de Lagrange per a un polinomi, que ens proporciona la següent fórmula:

$k = \sum_{i=1}^{n} b_i \cdot s_i$ on s_i representa el valor del polinomi avaluat en el punt i, mentre que $b_i = \prod_{\substack{j=1 \\ j \neq i}}^{j=n} \frac{j}{(j-i)}$ per a aquest cas concret en que i,j $\in$ {1,2,....., n} i on $\prod$ representa el multiplicatori dels termes. És clar que aquesta opció és molt més senzilla de calcular a partir de les avaluacions del polinomi en els diferents punts.

Amb aquest esquema aconseguim compartir un valor secret del jugador i: k_i entre tots els jugadors. Si cada jugador comparteix un valor secret k_i , tot jugador disposarà de (n-1) valors s_j j=1,.., n i j≠i corresponents a les parts del seu polinomi transmeses per la resta de jugadors

En el cas concret del jugador i, disposa de n-1 valors s_{ji} (j=1, 2, ...,n j≠i) que són els valor rebuts pel jugador i del jugador j al avaluar en el punt i el polinomi escollit per l'esmentat jugador 'j', mentre que s_{ii} és calculat pel propi jugador 'i' avaluant el seu polinomi en el punt 'i'.

$$
\begin{aligned}
s_{1i} &= k_1 + a_{11} \cdot i + a_{12} \cdot i^2 + \ldots + a_{1n-1} \cdot i^{n-1} \\
s_{2i} &= k_2 + a_{21} \cdot i + a_{22} \cdot i^2 + \ldots + a_{2n-1} \cdot i^{n-1} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
s_{ii} &= k_i + a_{i1} \cdot i + a_{i2} \cdot i^2 + \ldots + a_{in-1} \cdot i^{n-1} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
s_{ni} &= k_n + a_{n1} \cdot i + a_{n2} \cdot i^2 + \ldots + a_{nn-1} \cdot i^{n-1}
\end{aligned}
$$

Evidentment, el jugador i coneix els valors de k_i –el seu valor secret- i els diferents a_{i1}, ..., a_{in-1} –els coeficients del seu polinomi), mentre que de la resta d'equacions només coneix els diferents valors s_{1i}, s_{2i},...,s_{ni} –els valors rebuts de la resta de jugadors-.

Sumant cada jugador aquests valors rebuts i afegint el valor propi s_{ii} –que recordem és l'avaluació del polinomi propi en el punt i-, obtenim una combinació lineal de l'avaluació dels diferents polinomis de cada jugador en el punt i:

$$
\begin{aligned}
&(s_{1i} + s_{2i} + \ldots + s_{ii} + \ldots + s_{ni}) = \\
&(k_1 + k_2 + \ldots + k_i + \ldots + k_n) + \\
&(a_{11} + a_{21} + \ldots + a_{i1} + \ldots + a_{n1}) \cdot i + \\
&(a_{12} + a_{22} + \ldots + a_{i2} + \ldots + a_{n2}) \cdot i^2 + \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&(a_{1n-1} + a_{2n-1} + \ldots + a_{in-1} + \ldots + a_{nn-1}) \cdot i^{n-1}
\end{aligned}
$$

o el que és el mateix:

$$S_i = K + A_1 \cdot i + A_2 \cdot i^2 + \ldots + A_{n-1} \cdot i^{n-1} \text{ on}$$

- S_i representa l'avaluació en el punt 'i' del polinomi resultant de la suma dels 'n' polinomis. És un valor conegut per al jugador.
- K és la suma de tots els valors secrets, i és el valor que estem buscant.
- A_i és la suma dels termes de grau 'i' dels polinomis de tots els jugadors. Desconeixem el seu valor.
- 'i' és l'índex assignat al jugador 'i' i que serveix per calcular les avaluacions de polinomis a ell adreçades.

Es a dir, hem obtingut el valor del polinomi suma, que inclou la suma dels valors secrets, avaluada en el punt i., i cada jugador només coneix el valor S_i. Per tant, podem reconstruir el valor de la suma a partir de la compartició de tots els valors suma de cadascun dels jugadors (cadascú envia el seu a la resta), de nou plantejant i resolent el sistema d'equacions (n equacions, una que aporta cada jugador i n incògnites: $\{A_1, A_2, \ldots, A_{n-1}, K\}$ o be fent us de la fórmula proporcionada per la interpolació de Lagrange.

Així aconseguim que cada jugador pugui obtenir el valor de la suma de tots els valors secrets sense conèixer amb certesa cap altre valor que el propi, ja que ni les avaluacions dels polinomis individuals ni la dels valors sumats no li proporciona cap informació suficient per a determinar els valors aportats per la resta dels jugadors, en el ben entès que prèviament hem creat uns canals d'informació segur entre els diferents jugadors mitjançant un esquema de xifratge prou fort que permeti xifrada transmetre la primera avaluació, ja que en la segona no és necessari doncs com hem vist l'avaluació del polinomi suma és una informació necessàriament pública per a poder calcular el resultat final.

L'únic cas, d'altre banda inevitable, en el que els jugadors podrien arribar a conèixer el valor aportat per un jugador honest és que formessin una coalició per tal de tenir accés al conjunt de missatges que envia aquest jugador, de forma que tindrien accés a (n-1) avaluacions que deduint-les de la suma del jugador honest, els hi proporcionaria la n-èsima avaluació que els permetria plantejar el sistema d'equacions que els faria possible obtenir el valor secret aportat per aquests jugadors. Malgrat tot, el fet de utilitzar la tècnica de compromís de bit, garanteix la independència de les entrades, en el sentit que totes són aportades abans de l'aplicació de l'esquema de compartició de secrets i no poden canviar-se amb posterioritat.

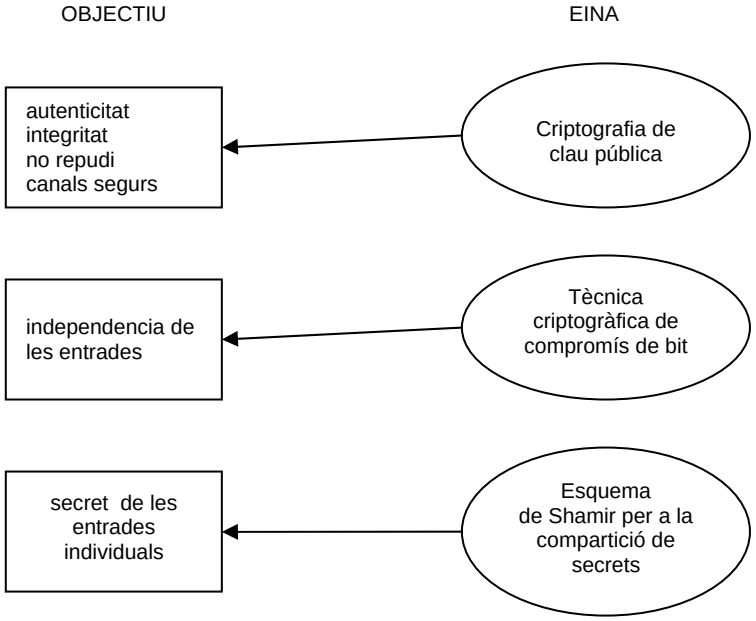
Amb aquest procés el resultat és que cada jugador haurà obtingut un número, igual per a tots, que és el resultat de la suma dels n valors secrets individuals. Aquest

número serà la llavor per a generar el número aleatori definitiu de la jugada en concret mitjançant alguna tècnica com:

- directament, calculant el seu mòdul 90 (o mòdul del nombre de números que restin per aparèixer)
- efectuar alguna transformació com per exemple diferents operacions XOR sobre un *hash* del número més un valor aleatori seleccionat al començament de la partida, i calcular el mòdul 90 abans esmentat sobre el resultat. Això, suposant que algú pogués finalment triar la suma, dificultaria que pogués fer ús d'aquesta avantatge

Finalment, indicar que l'esquema de compartició de secrets proposat per Shamir estava definit per a valors mòdul K, essent K un número primer alt. En el TFC plantejat no sembla necessari aquesta operació,, i amb les proves fetes sobre el paper no s'ha mostrat com que sigui un requisit necessari per al bon funcionament del protocol.

Resumint en un gràfic l'esquema plantejat:



Elecció i repartiment de cartrons

Una conseqüència de l'establiment del protocol de CMS per al càlcul de cada jugada és que no ha d'importar qui triï el cartró concret per a un jugador, ja que l'aleatorietat en l'extracció de cada jugada fa innecessària la preferència dels jugadors per uns números determinats, llevat de factors incontrolables com la superstició.

Per tant, i per a simplificar, considerarem que serà el servidor qui triï els cartrons per a cada jugador, això sí, de forma aleatòria, vigilant que no es reparteixin cartons repetits, i seguint unes determinades regles que semblen complir els cartrons de bingo.

Un cartró de bingo està format per deu columnes, representant les desenes compreses entre el número 1 i el número 90, i tres files, que contenint cadascuna cinc valors serveixen per a adjudicar el premi de línia al primer jugador que completi una qualsevol d'elles. Assenyalar que totes les desenes contenen deu números llevat de la primera, que conté 9 (números 1 al 9 inclosos) i la darrera que conté onze (80 al 90 inclosos ambdós).

Les regles específiques que han de complir els cartrons són:

- cada fila te 5 elements de diferents desenes.
- hi ha un o dos representants de cada desena en cada columna: com a mínim un element i com a màxim dos. Es a dir, no pot haver-hi una columna totalment plena o totalment buida.
- dintre d'una columna, els elements no es poden repetir (un mateix número no pot estar dos cops al mateix cartró) i estan ordenats de mes petit a mes gran, de forma que un valor en una columna és sempre inferior a un valor de la mateixa columna en files posteriors.

Un algorisme per a generar un cartró hauria de realitzar les següents operacions:

- crear un cartró amb 3 files i 9 columnes, per tant 27 posicions. En cada posició pot haver un blanc (zero) o contenir un valor que necessàriament ha de pertànyer a aquella desena concreta.
- per a la fila 1 i 2, seleccionar de forma aleatòria 5 desenes (columnes) per a cadascuna, i de cada desena escollir aleatòriament un valor que hi pertanyi i que no s'hagi escollit encara en aquest cartró
- per a la fila 3, primer hem de fer un recorregut pel cartró per a detectar les columnes que encara no tenen cap valor. Necessàriament, hem d'escollir un valor aleatori per a aquestes en la tercera fila i que seria de quatre com a màxim, suposant que totes les columnes haguessin coincidit en les dos columnes anteriors (cinc coincidències deixarien quatre columnes 'orfes'). Un cop fet això i amb les columnes restants, farem el mateix que hem fet per a les files 1 i 3 en l'apartat anterior, llevat que no serà possible agafar valors de columnes que ja tinguin dos valors assignats, es a dir, amb valors en les files primera i segona.
- el pas final, i per a complir el requisit de l'ordre creixent dels valors dintre de les columnes consistirà en fer un recorregut del cartró intercanviant posicions de files dels valors d'aquelles columnes que tinguin dos i el valor mes alt estigui en una fila superior.

Un cop construït el cartró, ja estem en disposició d'enviar-ho al jugador. En aquest punt hem de tenir en compte les següents consideracions de seguretat:

- suposarem que els jugadors són molt desconfiats i per tant no volen que ningú mes que ells (i evidentment el servidor) conegui el contingut del seu cartró. Per tant, el servidor enviarà el cartró en un determinat format, xifrat amb la clau pública del jugador receptor, i amb una signatura digital del propi servidor de forma que només el jugador destinatari pugui desxifrar-ho i comprovar l'autenticitat i la integritat del cartró.
- arribats a aquest punt, se'ns presenta un problema. Podria donar-se el cas que un jugador maliciós recopilés cartrons d'altres jocs previs, i que els presentés com correctes en altres jocs iniciats pel mateix servidor, si aquest no ha modificat la seva clau. La solució a aquest problema consisteix en afegir un valor addicional a cada text on s'hagi d'aplicar una signatura digital i que sigui únic per a cada partida.

  Un possible valor d'aquest tipus podria ser un valor aleatori comú a tots els jugadors, que en la nostra aplicació podríem seleccionar per exemple mitjançant l'execució del protocol de CMS per a l'elecció d'un número aleatori, de forma que aprofitaríem la suma obtinguda per a fer el paper de l'esmentat valor. Així, encara que un jugador col·leccionés cartrons de diferents partides no li farien cap servei, doncs els mateixos cartrons proporcionarien signatures diferents en diferents partides i només serien vàlids per a la partida en que van ser generats. Per a una seguretat mes gran, podríem afegir a aquest identificador la data en que la partida es va jugar.

Comprovació i verificació de premis

En un entorn de desconfiança mútua com el que hem plantejat en l'especificació del problema, no es pot permetre que el Servidor –tot i que ha generat i coneix els cartrons de tots els participants- actuï com a verificador i notari en el moment que algun jugador assoleixi un premi.

D'altre banda, donat que en el repartiment dels cartrons hem establert que són secrets per a tots els participants menys per al propietari i per al Servidor, haurem d'establir un mecanisme per tal que un jugador que canti línia o bingo, pugui demostrar que realment està en possessió d'un cartró guanyador.

Aquest mecanisme pot ser una signatura digital de cada línia que el servidor crearà amb el cartró, i que signarà amb la seva clau privada. Així, el servidor repartirà al jugador propietari el cartró i la signatura digital de cada línia, mentre que a la resta de jugadors els farà arribar només les signatures digitals de les línies del cartró, de forma que no podran conèixer la composició del cartró.

Aleshores el jugador que canti línia podrà demostrar que el seu premi és correcte enviant un missatge amb el detall de la línia –en clar, sense xifrar- i la signatura digital que d'aquesta línia li va proporcionar el servidor al repartir el cartró. Com el jugador disposa de la clau pública del servidor i coneix el format que aquest ha fet servir per a generar la signatura, podrà comprovar la seva validesa i la del premi.

Anàlogament en cas de 'bingo' el jugador ho demostrarà amb la signatura digital del cartró sencer, es a dir, la signatura digital de cadascuna de les línies que el composen. El jugador que vulgui verificar el cartró, ho farà verificant la signatura de les tres línies fent us de la clau pública del servidor.

Disseny de l'aplicació

Òbviament, l'aplicació desenvolupada es composa de dos parts independents: la part client, corresponent als jugadors i la part servidora, amb la funcionalitat de generar i repartir els cartrons i servir d'enllaç a les comunicacions entre els clients.

En aquest punt l'estructura d'objectes que requeria l'aplicació no semblava tan complicada com perquè les diferents eines de l'Enginyeria del Programari i fonamentalment els diagrames UML fossin d'utilitat en el seu desenvolupament., almenys de forma exhaustiva i seguint els mètodes d'enginyeria que aquesta disciplina proposa.

No obstant això, i amb la finalitat de facilitar la posterior implementació va resultar d'utilitat confeccionar:

- un guió pel servidor i un altre pels clients o jugadors, detallant el seu comportament i responsabilitats en cadascun dels moments del joc.
- un guió conjunt per ambdós , especificant el funcionament a baix nivell del protocol de càlcul multipart segur d'un nombre aleatori, explicat en l'apartat on s'ha desenvolupat l'esquema criptogràfic.
- un diagrama de col·laboració simplificat , per visualitzar l'estructura d'objectes de l'aplicació i la seva interrelació.

Guió del servidor

1. Crear Partida

   1.1. Es crea un socket servidor (*Server Socket*) en el port 2000 amb la finalitat d'esperar connexions de jugadors.

   1.2. Els jugadors envien un missatge de petició de connexió, contenint el seu certificat X509 que conté la seva clau Pública i que faran servir la resta de participants en l'esquema criptogràfic del joc. Al rebre aquesta petició, se li assigna un número –començant per 1 jugador 1, següent jugador2 i així successivament -, s'emmagatzema el certificat i finalment es crea un *socket* per intercanviar missatges entre el servidor i aquest jugador.

   1.3. El servidor comunica al client el nombre que li ha estat assignat mitjançant missatge al Jugador fent ús del *socket* prèviament obert i que inclou a més el certificat X509 del propi servidor, per tal que el client l'emmagatzemi per poder-lo usar amb posterioritat en la verificació de signatures digitals creades pel Servidor.

   1.4. Quan el nombre de jugadors que han demanat participar en la partida arriba al nombre introduït com a paràmetre en l'aplicació, el servidor passa a l'estat d'iniciar partida.

2. Iniciar partida

   2.1. El servidor comunica l'inici de la partida a cada jugador apuntat, enviant-li la llista complerta dels jugadors, amb el certificat X509 de cadascú que conté la seva clau pública.

   2.2. S'executa el protocol de generació aleatòria d'un número mitjançant la participació de tots els jugadors. Aquest número servirà per a identificar la partida, amb la finalitat de que ningú pugui recollir cartrons jugats en una partida anterior i presentar-les com si fossin d'aquesta.

   2.3. Per cada jugador, es crea aleatòriament un cartró seguint les regles de construcció que ha de complir. El servidor crea una signatura digital de cadascuna de les tres línies en combinació amb el número aleatori seleccionat en la ronda prèvia i que serveix d'identificador de la partida. El servidor signa cadascuna d'aquestes línies amb la seva clau privada, de forma que un jugador que presenti la signatura juntament amb la línia que la va originar, podrà demostrar a qualsevol altre jugador la seva autenticitat e integritat, sempre que aquest jugador disposi de la clau pública del servidor i 'confiï' en l'autoritat certificadora.

   2.4. El servidor envia el cartró de joc (tres línies i les seves corresponents signatures) a cadascun dels jugadors , xifrant aquesta informació amb la clau pública del jugador propietari, de forma que només ell pugui obtenir els nombres que composen el cartró.

   2.5. Cadascun dels jugadors dona la seva conformitat a la recepció del cartró.

   2.6. Un cop el servidor disposa de totes les conformitats, envia un missatge a cadascun dels jugadors contenint les signatures de cadascun dels cartrons. D'una banda, el repartiment de les signatures dels cartrons evita que el servidor o algú altre pugui generar un cartró vàlid generat amb posterioritat a l'inici de la partida i a mida dels diferents números aleatoris obtinguts fins aquell moment. D'altre banda, el fet de disposar de les signatures permet als jugadors comprovar premis obtinguts i declarats per altres jugadors.

3. Desenvolupament de la partida

   3.1. Execució del protocol de càlcul segur multipart d'un nombre aleatori fins el moment en que algun dels jugadors completa el seu cartró i obté un 'Bingo'. Sincronització dels missatges enviats pels jugadors com a conseqüència de l'execució del protocol.

4. Finalitzar partida

   4.1. després que un jugador ha cantat bingo, el servidor –i la resta de jugadors pel seu compte- comproven la correcció del premi, i si és vàlid, comunica als jugadors la fi de la partida i tanca els *sockets* corresponents.

Guió del jugador

1. Unir-se a partida
   1.1. S'efectua una petició de connexió al port 2000 del servidor, per unir-se a una partida creada per aquest.
   1.2. Envia un missatge inicial amb el certificat de clau pública.
   1.3. Es rep la resposta del servidor amb el número assignat dins de la partida i el certificat X509 corresponent al servidor.
   1.4. Un cop tots els jugadors s'han connectat, es rep missatge del servidor comunicant l'inici de la partida i incloent la llista complerta dels jugadors amb els diferents certificats X509 que contenen la seva clau pública.
   1.5. S'inicia el protocol de generació aleatòria d'un número, amb la computació de la funció suma per part de tos els jugadors i de forma segura. Aquest número es comunica al servidor –que comprovarà que rep el mateix de tothom- i servirà per a individualitzar qualsevol signatura digital aplicada a elements d'aquesta partida.
   1.6. Es rep el cartró, les línies individuals i les corresponents signatures digitals generades pel Servidor. S'envia al servidor conformitat de recepció del cartró, prèvia comprovació de les signatures digitals rebudes amb el cartró.
   1.7. Es rep del servidor un missatge amb les signatures digitals de cadascuna de les línies de tots els jugadors.
2. Desenvolupament de la partida
   2.1. De forma successiva i fins que algun jugador canti bingo:
      2.1.1. S'executa el protocol de computació multipart per a la generació de forma segura d'un número aleatori.
      2.1.2. Es comprova si s'ha produït o no algun premi en el propi cartró, comunicant-ho mitjançant un missatge a la resta de jugadors. En cas d'haver obtingut un premi s'adjunta en el missatge la línia(es) premiada(es) per que la resta de jugadors puguin verificar la correcció del premi.
      2.1.3. Es rep les comunicacions de premi de la resta de Jugadors , comprovant la validesa si s'escau.
      2.1.4. S'actualitzen els valors presentats a la interfície gràfica del client: cartró, llista de nombres i missatges del joc.
3. Acabament de la partida
      3.1.1. Havent cantar bingo un jugador, es comprova la correcció del premi, tancant els *sockets* i finalitzant l'aplicació

Guió del protocol de càlcul segur multipart d'un nombre aleatori

En aquest protocol , la part activa és composada pel conjunt de clients. En l'aplicació desenvolupada, s'ha assignat al servidor el paper de recollidor i distribuïdor de missatges, sense capacitat per modificar en cap cas el seu contingut llevat de l'adreçament.

1. El jugador tria un número secret, que serà la seva aportació individual al resultat desitjat de la suma conjunta. Aquest nombre ha de ser prou alt –per exemple de l'ordre de 2^{128} - per fer-lo segur contra atacs per 'força bruta' en cartrons 'col·leccionats' d'altres partides. Aquest nombre farà el paper de terme independent del polinomi.
2. El jugador tria els (n-1) coeficients del seu polinomi, essent 'n' el nombre de Jugadors que participen a la partida. Aquests coeficients poden ser positius o negatius i fins i tot nuls, encara que no és aconsellable. El seu ordre no cal que sigui tan gran com el del valor secret, però igualment a més ordre, més dificultats per possibles atacs.
3. El jugador avalua el seu polinomi en els punts corresponents als índex assignats als jugadors al principi de la partida: 1, 2, 3, ..., n i obté 'n' nombres enters, un per cada jugador inclòs ell mateix. Aquestes avaluacions fan el paper de compromisos dins de la tècnica criptogràfica del compromís de bit.
4. El jugador genera de forma aleatòria una clau simètrica de sessió, aplicable només en l'execució un sol cop del protocol.
5. El jugador xifra les avaluacions obtingudes amb la clau simètrica i amb la seva clau privada genera la signatura digital d'aquestes avaluacions. El parell avaluació xifrada – signatura digital el xifra amb la clau pública del jugador destinatari, de forma que només ell pugui accedir al seu contingut.
6. El jugador 'i' envia en un mateix missatge les 'n-1' avaluacions del seu polinomi corresponents a la resta de jugadors: $j=1, 2, ..., n \quad j \neq i$. amb el format detallat en el punt anterior. L'ordre de les parts dins del missatge sempre és coincident amb l'ordre dels índex assignats als jugadors a l'inici de partida, de forma que sempre sigui possible conèixer l'origen d'un missatge tot i no especificar-ho.
7. El servidor recull tots aquests missatges d'avaluacions. Separa les diferents avaluacions rebudes en les seves parts i les agrupa per destinatari. Aleshores envia un missatge a cadascun dels jugadors amb les avaluacions a ell adreçades, en el format indicat en el punt 5.
8. El jugador recull aquest missatge i el separa en les seves parts. Com sap qui és el remitent –l'índex ve marcat per l'ordre de les parts- aleshores desxifra el contingut amb la clau pública d'aquest remitent, obtenint les avaluacions dels polinomis de la resta de jugadors en el punt corresponent al seu índex, i que fan el paper de compromisos xifrats i les signatures digitals d'aquestes avaluacions que garanteixen la seva integritat i origen.
9. Un cop rebuts els compromisos, els jugadors 'alliberen' les claus de sessió que faran possible el seu desxifrat. S'envia un missatge al servidor amb la clau de sessió, sense xifrar perquè és una informació pública per a tots els jugadors.
10. El servidor recopila aquests missatges i ajuntant-los en un de sol, el remet a tots els jugadors.
11. Amb les claus de sessió els jugadors desxifren els compromisos, es a dir, les avaluacions dels diferents polinomis en el punt corresponent al seu índex.

Després comproven amb la clau pública del jugador remitent que les signatures digitals rebudes siguin congruents amb l'avaluació desxifrada.

12. Si tot és correcte, el jugador 'i' acumula totes les sumes rebudes afegint-hi l'avaluació del propi polinomi en el seu punt. S'obté l'avaluació en el punt 'i' del polinomi conjunt que és combinació lineal dels polinomis individuals. Aquest valor ha de compartir-se amb la resta de jugadors de forma que puguin calcular el resultat desitjat.
13. El jugador genera un missatge amb aquesta suma d'avaluacions. Al ser d'informació pública no cal que sigui xifrada, però sí que s'adjunta la seva signatura digital de forma que es garanteixi el seu origen i integritat.
14. El servidor recopila aquests missatges, creant un de sol a partir d'ells i remetent-lo als jugadors
15. Els jugadors recullen aquest missatge i prèvia comprovació de les signatures digitals dels components, apliquen la fórmula de Lagrange per a obtenir el resultat final: la suma de les entrades individuals que s'han mantingut en secret i que servirà de llavor del número aleatori
16. Els jugadors i el servidor calculen el nombre aleatori finalment extret. Tot i que és possible complicar la seva obtenció tant com es desitgi, finalment s'ha optat per la solució senzilla de fer:

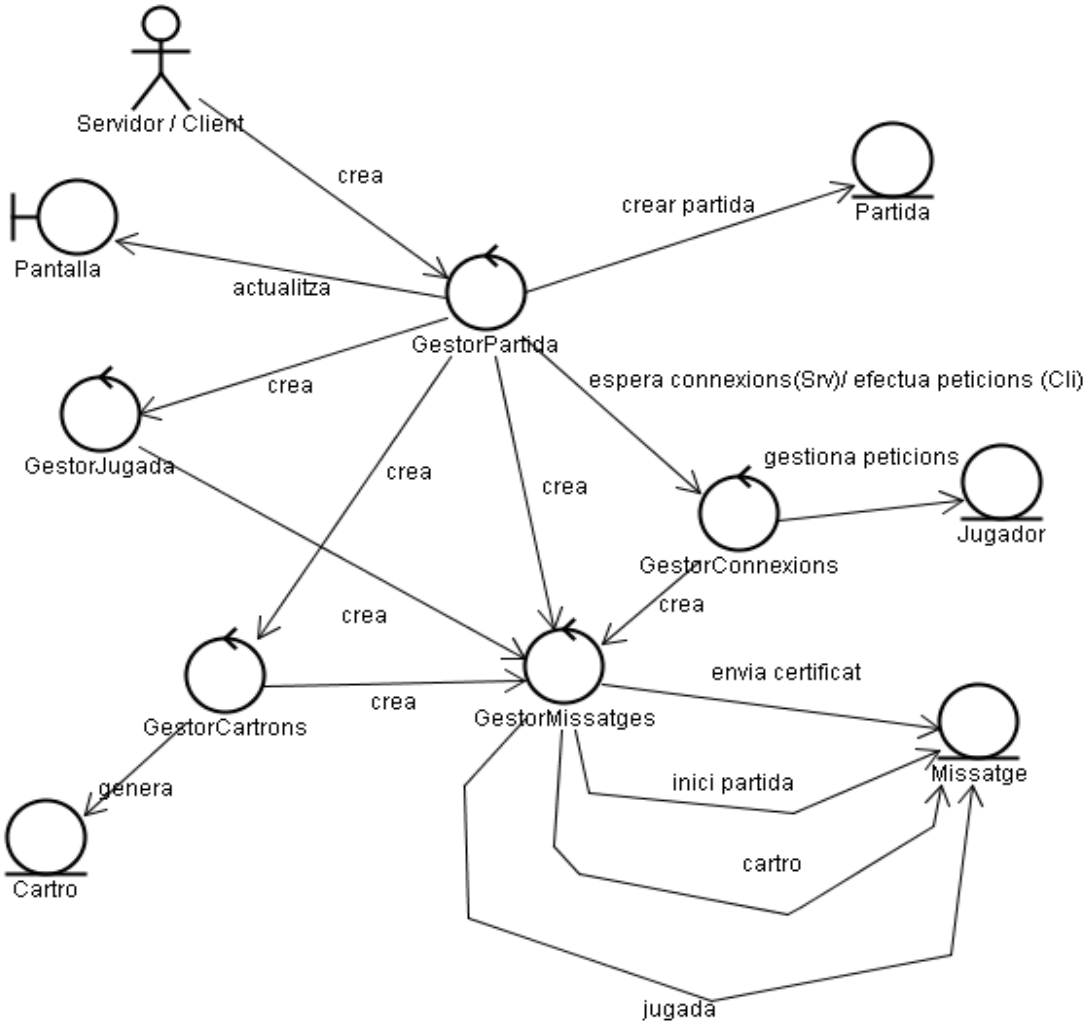
    nombre aleatori = (suma multipart segura) mod (nombres restants per sortir)

    Com que les entrades individuals no s'han escollit en mòdul congruent amb el número de nombres restants per sortir, aquesta forma de càlcul genera un petit biaix cap als nombres mes baixos, però aquest biaix és insignificant en nombres amb ordres de magnitud grans.

Estructura de l'aplicació

El diagrama de col·laboració simplificat proporciona una visió conjunta de les classes que composen l'aplicació i la seva interrelació.

En l'aplicació desenvolupada, el diagrama aplicable a la part servidora i a la part client pot fusionar-se en un de sol, ja que tot i les diferents responsabilitats, ambdós presenten un comportament simètric, llevat de la gestió de Connexions que en el cas del servidor és qui engega la gestió de la partida miutjançant la creació de diferents 'Threads'.



El disseny del programari s' ha basat en els tres nivells de classes que contempla l'enginyeria de software:

- classes de frontera, que gestionen la interacció de l'aplicació amb l'usuari i gestionen la presentació de resultats en pantalla . Donada la senzillesa de l'aplicació només hi figura la classe Pantalla.
- classes d'entitats, que modelen objectes del domini que en el nostre cas és un joc de Bingo. Són les classes: Cartró, Jugador, Partida, i Missatge.
- classes de control, que duen a terme la part més important de l'aplicació, actuant sobre les classes d'entitats i les classes de frontera. Són les classes: Gestor Connexions, GestorCartrons, Gestor Jugada, GestorPartida, i GestorMissatges.

Descriurem breument cadascuna d'elles.

Classes d'entitats:

- *Cartró*. Modela un cartró de Joc de Bingo. Els seus atributs són:
    - Un array d'enters de dos dimensions on la primera representa la fila i la segona la columna. En cada posició pot haver un '0' si està buida o un nombre enter en el interval [1,90] altrament.
    - Un altre *array* de booleans de dos dimensions i de la mateixa mida que serveix per a indicar si el número que hi ha a aquella posició ha sortit a la partida o no.
    - un *array* de *byte*[] de tres posicions emmagatzema les signatures digitals que el Servidor ha generat en la seva creació.

  Entre altres disposa de mètodes per a la generació aleatòria de cartrons i per a la generació i comprovació de les signatures digitals associades a les línies que el composen.
- *Jugador*. Encapsula la figura d'un jugador de bingo. Com atributs principals presenta:
    - un objecte Cartró, que representa el cartró de joc amb el que participa a la partida.
    - un objecte *Socket*, que serveix per a emmagatzemar el socket de comunicacions amb el servidor en cas de la part client i els sockets de comunicacions amb els diferents jugadors en cas de la part servidora de l'aplicació.
    - un objecte X509Certificate, que conté el seu certificat X509 on s'emmagatzema la seva clau pública.

  Els seus mètodes es limiten a proporcionar accés o donar valor als atributs que emmagatzema.
- *Partida*. Modela els trets d'una partida de Bingo i és possiblement l'entitat central de l'aplicació i paràmetre d'entrada a la majoria de mètodes de les classes de control. Els atributs més destacats amb els que compte:
    - un enter que conté el nombre de jugadors que participen en la partida.
    - un *array* d'objectes Jugador, de mida igual al nombre total de Jugadors que participen en la partida més un. Com ja s'ha explicat en el protocol, per comoditat de càlcul s'ha assigna a cada jugador un nombre enter, que a mes serveix per accedir als atributs del jugador en aquest array. L'índex '0' està reservat al servidor i és on els clients emmagatzemen les seves dades d'accés – *socket* i certificat – com si fos un jugador més.
    - un objecte *BigInteger* que representa el identificador únic de la partida en curs i que serveix per a juntament amb el seu contingut, signar digitalment les línies dels cartrons.
    - un array de booleans que en un moment donat i per cadascun dels possibles nombres en l'interval [1,90] indica si han sigut extrets o no en la partida en curs.
    - un enter que compta els nombres que resten per sortir en la partida en curs i que es fa servir per al càlcul del nombre aleatori, aplicant la

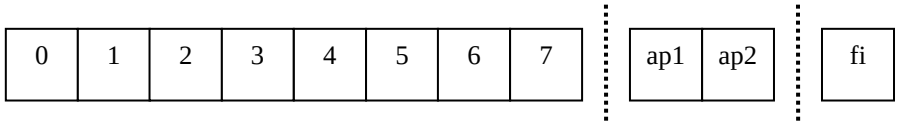
operació mòdul nombres restants al resultat d'executar el protocol de càlculs segur multipart.

- un *array* d'enters de mida variable que emmagatzema els nombres que resten por sortir. A mida que els nombres són extrets es treuen d'aquest objecte.
- un enter que representa l'índex assignat al jugador participant de la partida. Òbviament, aquest atribut només és d'utilitat per a la part client permetent 'saltar' el propi objecte Jugador en els recorreguts que afecten a tots els Jugadors.

Els mètodes que conté són d'accés i donar valors als seus atributs, llevat del mètode que a partir del resultat del protocol i els nombres restants extreu el nombre resultat de la jugada.

- *Missatge*. Representa l'entitat que gestiona i guarda el contingut dels missatges de comunicació entre la part servidora i la part client com a conseqüència de successos originats en l'execució de l'aplicació. Tots els continguts que s'han d'enviar es codifiquen en *bytes* i són encapsulats dins d'un objecte missatge.

  El seu únic atribut és un *array* de *byte[]* de mida variable en funció del contingut del missatge. El seu format consta d'una capçalera i un o més continguts:



Els *bytes* 0 i 1 representen el número de jugador origen del missatge. Els *bytes* 2 i 3 representen el jugador destí del missatge. En ambdós casos un valor de zero indica que es tracta del servidor.

El *byte 4* conté el tipus de missatge, si és de comunicació o resposta . En una futura versió podria també codificar conté un error. El *byte* 5 codifica el codi del missatge segons el succés que l'hagi generat i que indica el seu contingut: enviament de certificats, de claus de sessió, d'avaluacions de polinomi, de resultat de protocol, etc.

Els *bytes* 6 i 7 contenen un punter. Poden apuntar al final del missatge si aquest només transporta un contingut (per exemple el certificat que el servidor envia inicialment), o al següent apuntador en cas d'un missatge que transporti més d'un contingut (per exemple les claus de sessió alliberades pels jugadors i que el servidor ha recopilat per enviar-les conjuntament), de forma que puguin separar-se tots ells. En cas d'un missatge de contingut múltiple, el darrer punter apunta a la part final del missatge per a indicar que no hi ha més contingut.

Els mètodes d'aquesta classe donen accés als elements de la capçalera i permeten extreure en forma d'*array* de *bytes* els diferents continguts transportats.

Classes de control

La funcionalitat d'aquestes classes, tot i que simètrica varia lleugerament segons si correspon a la part servidora o a la part client de l'aplicació:

- *GestorConnexions*. En la part servidora, crea un objecte *ServerSocket* en el port 2000 per tal d'acceptar connexions dels clients i generar el corresponent *socket* de comunicacions.

  En la part client, crea un *socket* adreçat al port 2000 del servidor per sol·licitar la connexió a la partida. Un cop connectat, envia el missatge inicial amb el certificat i rep el missatge de resposta amb el certificat del servidor.

- *GestorCartrons*. En la part servidora te com responsabilitat el generar els cartrons per a cadascun dels jugadors , signar-los amb la seva clau pública i repartir-los a cadascun dels propietaris. Un cop repartits els cartrons, reparteix a tots els jugadors les signatures digitals de tots els cartrons.

  Per la seva part, la part client s'encarrega de rebre el cartró de joc i verificar la seva signatura digital. A més, rep les signatures de tots els cartrons de la resta de jugadors i els emmagatzema en l'objecte Cartró de cadascun d'ells.

- *GestorJugada*. En la part client, encapsula l'execució del protocol de càlcul segur multipart de la suma de les entrades individuals dels jugadors, tant a efectes d'obtenir el identificador de la partida com la llavor per extreure el nombre aleatori de la jugada. Conté dues constants que emmagatzemen el número de bits dels nombres a generar i dels coeficients del polinomi. L'aplicació distribuïda s'ha compilat amb un valor de 128 bits per a tots dos.

  La part servidora executa les tasques simètriques en el protocol: recopilació i remissió d'avaluacions de polinomi, de claus de sessió, de sumes parcials, etc., a més del calcular amb el mateix resultat que el client l'identificador de la partida o el nombre aleatori segons sigui l'inici de la partida o estigui en curs.

- *GestorPartida* Tant en la part servidora com en la part client crida a l'execució de les tasques de creació de la partida. Un cop iniciada la partida, s'encarrega de l'execució de les successives jugades actualitzant la interfície de pantalla, i comprovant en el cas del Client si s'ha obtingut línia o bingo i comunicant aquesta circumstància a la resta de jugadors.

  La part servidora s'encarrega de recopilar els missatges dels jugadors en un sol missatge que s'envia de nou a cadascun dels jugadors, per tal que aquests puguin saber si algun altre jugador ha obtingut premi en la jugada.

- *GestorMissatges*. Té com a responsabilitat assignada l'enviament i manipulació de tots els missatges generats per l'aplicació. La seva estructura és simètrica en la part servidora i en la part client, si hi ha un mètode que envia un missatge en una part, en l'altre hi ha el complementari que el rep i processa.

Classes frontera

- *PantallaCli*. Aquesta classe encapsula la interfície de l'aplicació amb el jugador. La pantalla es construeix amb un objecte que estén la classe *JFrame* proporcionada per Java que fa de contenidor de les tres àrees diferenciades:
    - un objecte *JPanel* que construeix el cartró de Joc. Aquest objecte està format al seu torn per un *array* d'etiquetes *JLabel[3][9]*, que es corresponen amb les posicions del cartró i que contenen un número si en la mateixa posició del cartró n'hi ha un número i una icona si en la posició corresponent del cartró hi ha un blanc.
    - un altre objecte *JPanel* que construeix l'àrea de nombres del cartró. Aquest objecte està composat per un *array* d'etiquetes *JLabel[90]* paral·lel a l'*array* de booleans de l'objecte Partida que emmagatzema tots els possibles números i si han estat o no extrets a la partida.
    - finalment un objecte *JTextArea* que dibuixa un àrea de text, que permet a la interfície rebre els missatges de text que enviï l'aplicació. Aquest objecte es passa com paràmetre en la creació d'un objecte *JScrollPane* perquè pugui disposar de les barres de desplaçament quan el nombre de línies excedeixi la mida de l'àrea

A mesura que es desenvolupa el joc, el resultat de cada jugada produeix una actualització en la visualització de la pantalla del Jugador.

L'objecte Pantalla rep un missatge de l'objecte Partida amb el text que ha de presentar i un altre missatge amb el nombre resultant de la jugada i la posició del cartró marcada en cas que aquest nombre figuri en el cartró, i si no figura passa com a paràmetre l'indicador -1 en fila i columna.

L'objecte Pantalla processa aquests missatges:

- escrivint el text en l'àrea de text.
- ombrejant amb el color de fons l'objecte *JLabel* de la posició corresponent al número resultat de la jugada dins del pannell de números.
- ombrejant amb el color de fons el l'objecte JLabel corresponent a la posició marcada del cartró en que hi figura el número. Si el número no està dins del cartró, simplement no fa res.
- en cas que de la jugada s'esdevingui un premi, aleshores s'ombreja amb un color diferent la línia o línies que han resultat premiades.

Pot apreciar-se una mostra gràfica del resultat en l'apartat següent on es detalla el funcionament de l'aplicació.

Manual del usuari

Instal·lació i execució

L'aplicació s'ha desenvolupat mitjançant el llenguatge de programació Java en la versió del SDK 1.5.0.03

A partir del codi font, el compilador Java crea un pseudocodi que és interpretat per la màquina virtual Java. Caldrà assegurar-se doncs que l'ordinador on s'hagi d'instal·lar i executar l'aplicació –client o servidor- disposi d'aquest dispositiu de programari que pugui interpretar el pseudocodi.

En el moment d'elaborar aquest TFC la Màquina Virtual Java està inclosa en les següents distribucions gratuïtes:

- *J2SE Software Development Kit (SDK).* Proporciona l'entorn que possibilita el desenvolupament d'aplicacions Java
- *The J2SE Java Runtime Environment (JRE).* Permet als usuaris finals executar aplicacions Java, a partir de fitxers compilats (extensió *.class)

Ambdós distribucions es troben en la versió 1.5, però donada la senzillesa de l'aplicació no hauria d'haver-hi problemes per executar l'aplicació amb versions 1.4 o superiors.

Un cop verificada aquesta condició, la instal·lació és molt simple, només cal descomprimir els fitxers de pseudocodi (extensió *.class) en una carpeta qualsevol, vigilant que en el cas del client hi hagi en el mateix directori el fitxer '*blancCartro.jpg'* que conté la imatge que es dibuixa al cartró en el cas d'una casella buida.

Un altre aspecte necessari per al correcte funcionament de l'aplicació en la part Client i la part Servidor és la presència de les llibreries IAIK en un directori que pengi del directori on resideix l'aplicació. Aquestes llibreries estan incloses en la distribució de les aplicacions –tant Servidor com Client- que en ser descomprimides, crearan els directoris i fitxers necessaris.

Finalment, com a requeriment previ per poder executar l'aplicació cada participant –Client i Servidor- ha de disposar d'un parell de claus RSA Privada - Pública, doncs l'aplicació fa un ús intensiu del xifrat RSA i la signatura digital.

Amb el fitxer de distribució s'acompanyen cinc parells de fitxers de Certificat X509 autosignats i clau privada amb longitud de 2048 bits (servidor.crt, servidor_pvk jugador1.crt jugador1_pvk, ..) pel ús de qualsevol d'ells en la seva part client o servidora de l'aplicació, ja que el nom dels fitxers és purament indicatiu.

Com s'ha especificat a l'apartat de l'esquema criptogràfic l'aplicació en aquesta versió dona per bona qualsevol infrastructura de clau pública que li sigui presentada –no comprova si el certificat està signat per una CA (*Certificate Authority*) de confiança per al usuari, si es vàlid o si ha estat revocat- i només requereix que el parell clau privada – clau pública siguin correctes, ja que altrament l'aplicació llençaria una excepció al provar de desxifrar o comprovar una signatura digital.

L'execució és també molt senzilla, obrint una finestra de comandes i situats en el directori on s'hagin col·locat els fitxers *.class:

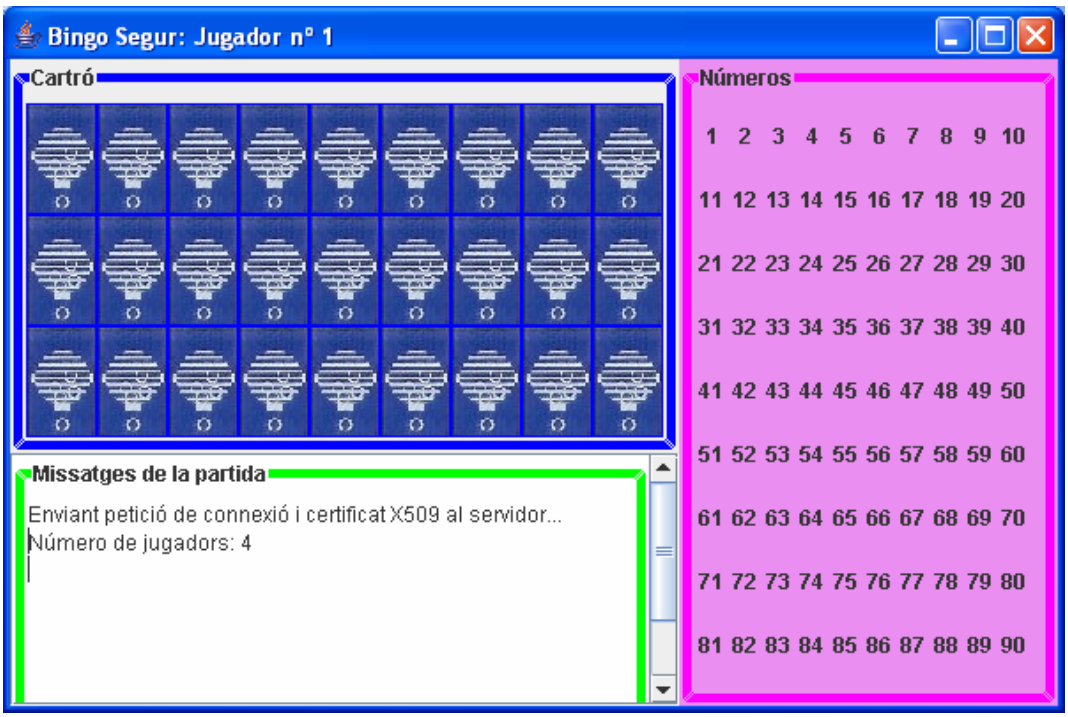
- en el cas del client teclejarem: *java BingoSegurClient [adreça IP del servidor] [ruta al fitxer que conté el certificat X509] [ruta al fitxer que conté la clau privada RSA]*
- en el cas del servidor l'ordre serà: *java BingoSegurServidor [nombre de Jugadors] [ruta al fitxer que conté el certificat X509] [ruta al fitxer que conté la clau privada RSA]*

S'ha de tenir en compte que en la versió actual de l'aplicació és condició necessària que hagi un servidor 'escoltant' peticions per tal que els clients puguin ser executats. Altrament es generarà una excepció de 'connexió rebutjada' i finalitzarà l'aplicació client.

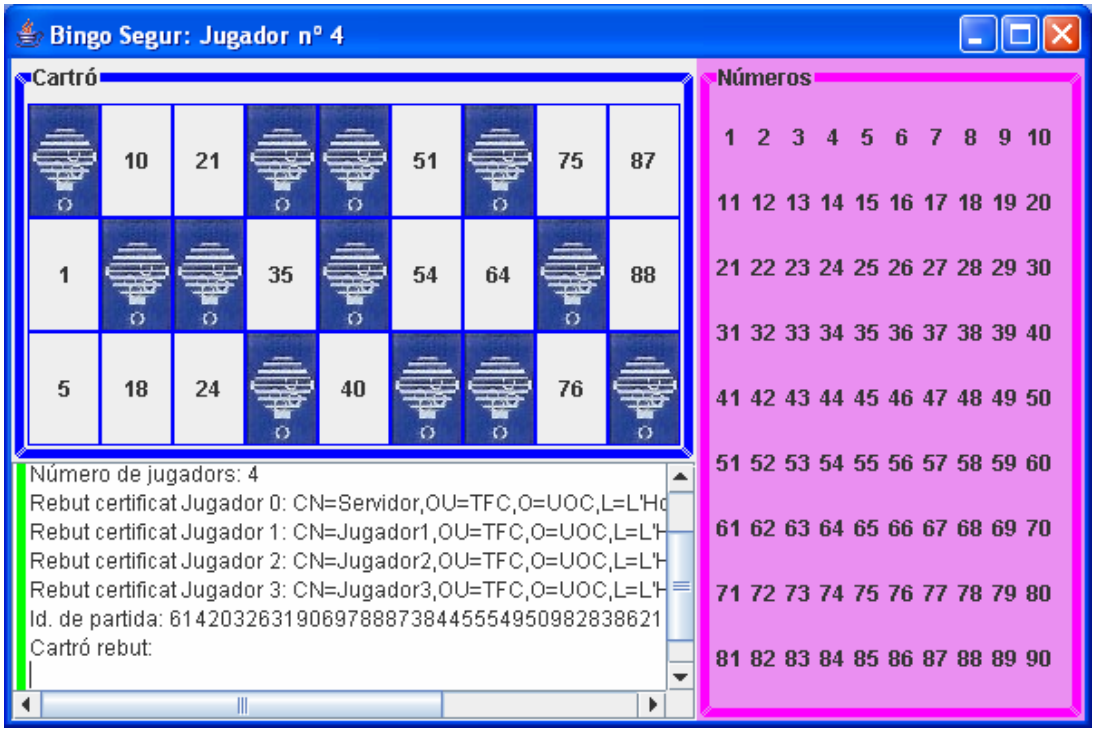
L'execució del servidor no engega cap interfície gràfica i els missatges que genera com a conseqüència del seu funcionament són bolcats directament a la consola. Tampoc cal la intervenció de l'usuari, doncs els torns del joc es van executant seqüencialment fins a la fi del joc.

El procés d'execució del client tampoc requereix de la intervenció de l'usuari, ja que de forma simètrica al servidor els torns van succeint-se fins a que algun dels jugadors complerta el cartró . No obstant, i a diferència de la part servidora, l'execució del client sí engega una interfície gràfica amb tres àrees diferenciades: a dalt i a l'esquerra el cartró de joc, en la part baixa i a l'esquerra un àrea on l'aplicació desa els missatges de sortida i finalment a la dreta un àrea amb tots els possibles números.

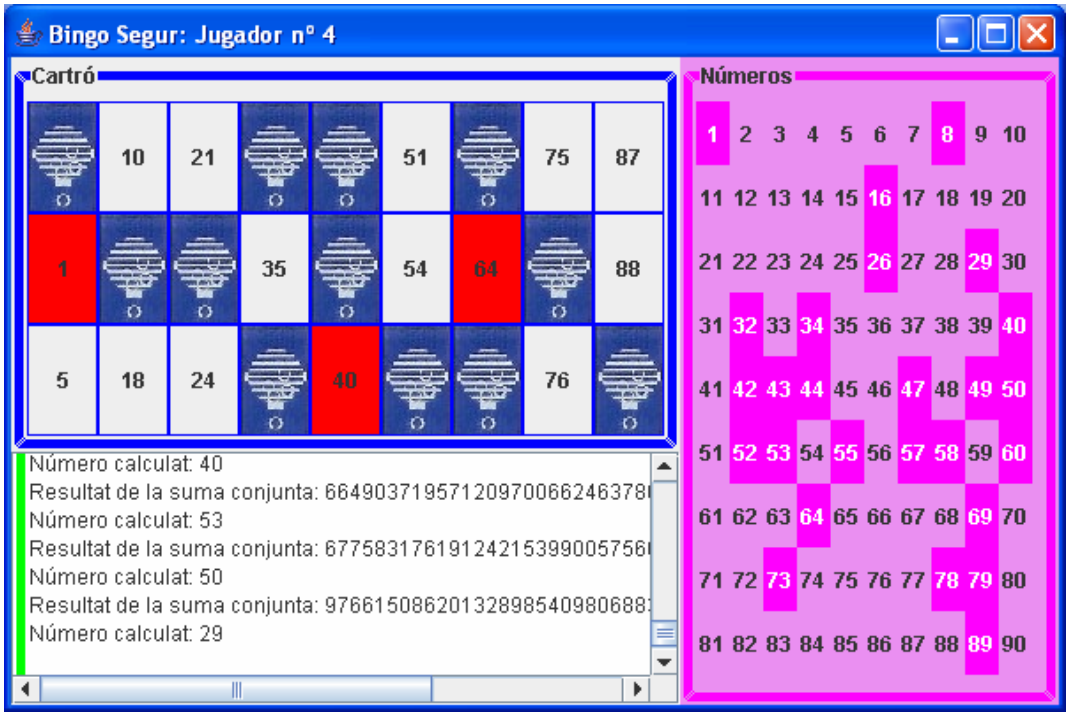
En el seu inici, i en tant no s'han connectat tots els jugadors de la partida, el cartró es presenta en blanc, sense nombres.



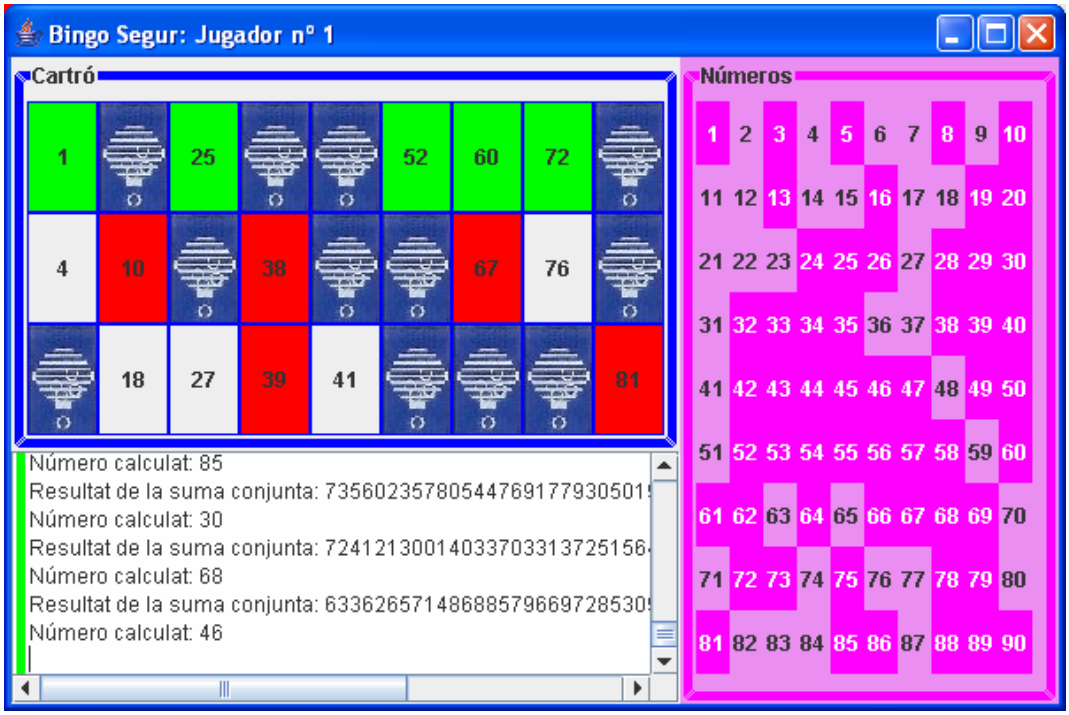
Un cop connectats els jugadors, el servidor genera i reparteix un cartró a cadascun dels jugadors, que és dibuixat a la pantalla. A l'àrea de text es llisten els noms distingits dels propietaris dels certificats rebuts i el identificador de la partida obtingut a partir de l'execució inicial del protocol.



A partir d'aquest moment, comencen a generar-se els nombres de cada jugada com resultat de la execució successiva del protocol de càlcul multipart. Á l'àrea de text es presenta la suma conjunta resultat del protocol i el número aleatori obtingut a partir d'ella, i s'actualitzen les caselles del cartró i del pannell en funció d'aquests nombres.



En cas que aparegui algun premi, l'aplicació ombreja amb un color diferent la línia o línies afectades.



Un cop que algun dels jugadors completa el seu cartró, l'aplicació Servidor finalitza automàticament, tornant a la línia de comandes. L'aplicació Client acaba quan el usuari tanca la finestra.

Desinstal·lació de l'aplicació

Donada les característiques de l'entorn Java, per desinstal·lar l'aplicació en qualsevol de les seves versions només cal esborrar tots els fitxers i directoris que la composen.

Proves de funcionament efectuades

En el cas concret de l'aplicació objecte d'aquest TFC no hi ha dades d'entrada que hagin de ser processades i conseqüentment no hi ha un possible plantejament de 'joc de proves' que permeti comprovar el seu correcte funcionament.

Per tant, hagués sigut necessari i suficient provar que en qualsevol entorn de xarxa –preferiblement un entorn WAN amb Internet al mig – podia executar-se l'aplicació servidor en una de les estacions, acceptant connexions d'almenys quatre clients normalment executats en altres estacions del mateix entorn o un altre que hi estigui connectat mitjançant la xarxa.

Essent aquest l'entorn ideal de proves, plantejava no obstant el problema de la dificultat de trobar quatre ordinadors amb connexió a Internet on instal·lar l'aplicació client per ser testada i sobretot per a coordinar un moment on fer les proves i que sigui convenient per a tots els participants.

En ordre de preferència el següent nivell consistia en efectuar les proves en ordinadors connectats a una LAN. De no disposar una xarxa pròpia –com és el meu cas- el problema es convertia en trobar una xarxa que tingués una certa permissivitat amb la instal·lació de programari i amb el tràfic a determinats ports no autoritzats. Així, elements con '*firewalls*' en les estacions o a nivell de xarxa en cas que algun dels clients o el servidor estiguessin fora del segment, podien impedir el tràfic de missatges propis de l'aplicació i conseqüentment el seu correcte funcionament.

Per aquest motiu, tot i haver pensat en la possibilitat de fer les proves a la xarxa de la feina vaig haver de descartar-la, al requerir de la intervenció d'un Administrador que autoritzés la instal·lació del programari i la seva execució, permetent el tràfic en els ports dels que aquest programari fa ús.

Al no ser possibles cap dels dos nivells anteriors, l'opció que quedava era la de simular-los, ja que si era possible assegurar el funcionament en aquest entorn simulat podríem fer l'extensió a l'entorn real, o almenys fer que les proves necessàries en aquest fossin menys nombroses i es fessin únicament quan hi tinguéssim una certa confiança en que l'aplicació funcionava segons el previst.

La simulació d'un entorn de xarxa passa per l'ús d'adreces *'loopback'* del tipus 127.0.0.*, que són interpretades com adreces de xarxa però tot el tràfic és dirigit a la pròpia estació.

Per raons de comoditat, la majoria de les proves s'han efectuat sota l'entorn de desenvolupament '*Eclipse*', que permetia executar el codi en mode *debug* i accedir de forma immediata a la seva modificació en cas de detectar qualsevol error.

Les proves finals s'han efectuat un cop l'aplicació era estable i amb la mateixa estratègia de simulació de xarxa, compilant el codi font amb el compilador de Java i executant les aplicacions des de la línia de comandes, amb tres, quatre , cinc clients, funcionant correctament llevat del lògic alentiment observat al augmentar el nombre de clients.

Conclusions

En el present TFC s'ha plantejat l'aplicació de tècniques de computació multipart segura al cas particular d'un joc de Bingo, amb la finalitat que els números de cada jugada puguin ser generats de forma segura pels propis jugadors i sense la intervenció d'un tercer, a partir del càlcul de la suma de les entrades individuals generades pels 'n' participants.

A tal efecte s'ha construït un protocol criptogràfic que permet garantir la independència i el secret de les entrades individuals.

El secret de les entrades s'ha aconseguit mitjançant l'aplicació de l'esquema de compartició de secrets proposat per Shamir, en que cada jugador genera un polinomi del que el valor secret és el terme independent i comparteix únicament avaluacions d'aquest polinomi per a cada jugador. Qualsevol participant que volgués accedir al valor secret necessitaria n-avaluacions del polinomi generat i com molt podrà disposar de les n-1 avaluacions que ha enviat el propietari.

La independència de les entrades s'ha assolit mitjançant la utilització de la tècnica criptogràfica del 'compromís de bit'. Cada jugador ha generat les avaluacions inicials, xifrades amb una 'clau de sessió' simètrica que només és alliberada un cop rebudes totes les avaluacions. Qualsevol jugador deshonest que volgués crear una entrada en funció de les aportades per la resta, hauria primer de trencar el xifratge dels jugadors honestos i en un temps molt limitat.

S'ha implementat aquest protocol en una aplicació Java que simula un joc de Bingo en el que la suma calculada serveix com a llavor per a l'extracció del nombre de la jugada.

Aquest esquema pot ser aplicable a tot tipus de jocs multipart en els que una jugada pugi ser determinada per un nombre. Com exemple podria aplicar-se a una partida de cartes a un joc com el 'Set i Mig' o el 'Black Jack'.

Tot i que funcional el producte implementat és clarament susceptible de millorar en futures versions sobretot en el tractament dels certificats digitals, comprovant que estan signats per una autoritat de confiança per al receptor i no han estat revocats.

Bibliografia

Per a Criptografia, en general:

- Fuster, A., de la Guia, D., Hernández, L., Montoya, F. y Muñoz, J. (2000), *Técnicas Criptográficas de protección de datos*, 2ª edición, Ra-Ma (Madrid)
- Domingo, J., Herrera, J., Rifà, H. (2006) *Mòduls didàctics Criptografia,* UOC (Barcelona)

Per a computació multipart, alguns articles trobats a Internet:

- Cramer, R., Damgard, I (2004), *Multiparty Computation, an Introduction*, lecture notes
- Goldreich, O. (1998), *Secure Multi-Party Computation*, working draftl
- Sweeney and Shamos, A.(2004) , *A Multiparty Computation for Randomly Ordering Players and Making Random Selections*, Carnegie Mellon University, School of Computer Science, Technical Report

Pel llenguatge de programació Java:

- Froufe Quintas, Agustin (2005), *JAVA™ 2 manual de usuario y tutorial,* 4ª Edición, Ra-Ma (Madrid)

Per a *sockets* i comunicació de processos:

- Peig, E., Perramon, X. *Programació de sockets – xarxes- mòduls didàctics*, UOC (Barcelona)

Per a disseny de programari:

- Campderrich, B. (2004) *Enginyeria del programari- mòduls didàctics*, UOC (2004)

Recursos de programari usats:

- Llenguatge de programació: Java 2 Platform Standard Edition Development Kit 5.0. Disponible a http://java.sun.com/
- Entorn de desenvolupament: Eclipse 3.0.2 La darrera versió és la 3.1.2 i pot descarregar-se gratuitament des de http://www.eclipse.org
- Creació de diagrames de disseny: Jude take versió 1.2.7. Pot obtenir-se la sense càrrec i previ registre la versió actualitzada Jude Community 3.0 a http://jude.change-vision.com/jude-web/index.html
- Paquet ofimàtic: Microsoft Office 2002
- Navegador web: Internet Explorer versió 6.0
- Generació de claus RSA i certificats X509: OpenSSL-v0.9.8a. Pot obtenir-se aquest programari a http://www.openssl.org/

Annex A : Recerca de llibreries criptogràfiques

En la fase de recerca de llibreries criptogràfiques per a l'entorn Java que implementessin JCE (Java Cryptography Extension) es van localitzar les següents opcions:

- Cryptyx (www.cryptix.org) . Es tractava d'una llibreria 'open-source' , amb la darrera versió alliberada en Març de 2005. Tot i que estava àmpliament referenciada en moltes pàgines web, semblava ser que s'havia aturat en el seu desenvolupament doncs aportava un proveïdor per a versions 1.4 i anteriors del JDK.
- Bouncy Castle (www.bouncycastle.org) . Un altre llibreria 'open-source' amb moltes referències a Internet, però que a diferència de Criptyx mantenia el seu desenvolupament actiu. En el moment en que aquest TFC va ser desenvolupat, la darrera versió del proveïdor JCE era aplicable a la versió 1.5 del JDK.
- IAIK (www.iaik.at) . Aquesta era una llibreria comercial, però que per usos educacionals o d'investigació proporcionava una versió d'avaluació gratuïta. Plenament actualitzada, donava suport a les versions mes recents del JDK.

Descartant la llibreria Cryptix, la comparació entre les dos restants va donar els següents resultats:

- ambdós implementaven les funcionalitats bàsiques que previsiblement requeriria el TFC: xifratge simètric i asimètric, signatures digitals, funcions hash.
- totes dues incorporaven codi d'exemple i documentació en format JavaDoc, tot i que la aportada per IAIK semblava una mica més detallada.
- IAIK aportava classes per a la generació de nombres aleatoris, mentre que 'Bouncy Castle' no disposava d'aquesta funcionalitat.
- IAIK incorporava unes classes d'utilitats que podien facilitar la tasca en diversos aspectes com: comparació i manipulació de bytes, realitzar operacions XOR, codificació/descodificació en Base 64, llegir fitxers i emmagatzemar-los en bytes, ...

Les funcionalitats addicionals aportades per IAIK i la seva aparença més senzilla i estructurada, van inclinar la balança a favor del seu us en el desenvolupament d'aquest TFC.

Annex B: Algorisme per a generar un cartró de bingo

```
var
        constant numFila=3;
        constant numColumnes =9;
        cartro[numFilas,numColumnes] enter;
        numValores[numColumnes] enter;
        seleccionats[90], i, j enter, valor, enFila;
        valorsForçats=0 enter
        final booleà
fvar

inicialitzar a zero matrius cartro[numFilas,numColumnes], numValores[numColumnes]
        i seleccionats[90]

per f=1 fins f = numFilas
        // tractament especific de la fila 3 del cartro
        si f = numFilas
                per i=1 fins i=9
                        // si d'una columna no hi ha cap valor
                        // en la fila 3 haurà de tenir un
                        si numValores[i]=0 aleshores
                                cartro[f,i]= valor enter aleatori
                                        corresponent a la desena escollida
                                //si el valor seleccionat ja
                                //s'ha posat al cartró, aleshores
                                //hem de triar altre
                                mentre seleccionats[cartro[f,i]]<>0
                                        cartro[f,i]= valor enter aleatori
                                        corresponent a la desena escollida
                                fmentre
                                // marquem el valor seleccionat
                                seleccionats[cartro[f,c]]=1
                                // augmentem el comptador de valors introduits
                                // de forma forçada
                                valorsForçats= valorsForçats+1
                                numValores[i]=numValores[i]+1
                        fsi
                fper
        fsi

        // tria aleatòriament les cinc columnes d'una fil.la, excepte
        // en la fila 3, en la que pot ser ja s'han introduït valors
        // segons el contingut de valorsForçats
        per i= (0+ valorsForçats) fins 5
                // tria una columna aleatòria
                c = valor enter aleatori en interval (1,numColumnes)
                // si la columna ja te valor en aquesta fila o si en
                // el conjunt ja te dos valors –màxim per a una columna-
                // aleshores tria un altre columna aleatòria
                mentre (cartro[f, numColumnes]<>0 o numValores[numColumnes]=2)
                        c = valor enter aleatori en interval (1,10)
```

```
                fmentre
                // un cop escollida la columna, tria un valor aleatori
                // de la seva desena
                cartro[f,i]= valor enter aleatori corresponent a
                                la desena escollida
                // si aquest valor ha estat escollit en altre fil.la
                //aleshores tria altre
                mentre seleccionats[cartro[f,i]]<>0
                        cartro[f,i]= valor enter aleatori
                                corresponent a la desena escollida
                fmentre
                // actualitza comptador de valors introduits en columna
                numValors[i] = numValors[i]+1
                // marca valor escollit
                seleccionats[cartro[f,i]]=1
        fper
fper

//recorregut de la matriu bidimensional cartro[numFilas,numColumnes]
//ordenant de mes petit a mes gran aquells valors de les columnes que
//tinguin dos elements i que el valor mes alt estigui en una fila
//superior que el valor mes baix

per i=1 fins numColumnes
        valor = -1
        enFila=0
        j=1
        final = fals
        mentre j <= numFiles i no final
                // si la posició te un número
                si cartro[j,i]<>0
                        // si és el primer, el guarda per a comparar
                        si valor = -1
                                valor= cartro[j,i]
                                enFila = i
                        // si ja ha guardat un, fa la comparació
                        sino
                                // si el valor és més gran, fa intercanvi
                                si valor > cartro[j,i]
                                        cartro[enFila,i]=cartro[j,i]
                                        cartro[j,i]= valor
                                fsi
                                // si ha fet la comparacio de dos valors
                                // per la definició del cartró
                                // no pot haver un altre valor a la mateixa
                                // columna
                                final = cert
                        fsi
                fsi
                // anem a la fila següent
                j=j+1
        fmentre
fper
```

Glossari

CA Certificate Authority o autoritat de certificació. Ent que certifica la correspondència entre una identitat i una clau pública, mitjançant la seva signatura digital.

Certificat digital. Document digital mitjançant el que una tercera part (CA o *Certificate Authority*) garanteix la vinculació entre la identitat d'un subjecte i la seva clau pública, i ho fa signant aquest document amb la pròpia clau privada de la CA. Aquesta certificació només te sentit si el receptor del certificat diposita la seva confiança en la CA signant. Un dels formats més estesos és el X509.

Certificat X509. Estàndard de certificat digital creat per la Unió Internacional de Telecomunicacions.

Clau de sessió. Clau simètrica creada per a la seva utilització en un únic procés de xifrat – desxifrat. Es fa servir en la tècnica de compromís de bit per a xifrar els compromisos i també per exemple en la tècnica del 'sobre digital', on el contingut es xifra amb la clau de sessió i aquesta es xifra amb la clau pública del destinatari, de forma que només aquest pugui finalment desxifrar-lo.

Compromís de bit. Tècnica criptogràfica que permet el repartiment mutu de compromisos (valors) entre un conjunt de parts i obliga a que cap de les parts pugui obrir el compromís per voluntat pròpia abans de que cap altre ho faci, mitjançant el xifratge d'aquests compromisos amb una 'clau de sessió' simètrica, que només s'allibera –es comunica a la resta de participants- un cop s'han rebut la totalitat dels compromisos.

Computació Multipart Segura. Àrea d'investigació en el camp de la Criptografia que estudia la possibilitat de calcular de forma conjunta amb 'n' parts funcions de 'n' variables que idealment han de mantenir-se en secret per a tot aquell participant que diferent del seu propietari fins i tot desprès d'obtenir el resultat públic de la funció.

Esquemes de compartició de secrets. Veure *esquemes de llindar (m,n)*

Esquemes de llindar (m,n) Tècnica criptogràfica mitjançant la que un secret és dividit en 'n' parts, requerint-se de la conjunció de m d'elles (m<=n) per a la seva reconstrucció.

Infrastructura de clau pública. Conjunt de maquinari, programari, persones, polítiques i procediments necessaris per a crear i gestionar certificats digitals basats en Criptografia de Clau Pública.

Signatura digital. Tècnica criptogràfica que mitjançant l'ús de Criptografia de Clau Pública possibilita assegurar la identitat de l'emissor d'un missatge (autenticitat) i al mateix temps impedir que aquest pugui negar l'autoria del missatge (no repudi). L'emissor xifra el missatge –usualment un resum de l'original- amb la seva clau privada, de forma que qualsevol receptor que disposi de la seva clau pública pot desxifrar aquesta signatura o resum i comprovar que coincideix/ha sigut generat amb el missatge