Càlcul distribuït en BOINC per el trencament de l'algorisme RC5

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
David Cervelló Batlle
i dirigit per
Jordi Herrera Joancomartí
Bellaterra, 10 de juny de 2007

Universitat Autònoma de Barcelona
Escola Tècnica Superior d'Enginyeria

Sol·licitud de lectura del Projecte Final de Carrera

Sol·licitud de lectura del Projecte Final de Carrera de la titulació Enginyeria Superior Informàtica.

David Cervelló Batlle, amb el DNI/Passaport núm. 45641627-J
nascut/da el 02/04/1984 i amb domicili a Terrassa,
carrer Santa Llúcia, núm. 165, pis i porta B,
codi postal 08222, telèfon 937362572, e-mail 213093@campus.uab.cat,

EXPOSO:

Que, havent elaborat el Projecte Final de Carrera
Càlcul distribuit en Bonic per el trencament de l'algorisme RC5,

sota la direcció del/de la profesor /a (indicar el nom del director/a)
Joan Herrera Joancomartí,

*Nombre de Projecte web: 96.

DEMANO:
La constitució d'un Tribunal per a la lectura del Projecte Final de Carrera.

UAB
Escola Tècnica Superior d'Enginyeria
Data: 30 MAIG 2007

Bellaterra (Cerdanyola del Vallès), 17/05/2007

El/la Director/a del projecte	El/la alumne /a
Nom Joan Herrera Joancomartí	Nom David Cervelló Batlle
Departament DEIC	
Signatura Director/a	Signatura Alumne/a
Data 17-5-2007	Data 17/05/2007

*Per la titulació d'Enginyeria Informàtica

Índex

Índex de figures

Índex de taules

1. Introducció

Aquest projecte de fi de carrera anomenat "Càlcul distribuït en BOINC per el trencament de l'algorisme RC5" tracta sobre criptografia i càlcul distribuït i concretament de com es pot utilitzar el càlcul distribuït per trencar un criptosistema.

Com es veurà, consisteix de participar en el concurs de l'RSA Laboratories i trencar el criptosistema proposat. Per dur a terme aquest objectiu es partirà de la suposició de Kerckhoffs per realitzar un atac de força bruta sobre el criptosistema donat.

Pel trencament del criptosistema s'utilitzarà una plataforma de càlcul distribuït anomenada Berkeley Open Infrastructure for Network Computing (BOINC) de la Universitat de Berkeley, molt coneguda gràcies al projecte SETI@home. Així doncs, s'utilitzaran les facilitats proveïdes per la BOINC per repartir milions de possibles claus entre diversos clients per a que provin si alguna de les claus rebudes és la solució que desxifra el criptograma proposat.

1.1. Objectius del projecte

Tot i que pot semblar que l'objectiu del projecte sigui guanyar el concurs, del que se'n parlarà al següent punt, aquest és massa ambiciós i sobrepassa, en molt, l'objectiu d'un projecte final de carrera. Es veurà que per trencar el criptosistema, s'haurien de provar la meitat de les claus[1], 2^{71}, que són 2.361.183.241.434.822.606.848 claus. Malgrat tot, en el curt espai de temps del que es disposa per dur a terme el trencament de la clau, serà totalment inviable.

Per tant, l'objectiu del treball és posar en funcionament la infraestructura necessària per possibilitar el trencament del criptosistema si es disposés del temps necessari. Això es tradueix en habilitar un ordinador amb el servidor de la BOINC correctament configurat i preparat per crear, enviar i validar el conjunt de claus. Com es veurà més endavant,

[1] Al punt 2.2 es veurà per què es diu de provar només la meitat de l'espai de claus.

aquest procés consisteix en desenvolupar un programa client que s'executa als ordinadors dels voluntaris i desenvolupar també les aplicacions del servidor específiques d'aquest projecte.

Un altre dels objectius d'aquest projecte és el de donar una estimació de quant de temps es necessitaria segons els recursos dels que es poguessin disposar i veure, de forma empírica, quina és la realitat dels projectes que utilitzen recursos de càlcul públics i voluntaris, en anglès coneguts com a *Public Resource Computing* (PRC).

Finalment, l'últim dels objectius és el d'aconseguir que el màxim de gent cedeixi voluntàriament els seus recursos computacionals per l'execució d'aquest projecte. És a dir, intentar donar un caire social a aquest projecte i fer que es conegui una mica.

1.2. Motivació

Les motivacions alhora de realitzar aquest projecte són bàsicament dues. En primer lloc, poder aplicar els coneixements en Seguretat Computacional, i aplicar-los. En segon lloc, comprovar la resistència dels algorismes criptogràfics, en aquest cas de l'algorisme RC5, en front a atacs per força bruta.

Per l'altra banda, una motivació afegida per dur a terme aquest projecte és, també, assolir coneixements en càlcul distribuït i aplicar-los. Primer, veure per a què pot ser útil el càlcul distribuït i seguidament aplicar-ho a la tasca d'intentar trencar un criptosistema.

Combinant les dues motivacions, doncs, sorgeix la idea de realitzar un atac per força bruta utilitzant càlcul distribuït. Per fer-ho en base a un desafiament real, es va triar el proposat per RSA Laboratories, que es descriu més endavant.

1.2.1. Atac per força bruta mitjançant càlcul distribuït

Sense entrar en detalls sobre què és el càlcul distribuït ni que són els sistemes distribuïts, doncs aquests es veuran a l'apartat 3, ni tampoc sobre els atacs per força bruta, tractats en l'apartat 2, cal fer una breu introducció sobre què aporta aquesta combinació d'ambdues tècniques.

Quan es vol comprovar la resistència d'un algorisme de xifrat, es poden utilitzar diverses tècniques de criptoanàlisi. Depenent de l'algorisme, es poden estudiar diverses formes per intentar trencar un criptosistema que l'utilitzi. Per exemple, en DES es pot realitzar criptoanàlisi diferencial, lineal o atacs per força bruta. En el cas de l'RC5, fins a dia d'avui no s'ha trobat cap manera (coneguda) de realitzar un atac, que no sigui mitjançant la força bruta.

És en aquest punt que entra el càlcul distribuït. Provar un espai de claus tan gran com 2^{72} és actualment inviable si no es disposa de supercomputadors, o de grans recursos computacionals en general. Per tant, el càlcul distribuït, que com es veurà és el mètode en que diverses parts d'un programa s'executen simultàniament en dos o més ordinadors que es comuniquen per la xarxa, és totalment necessari per acomplir el propòsit de trencar aquest criptosistema.

En aquest cas, els recursos necessaris els proporcionarà la *Public Resource Computing* (PRC). És a dir, els recursos els proporcionaran voluntaris que cedeixin les seves màquines quan aquestes estan en estat *idle*. Per exemple, quan no estiguin treballant amb el seu ordinador i per tant la CPU estigui en estat ociós, sense utilitzar-se la seva potència de càlcul.

Com a sistema distribuït s'ha triat la Berkeley Open Infrastructure for Network Computing (BOINC), àmpliament utilitzada per molts projecte que aprofiten la idea de la PRC per dur a terme les seves tasques. En aquest cas, les tasques a dur a terme pels voluntaris seran les de provar un rang de claus determinat, dintre de l'espai de claus total.

1.2.2. El desafiament d'RSA Laboratories

RSA Laboratories[2], pertanyents a l'empresa EMC[3], va publicar l'any 1997 una sèrie de reptes criptogràfics consistents en les competicions o desafiaments que s'expliquen a continuació.

Actualment, RSA Laboratories manté dos desafiaments oberts, que es descomponen en diverses classes:

- Clau pública: factorització RSA
- Clau privada: trencament de DES i RC5

El primer dels desafiaments proposa factoritzar un producte de dos primers grans, que pot ser utilitzat en l'algorisme de xifrat RSA com a clau pública[4]. Segons els autors del desafiament aquests nombres han estat generats de manera que els seus factors no pugin ser obtinguts per cap altra manera que factoritzar el nombre.

El segon dels desafiaments, el de clau privada, proposa trobar la clau secreta d'un criptosistema. Els algorismes utilitzats són el DES i l'RC5, aquest últim en diverses modalitats. Amb DES existeix una única competició, ja que les claus d'aquest algorisme de xifrat només poden ser de 56 bits. A més a més, aquesta competició ja està finalitzada. En RC5 existeixen dotze competicions, ja que com es veurà al punt 2.1.2, aquest algorisme admet diferents parametritzacions. L'objectiu d'RSA Laboratories és el de quantificar el nivell de seguretat en front a qualsevol tipus d'atacs. Com es veurà a l'apartat 2.2, actualment la única manera existent consisteix en un atac per força bruta.

Les diverses competicions existents per al trencament de l'RC5 són les següents[5]:

Identificador	Longitud clau	Estat	Temps emprat
RC5-32/12/5	40 bits	Acabat	3.5 hores
RC5-32/12/6	48 bits	Acabat	313 hores
RC5-32/12/7	56 bits	Acabat	265 dies
RC5-32/12/8	64 bits	Acabat	1757 dies[6]
RC5-32/12/9	72 bits	En procés	-

[2] Veure la pàgina web http://www.rsa.com/rsalabs/
[3] Veure la pàgina web http://www.emc.com/
[4] La part privada de la clau va ser descartada.
[5] Veure la pàgina web http://www.rsa.com/rsalabs/node.asp?id=2106
[6] De temps actiu segons va informar el guanyador.

RC5-32/12/10	80 bits	En procés	-
RC5-32/12/11	88 bits	En procés	-
RC5-32/12/12	96 bits	En procés	-
RC5-32/12/13	104 bits	En procés	-
RC5-32/12/14	112 bits	En procés	-
RC5-32/12/15	120 bits	En procés	-
RC5-32/12/16	128 bits	En procés	-

Taula 1: Competicions d'RSA Laboratories pel trencament de l'RC5

Com es pot veure a la Taula 1, les quatre primeres modalitats ja estan concloses, i al punt 3.4 es veurà que en les dues últimes el criptosistema ha estat trencat per l'equip de *distribuited.net*.

Les modalitats d'RC5 només difereixen en el nombre de bits de la clau[7], paràmetre que com es veurà afecta al nivell de seguretat del criptosistema. Els altres dos paràmetres es mantenen constants. Per cadascun dels reptes se'ns dóna la següent informació:

- L'identificador del repte, que alhora informa de l'algorisme i els seus paràmetres.
- La data de publicació del repte.
- L'estat.
- El vector d'inicialització (IV) a utilitzar pel mode de xifrat Cipher Block Chaining.
- El text xifrat en hexadecimal.

A més a més, també s'indica que per poder comprovar que s'ha trobat la clau de xifrat correcta, el text en clar comença amb la següent frase: "The unkown message is: ". El mètode utilitzat pel farciment de bytes, en cas que el missatge no sigui un múltiple de dues vegades la mida de paraula, és omplir amb bytes amb el valor hexadecimal resultant de calcular el nombre de bits de farciment. Si es dóna el cas que no cal fer el farciment, llavors s'afegiran 8 bytes amb el valor hexadecimal *0x08*. El següent exemple il·lustra com es faria el farciment de bits:

[7] A l'apartat 2.1.2 es veurà aquest algorisme i la seva parametrització.

Frase a xifrar	*Bit padding demonstration message.*
Frase resultant	*The unkown message is: Bit padding demonstration message*
Frase resultant en hexadecimal	*54 68 65 20 75 6E 6B 6F 77 6E 20 6D 65 73 73 61 67 65 20 69 73 3A 20 42 69 74 20 70 61 64 64 69 6E 67 20 64 65 6D 6F 6E 73 74 72 61 74 69 6F 6E 54 68 65 20 75 6E 6B 6F 77 6E 20 6D 65 73 73 61 67 65 20 69 73 3A 20 42 69 74 20 70 61 64 64 69 6E 67 20 64 65 6D 6F 6E 73 74 72 61 74 69 6F 6E 20 6D 65 73 73 61 67 65 0D 0A 08 08 08 08 08 08*

Taula 2: Farciment de bits pels desafiaments d'RSA Laboratories

Per a que les persones que vulguin participar en algun dels desafiaments puguin provar la seva aplicació, RSA Laboratories dóna uns pseudo desafiaments de prova on, a més a més de la informació anterior, també es dóna el text original. Com es veurà a l'apartat 5.1, s'ha utilitzat el pseudo test corresponent per fer unes aplicacions de prova que verifiquessin el correcte funcionament de la implementació triada.

1.3. Estructura de la memòria

Aquesta memòria s'estructura en cinc apartats principals, més aquesta introducció, les conclusions, la bibliografia i els annexes. Els cinc apartats principals són els següents:

- Conceptes de criptografia
- Càlcul distribuït per a la realització d'atacs
- El sistema distribuït BOINC
- Desenvolupament de la infraestructura pel trencament de l'RC5-32/12/9
- Funcionament del projecte

En el primer d'aquests apartats s'expliquen els conceptes bàsics sobre criptografia, centrant-se en la criptografia de clau privada. Alguns conceptes són el text en clar i xifrat, els processos de xifrar i desxifrar i com es relacionen, els protocols criptogràfics, etc. En el context de la criptografia de clau privada, s'expliquen els modes de xifrat i ja més en detall, s'explica l'algorisme de xifrat RC5, utilitzat en aquest projecte. Finalment també es veu la suposició de Kerckhoffs, que és el pilar fonamental de l'atac per força bruta a realitzar.

En el segon dels apartats citats, es dóna una introducció al càlcul distribuït, definint en què es basa i els seus avantatges i desavantatges. També es veuen les diferents arquitectures dels sistemes distribuïts i què és i què suposa utilitzar recursos públics per al càlcul distribuït. Es dóna una explicació més extensa i detallada del càlcul distribuït per a la realització d'atacs per força bruta i s'introdueixen les principals eines de càlcul distribuït existents. Finalment es comenten també els diferents projectes de càlcul distribuït que es poden trobar a Internet, des de dues perspectives diferents, que són projectes que utilitzen la BOINC i els projectes sobre criptografia. Es posa èmfasis en els segons, sobretot en el projecte de *distribuited.net*.

Com a últim capítol teòric es veu la infraestructura utilitzada, anomenada BOINC. Es descriu en què es basa i quin tipus d'aplicacions s'adapten millor a la idea de BOINC. Com a segon apartat es veu l'estructura del model seguit per BOINC, anomenat client-servidor, i els components d'aquesta estructura. Es comenten també els dos conceptes principals de BOINC, les unitats de treball i els resultats i un concepte molt important com la redundància en el càlcul. Un cop vistos aquests conceptes s'expliquen en detall el model d'emmagatzemament de la BOINC i la utilitat de cadascun dels dimonis que forma part del servidor. També es parla sobre la seguretat dels clients i com s'evita l'execució de codi que no pertany al projecte. Com a últim apunt sobre BOINC, es veu el cicle de vida de tota unitat de treball, per poder entendre millor el funcionament d'aquesta infraestructura.

En el primer dels apartats pràctics es comenten les dues etapes del desenvolupament. Per cada etapa s'expliquen quines aplicacions s'han desenvolupat i la seva utilitat. Es veu també la posada en funcionament de la infraestructura, és a dir, el procés que s'ha seguit per a la instal·lació i creació del projecte. Sobre la posada en funcionament, també es dóna una guia de consulta ràpida però detallada sobre el procés de posada en funcionament i les comandes utilitzades als annexes.

En el segon i últim apartat pràctic es veu el funcionament del projecte des del punt de vista del cicle de vida de les unitats de treball introduït a l'apartat 4.7. També s'explica el procés de compilació de les aplicacions mitjançant les eines que les APIs de BOINC proveeixen. Finalment es veu el conjunt de proves realitzat sobre la infraestructura, que

es pot trobar més en detall als annexes, juntament amb l'evolució del trencament del criptosistema.

2. Conceptes de criptografia

La criptografia es pot definir com la ciència de xifrar i desxifrar informació utilitzant tècniques matemàtiques de manera que es possibiliti l'intercanvi d'informació entre dues o diverses parts per tal que aquesta pugui ser sols llegida per aquestes parts [1]. La finalitat d'aquesta ciència és tant la de garantir el secret en la comunicació entre dos o més entitats com la de que la informació enviada sigui autèntica. Aquest segon punt inclou tant que la informació no hagi estat modificada com que l'emissor sigui qui assegura ser. Aquestes propietats són les de privadesa, integritat i autenticitat, respectivament.

Conceptes bàsics

Parlant en termes criptogràfics, la informació original que es vol xifrar s'anomena text en clar, ja sigui text intel·ligible pels ésser humans com algun altre tipus d'informació (executables o qualsevol tipus de dades). El procés de convertir el text en clar en quelcom inintel·ligible per l'ésser humà s'anomena xifrar. Aquest procés consta d'un algorisme de xifrat basat en l'ús d'una clau, anomenada clau de xifrat, que es manté oculta al públic. L'aplicació d'aquest algorisme amb una clau concreta transforma el text en clar, l'entrada de l'algorisme, en un text xifrat o també anomenat criptograma.

El procés invers s'anomena desxifrat, en el que es desfà el procés anterior, recuperant, a partir del criptograma i la clau corresponent, el text en clar original. En el cas que s'utilitzés una clau diferent[8], no es podria recuperar el text original.

En termes matemàtics s'expressarien les anteriors definicions de la següent manera: si s'anomena M el conjunt de textos en clar, *C* al conjunt de criptogrames corresponents a xifrar els textos en clar i *K* al conjunt possible de claus, llavors el conjunt d'algorismes de xifrat i desxifrat, *E* i *D* respectivament, s'expressen com:

$$E = \{ E_k \mid M \rightarrow C, \ \forall k \in K \}$$
$$D = \{ D_k \mid C \rightarrow M, \ \forall k \in K \}$$

[8] Diferent o sense cap relació matemàtica no trivial.

Els algorismes de xifrat i desxifrat, com s'ha dit, han de complir la propietat que l'un sigui l'invers de l'altre, és a dir:

$$D_k(E_k(m)) = m, \ \forall k \in K, m \in M$$

S'entén per protocol criptogràfic com un protocol abstracte o concret que realitza funcions relacionades amb la seguretat i aplica mètodes criptogràfics [2]. Per criptosistema s'entén el conjunt d'algorismes criptogràfics juntament amb els processos de gestió de claus que suporta l'ús dels algorismes en alguns contextos d'aplicació [3].

Els algorismes de xifrat es divideixen en dues grans famílies, anomenades de clau simètrica i de clau asimètrica. Aquestes dues grans famílies també es coneixen pels noms de criptografia de clau pública i de clau privada, nom que s'utilitzarà d'ara en endavant.

La gran diferència entre aquestes dues famílies rau en si les claus per xifrar i desxifrar són la mateixa o bé si són diferents. En criptografia de clau pública es tenen un parell de claus, una per xifrar i l'altra per desxifrar. Aquestes dues claus, com és lògic, guarden una relació matemàtica entre elles. Una variant de la criptografia de clau pública és la de corbes el·líptiques, basada en les matemàtiques de les corbes el·líptiques.

Donat que el cas que ocupa aquest projecte utilitza un algorisme de clau privada, es deixa de banda la criptografia de clau pública per centrar-se en la de clau privada. En aquesta, s'utilitza la mateixa clau tant pel xifrat com pel desxifrat.

2.1. La criptografia de clau privada

En la comunicació mitjançant aquest tipus de criptografia, les dues o diverses parts han de posar-se d'acord en la clau a usar pel xifrat de les dades. Aquest procés es realitza usant algun protocol criptogràfic, que ha de garantir que només les parts en la comunicació tindran accés a aquesta clau. En la Figura 1 es pot veure com es realitzaria la transmissió d'informació xifrada mitjançant un esquema de clau privada:

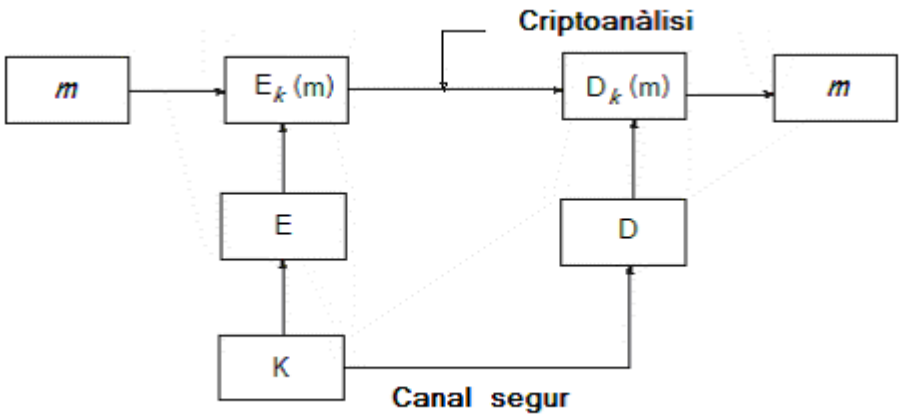

Figura 1: Transmissió d'informació mitjançant un esquema de clau privada

La transmissió de la clau es sol realitzar mitjançant criptografia de clau pública, xifrant la clau privada del criptosistema per enviar-la.

Els algorismes de clau privada es poden separar en dos grans conjunts, anomenats xifradors de flux i xifradors de bloc. En els xifradors orientats a flux les unitats[9] es xifren d'una en una, com un flux, d'aquí el seu nom. Cada unitat es xifrarà segons l'estat actual del sistema. Només esmentar que els xifradors orientats a flux es divideixen en dues categories: els síncrons i els auto-sincronitzables. No s'entrarà en més detall ja que l'algorisme de xifrat que ocupa aquest projecte és, precisament, orientat a bloc.

Els algorismes de clau privada orientats a bloc treballen sobre grups de bits de longitud determinada, anomenats, com el nom d'aquest tipus de xifradors indica, blocs. Si es vol xifrar un text en clar superior a la longitud d'aquest bloc, llavors entren els modes de xifrat.

2.1.1. Modes de xifrat

Com es veurà a l'apartat 2.1.2, l'RC5 és un algorisme de xifrat en bloc, ja que agafa grups de dues paraules per xifrar. Aquest mode de xifrat té un problema, i és que amb el

[9] Usualment bits o bytes.

mateix text d'entrada i la mateixa clau sempre es produeix la mateixa sortida. Per tant, criptogràficament aquesta opció no és la més segura.

En aquest punt entren els modes de xifrat. Aquests modes s'han desenvolupat per proveir de confidencialitat als xifradors de bloc. Els modes més senzills només proveeixen de confidencialitat, mentre que altres modes més elaborats també aporten integritat del missatge. Entre els primers modes de xifrat hi ha l'Electronic Code Book (ECB), el Cipher Block Chaining (CBC), l'Output FeedBack (OFB) i el Cipher FeedBack (CFB). D'altres més avançats són el Counter with CBC-MAC (CCM), el EAX, el Galois/Counter Model (GCM) i l'Offset Codebook Mode (OCB) [4].

Excepte l'ECB, els altres modes solen utilitzar un vector d'inicialització anomenat IV (Inicialization Vector). Mitjançant aquest vector s'incorpora un factor d'aleatorietat a l'algorisme de xifrat sempre i quan amb una mateixa clau no s'utilitzi el mateix IV.

La longitud d'aquest vector sol ser la mateixa que la del mida de bloc o bé que la de la mida de la clau. Donat que s'utilitza per xifrar la informació, també es necessitarà alhora de desxifrar-la. La manera de transmetre aquest vector a la persona que necessiti desxifrar la informació dependrà del protocol utilitzat, com per exemple agregant-lo al paquet que es vol transmetre o intercanviar-lo junt amb la clau de xifrat.

Donat que l'objectiu d'aquest PFC és posar en funcionament la infraestructura per trencar l'RC5 en mode CBC, només es veuran els tres primers modes de xifrat (ECB, CBC, CFB i OFB), i posant èmfasi en el CBC.

El mode ECB

Electronic Code Book (ECB) és el mode de xifrat en bloc més simple que existeix. En l'ECB es divideix el text en clar en blocs de longitud fixa i es xifren cadascun per separat. A la Figura 2 es pot veure el seu funcionament (xifrat i desxifrat respectivament):

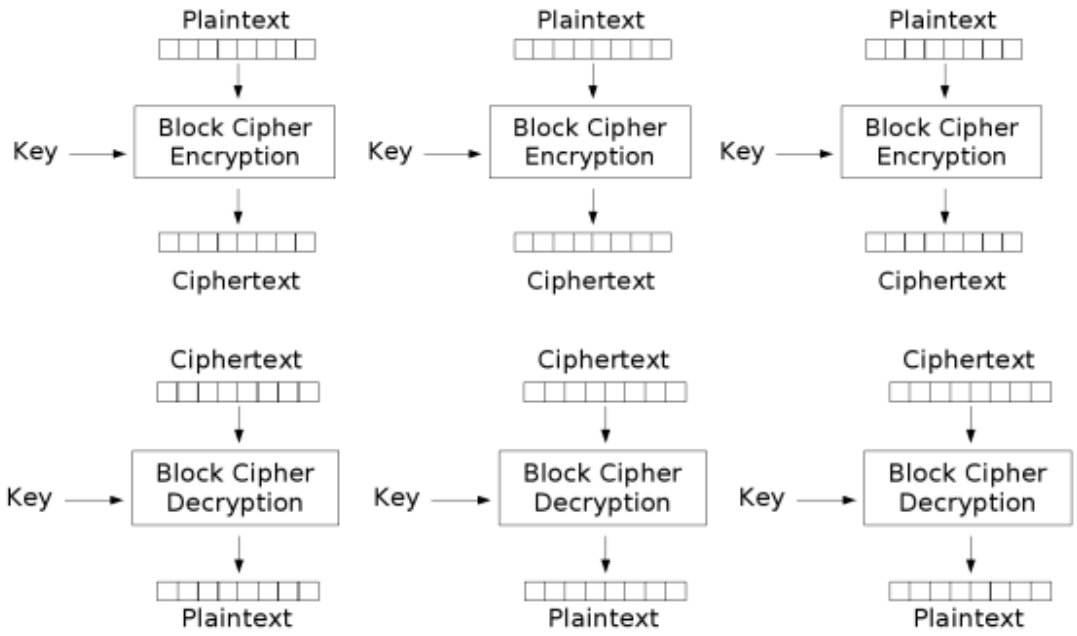

Figura 2: Esquema de xifrat ECB

Les propietats de l'ECB són que si es canvia d'ordre dels blocs de text xifrat no afecta al desxifrat, donada la independència dels blocs. Per tant es pot xifrar i desxifrar paral·lelament sense problemes. ECB no té propagació d'errors, ja que al ser el xifrat de cada bloc independent dels restants, un error de transmissió d'un bloc xifrat només afecta al propi bloc.

El mode CBC

En Cipher-Block Chaining (CBC) al primer bloc de text en clar, abans de xifrar-lo, se li aplica l'operació XOR amb el vector d'inicialització. La sortida del xifrador de bloc serveix com a IV pel proper bloc a xifrar i així successivament. Usant aquest mode, només cal canviar l'IV inicial per aconseguir que un mateix text en clar sigui xifrat de manera diferent, ja que cada bloc de text xifrat depèn de tots els blocs anteriors.

La Figura 3 mostra l'esquema d'aquest mode:

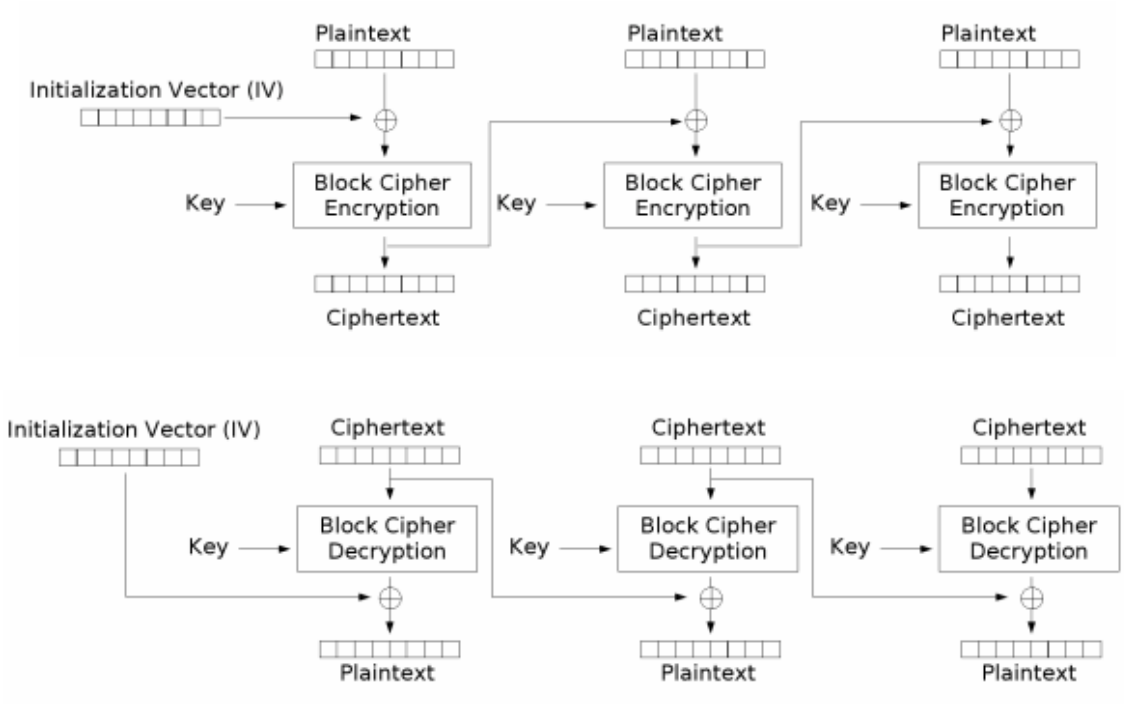

Figura 3: Esquema de xifrat CBC

Matemàticament, el xifrat i desxifrat en CBC s'expressa respectivament com[10]:

$$C_i = E_k(P_i \oplus C_{i-1}),\ C_0 = IV$$

$$P_i = D_k(C_i) \oplus C_{i-1},\ C_o = IV$$

CBC té l'inconvenient que no permet xifrar paral·lelament, sinó que la seva execució ha de ser seqüencial, tot i que el desxifrat es pugui paral·lelitzar. Un altre inconvenient és que donat que es xifren blocs, si les dades no són múltiples de la mida de bloc, aquestes s'han de farcir amb zeros, per exemple. Al contrari que en l'ECB, si es canvia l'ordre dels blocs xifrats, això afecta a l'hora de desxifrar, ja que es necessita tenir els blocs ordenats. La propagació d'errors en CBC només afecta al propi bloc on s'ha produït l'error i al immediatament següent, però mai afecta als restants. Per altra banda, si en la comunicació dels blocs xifrats es perd algun bit, i no es detecta per poder demanar la retransmissió del bloc, el sistema no es pot recuperar.

[10] Es considera que el primer bloc té índex 1.

A més a més, si es vol dotar d'autenticació a aquest mode, abans de xifrar el següent bloc, es realitza l'operació XOR entre les dades xifrades i les originals. Igualment, en el procés de desxifrat també es realitza l'XOR.

El mode CFB

En Cipher FeedBack (CFB) es converteix el xifrador de bloc en un xifrador de flux auto-sincronitzat.

El mode de funcionament es mostra en la Figura 4:

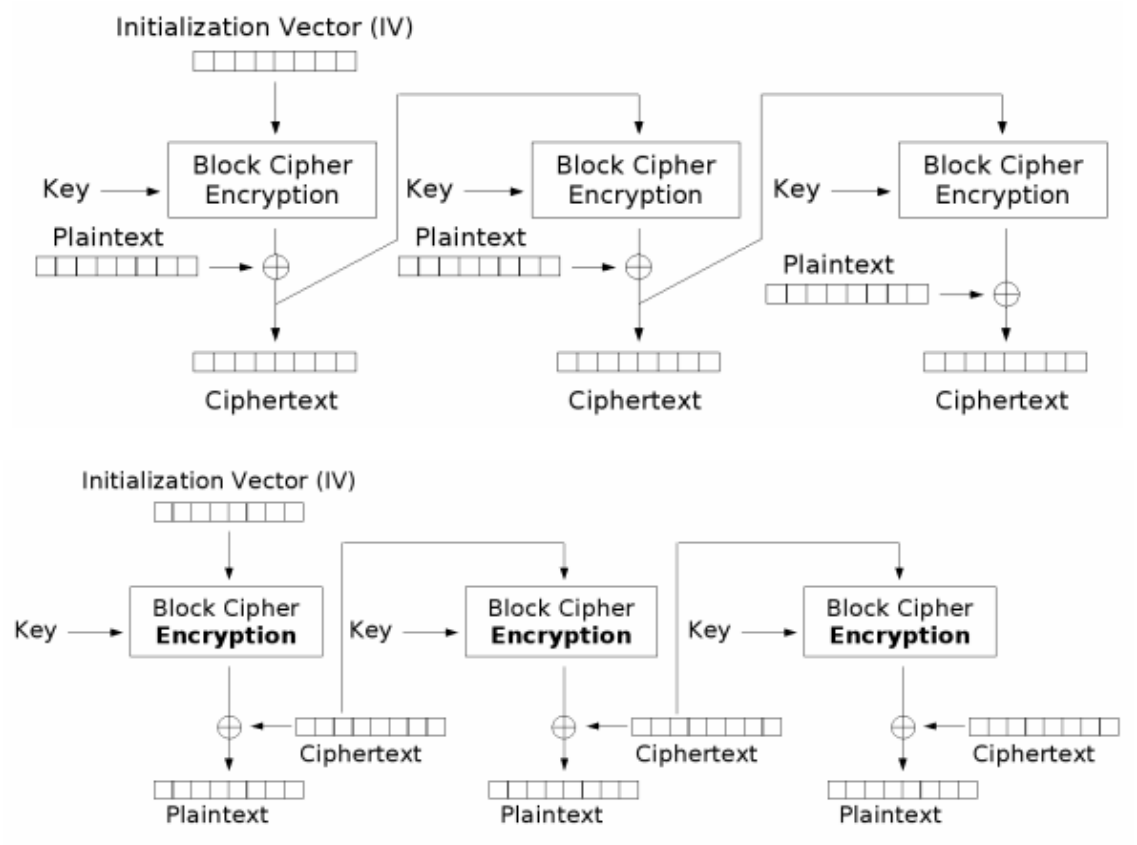

Figura 4: Esquema de xifrat CFB

Matemàticament s'expressa com:

$$C_i = E_k(C_{i-1}) \oplus P_i,\ C_0 = IV$$

$$P_i = E_k(C_{i-1}) \oplus C_i,\ C_0 = IV$$

Donat que en CFB es transforma el xifrador de bloc en xifrador de flux, es pot escollir la mida de les dades a xifrar. Si n és la mida del bloc, es considera r ($1 \leq r \leq n$) la mida (variable) de les dades que es xifrarà cada vegada. Per tant l'XOR que s'aplica a r bits de l'IV xifrat amb r bits del text en clar. A la Figura 5 es pot apreciar:

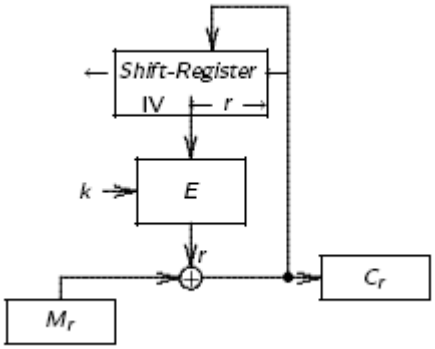

Figura 5: Esquema de xifrat de flux en CFB

A l'igual que el mode CBC, el procés de xifrat no es pot paral·lelitzar mentre que el de desxifrat sí. També els errors només es propaguen entre uns pocs blocs, concretament $\lceil n/r \rceil$. Però si $r = 1$ llavors l'error només es propaga al bloc següent.

El mode OFB

Output FeedBack converteix un xifrador en bloc en un xifrador de flux síncron. Es xifra inicialment el vector IV i a aquest resultat se li aplica l'operació XOR amb el text en clar per produir el text xifrat. El resultat de xifrar l'IV sense aplicar-li l'XOR es torna a xifrar per tornar-li a aplicar l'XOR amb el següent tros de text en clar i així successivament. L'esquema de xifrat i desxifrat es veu a la Figura 6:

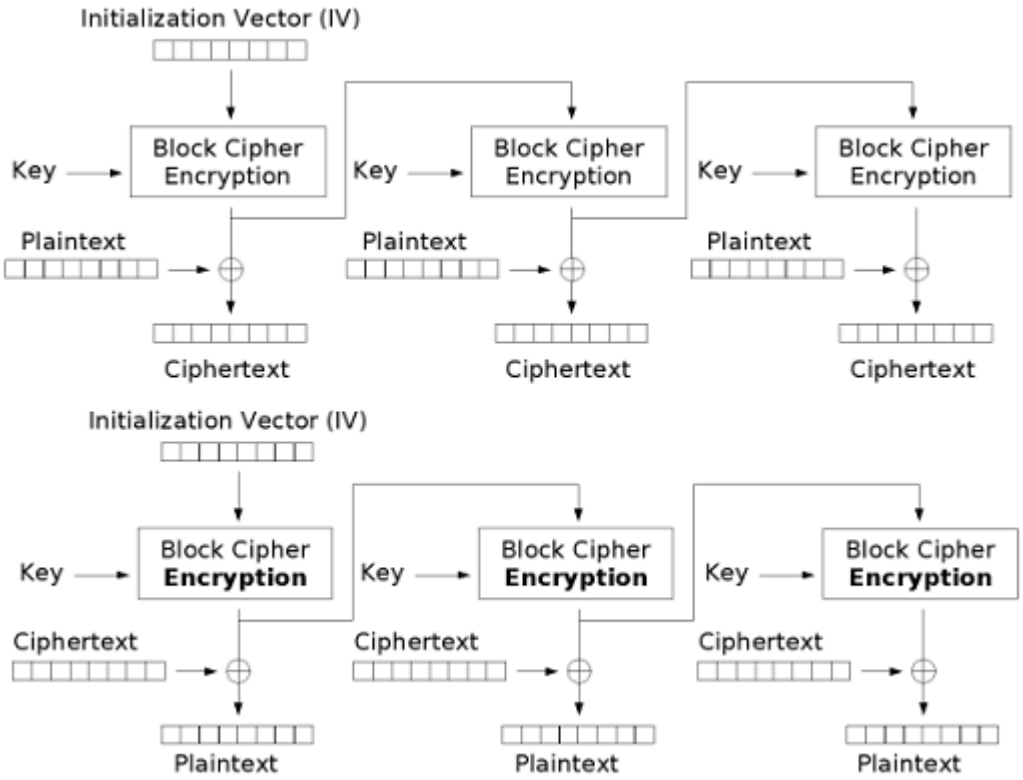

Figura 6: Esquema de xifrat OFB

Igual que en CFB, també es pot veure l'esquema de la següent manera, a la Figura 7:

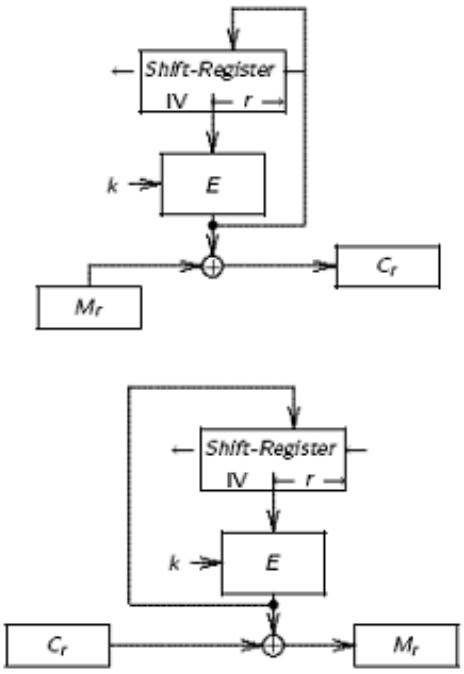

Figura 7: Esquema de xifrat de flux en OFB

Com en CFB, es converteix el xifrador de bloc en un de flux, ja que es pot xifrar dades de mida *r*. Un error de transmissió en un bit concret només afecta a un únic bit al desxifrar el text.

2.1.2. L'algorisme RC5

Introducció i objectius de l'RC5

Aquest algorisme va ser dissenyat per en Ronald Rivest per a l'empresa RSA Data Security. RC5 pertany a la família d'algorismes de clau privada i xifratge en bloc. Els objectius dels seus creadors són els següents (veure [5]):

- L'RC5 és convenient per ser implementat tant en hardware com en software. Es busca utilitzar operacions computacionals primitives que es puguin trobar a la majoria de microprocessadors.
- L'RC5 és ràpid, la qual cosa implica orientar-lo a operar amb paraules com a unitat bàsica de processament. Això implica que el text a xifrar es divideix en blocs d'una llargada fixa.
- L'RC5 és adaptable a processadors amb diferents longituds de paraules. Aquest fet fa que com a paràmetre de l'algorisme es té la longitud de paraula, anomenada *w*.
- L'RC5 té una estructura iterativa amb un nombre variable de passades. Aquest nombre és el segon paràmetre de l'algorisme i s'anomena *r*. Amb aquest paràmetre s'aconsegueix fluctuar entre tenir més velocitat o més seguretat, segons l'aplicació desitjada.
- L'RC5 té una clau de xifrat de longitud variable que permet triar el nivell de seguretat. La longitud de la clau de xifrat és el tercer paràmetre de l'algorisme i s'anomena *b*.
- L'RC5 és fàcil d'implementar gràcies a la seva estructura i no requereix gaire memòria. Aquests fets fan que es pugui utilitzar en entorns amb pocs recursos computacionals i de memòria.
- L'RC5 proporciona una seguretat molt elevada quan es trien els paràmetres *w, r* i *b* adequadament.

- L'RC5 pretén introduir un nou tipus de primitiva criptogràfica anomenada *rotacions dependents de les dades*.

Parametrització de l'RC5

Com ja s'ha dit, l'RC5 té tres paràmetres, *w, r* i *b* que determinen la rapidesa, l'ús de memòria i la seguretat de la implementació d'aquest algorisme, respectivament.

Alhora de tractar el text en clar, s'agrupen les dades en longituds de $2 \cdot w$ bits, ja que *w e*s mesura en bits. Per tant, i posant d'exemple un valor de $w=32$ a cada iteració de l'algorisme es treballa amb 8 bytes de dades del text que es vol xifrar. Tot i que l'algorisme accepta qualsevol longitud de paraula $w > 0$, es recomana treballar amb la longitud de paraula del processador. Avui en dia es parla de valors de *w* de 16, 32 ó 64 bits.

El segon paràmetre de l'RC5, com ja s'ha comentat, determina el nivell de seguretat, més alt com més gran sigui el seu valor. El nombre de passades, *r,* determina la taula estesa de la clau de xifrat, anomenada *S*. El valor d'*r* està comprès entre 0 i 255. Si *r* pren el valor 0, llavors no es xifren les dades, i per valors molt baixos la seguretat és fàcilment violable. D'altra banda, agafant un valor molt alt s'augmentaria el temps de xifrat i desxifrat en gran mesura i no seria usable en molts tipus d'aplicacions.

La taula estesa de la clau es deriva de la clau secreta proporcionada per l'usuari i s'utilitza en les operacions del xifratge i desxifratge del text. La longitud *t* d'aquesta taula es mesura en paraules (de longitud *w*) i es calcula de la següent forma: $t = 2(r+1)$. Per tant, un valor de *r* major a més a més de reduir la velocitat de processat de l'algorisme també n'incrementa els requeriments de memòria.

El paràmetre *b* pot acceptar valors compresos entre 0 i 255, mesurats en bytes, i formen la clau de xifrat, que s'anomena *K*. *K* té els valors *K[0], K[1], ..., K[b-1]*.

Els dos primers paràmetres (*w, r*) són els que determinen l'algorisme en major grau, ja que entre els dos es modifiquen tots els aspectes de l'RC5 (velocitat, ús de memòria i seguretat). En canvi, el tercer paràmetre, *b*, només modifica la seguretat del xifrat.

Quan es parla de l'RC5 es diu que no és un "únic algorisme", ja que segons els tres valors anteriors es té un algorisme diferent. Notacionalment es diu que s'utilitza un RC5-*w/r/b*. Per exemple, si trien els valor *w=32, r=12* i *b=9* s'obté l'RC5 que es pretén intentar trencar en aquest projecte. Els blocs són de 32 bits, per tant l'entrada de dades és de 64 bits (8 bytes), es tenen 12 passades, la clau té 72 bits i la taula estesa mesura 104 bytes.

Se sol considerar un problema d'aquest algorisme el fet de deixar la tria dels paràmetres a qui desenvolupi l'aplicació que l'utilitzi. No obstant, els creadors de l'algorisme recomanen un conjunt de paràmetres idonis que cadascú pot modificar segons el nivell de seguretat i velocitat que requereixi la seva aplicació.

En RC5 s'utilitzen tres primitives, juntament amb les seves inverses:

- **+:** Suma en complement a dos de paraules, mòdul 2^w. La funció inversa es denota amb el signe **-**.
- $\oplus$**:** XOR a nivell de bits.
- **<<<:** Rotació cíclica a l'esquerra d'una paraula. S'interpreta com la rotació d'*y* bits (*y* mòdul *w*) i s'escriu com: $x <<< y$. L'operació inversa es denota com $x >>> y$.

Procés de xifrat i desxifrat

L'RC5 consta de tres components:

- l'expansió de la clau
- el xifrat
- el desxifrat.

Cal recordar que aquest algorisme xifra blocs de $2 \cdot w$ paraules del text en clar i desxifra $2 \cdot w$ paraules del text xifrat. A més. utilitza la taula estesa de la clau, que és el primer pas

que es realitza en l'RC5, i que consta de $t = 2(r + 1)$ paraules de w bits, que s'anomena S. S'assumeix, també, que s'empaqueten els bytes en format *little-endian*.

Expansió de la clau

Aquesta part de l'RC5 transforma la clau K de l'usuari (de b bytes) en la taula S, amb paraules pseudo-aleatòries determinades segons la clau K. L'expansió de la clau consta de tres parts i s'utilitzen dues "constants màgiques" precalculades.

Càlcul de les constants màgiques: Les constants s'anomenen P_w i Q_w i tenen longitud de w bits. Es calculen aplicant les següents fórmules:

$$P_w = Odd((e - 2) \cdot 2^w)$$
$$Q_w = Odd((\Phi - 1) \cdot 2^w)$$

on

$$e = 2.718281828459...$$
$$\Phi = 1.618033988749...$$

On e és la base pels logaritmes neperians, Φ és el nombre auri i la funció $Odd(x)$ calcula l'enter senar més proper al nombre x.

Convertir la clau secreta de bytes a paraules: Primer, es copia la clau secreta $K[0...b-1]$ en un vector $L[0...c-1]$ amb $c = \lceil b/u \rceil$ i $u = w/8$ (que significa el nombre de bytes per cada paraula). S'omple cada paraula d'L amb u bytes consecutius de K. Qualsevol posició que quedi buida en L serà omplerta amb zeros. Pel cas especial en que $b = c = 0$ es suposa $c = 1$ i s'omple $L[0]$ amb zeros.

El pseudocodi de l'algorisme, assumint que els bytes no tenen signe i que inicialment L està inicialitzada a 0, és el següent:

```
c = ⌈max(b,1)/u⌉
per i = b - 1 fins 0 fer
        L[i/u] = (L[i/u] <<< 8) + K[i]
fi
```

Es pot veure que *L[0..c-1]* inicialment és 0, i s'introdueix la clau a mesura que es van rotant els valors de *L*.

Inicialitzar el vector *S*: En el segon pas de l'expansió de la clau s'inicialitza el vector S amb un patró de bits pseudo-aleatoris prefixat, utilitzant una progressió aritmètica mòdul 2^w determinada per les constants P_w i Q_w que ja es tenen precalculades.
L'algorisme és el següent:

```
S[0] = Pw
per i = 1 fins t - 1 fer
        S[i] = S[i – 1] + Qw
fi
```

Introduir la clau *K*: En el tercer i últim pas s'introdueix la clau *K* en tres passos entre els vectors *S* i *L*. Donat que *S* i *L* tindran mides diferents, el més llarg serà processat tres vegades mentre que l'altre més cops.
L'algorisme és el següent:

```
i = j = 0
A = B = 0
fer 3 * max(t, c) vegades
        A = S[i] = (S[i] + A + B) <<< 3
        B = L[j] = (L[j] + A+ B) <<< (A + B)
        i = (i + 1) mod t
        j = (j + 1) mod c
fi
```

A partir de la taula calculada es fa difícil determinar la clau *K* ja que aquesta funció en certa manera és força unidireccional.

Xifrat

Assumint que es tenen les dues paraules de mida *w* a xifrar en els registres *A* i *B*, el pseudocodi de l'algorisme de xifrat és el següent:

```
A = A + S[0]
B = B + S[1]
per i = 1 fins r fer
        A = ((A ⊕ B) <<< B) + S[2*i]
        B = ((B ⊕ A) <<< A) + S[2*i+1]
fi
```

Es pot veure com les rotacions dependents de dades anteriorment esmentades juguen un paper fonamental en el procés de xifrat. El resultat es guarda en els registres *A* i *B*.

Desxifrat

El pseudocodi és el següent:

```
per i = r fins 1 fer
        B = ((B – S[2*i+1]) >>> A) ⊕ A
        A = ((A – S[2*i]) >>> B) ⊕ B
fi
B = B - S[1]
A = A - S[0]
```

El resultat també queda guardat en els registres *A* i *B*.

Es pot consultar un exemple d'aquest procés a l'annex B.

Anàlisi del procés

El punt fort de l'RC5 són les rotacions dependents de les dades que es duen a terme. Aquestes rotacions no estan predeterminades i depenen de les dades d'entrada. En el disseny de l'algorisme els seus autors buscaven provar la fortalesa criptogràfica d'aquestes rotacions.

Un tros de la taula estesa s'afegeix inicialment a les dades en clar i cada passada de l'algorisme de xifrat afegeix un altre tros de la taula als resultats parcials calculats. Aquestes operacions permeten que cada passada actuï de forma força diferent, segons la quantitat de rotacions utilitzades. L'operació d'XOR té la propietat d'allau ja que cada bit de l'entrada causa canvis a diversos bits en passades posteriors.

L'algorisme de xifrat i desxifrat és molt compacte i pot ésser codificat d'una manera molt eficient en llenguatge assemblador en força processadors. La taula *S* és força petita i s'hi accedeix seqüencialment, la qual cosa minimitza el *hit-ratio* a la memòria cau.

2.2 La suposició de Kerckhoffs per a atacs de força bruta

La suposició de Kerckhoffs és en la que es basa aquest projecte pel trencament del criptosistema proposat per RSA Laboratories. Kerckhoffs va postular que un criptosistema ha de ser segur encara que es conegui tot sobre el sistema, excepte la clau. En altres paraules, un criptoanalista pot saber amb quin algorisme s'ha xifrat el text en clar, quin mode de xifrat s'ha utilitzat (en cas de criptografia de clau privada, que és el cas d'aquest projecte) i fins i tot el vector d'inicialització utilitzat. Sempre i quan el que no es conegui sigui la clau, i evidentment el text en clar, el criptosistema ha de romandre indesxifrable. Shannon ho va formular com "l'enemic coneix el sistema".

Kerckhoffs va escriure sis lleis alhora de dissenyar un bon algorisme de xifrat, una de les quals és la suposició de la que s'està parlant. Aquestes lleis són les següents (veure [6]):

- El sistema ha de ser indesxifrable a la pràctica, sinó matemàticament.
- No es requereix que sigui secret, hauria de poder caure a les mans de l'enemic sense cap inconvenient.
- La clau ha de ser comunicable i retenible sense l'ajuda de notes escrites i a més a més modificable segons la voluntat de les parts implicades.
- S'ha de poder aplicar a correspondència telegràfica.
- Ha de ser portable, i el seu ús i funcionalitat no ha de requerir la participació de diverses persones.
- Finalment, és necessari que sigui fàcil d'utilitzar donades les circumstàncies que comanden la seva aplicació. No ha de requerir grans esforços mentals o el coneixement de grans conjunts de normes.

Aquests principis van ésser dissenyats per aplicar-los a xifradors militars, tot i que contrasta amb la seguretat a través de l'obscuritat, que seria el que utilitzarien els governs o els militars per protegir la seva informació.

Quan es diu que el sistema ha de ser indesxifrable a la pràctica, significa que només es poden realitzar atacs de força bruta per trencar-lo. Aquest fet exclou qualsevol tipus d'atac de criptoanàlisi clàssic o modern, com l'anàlisi de freqüències, criptoanàlisi diferencial, lineal, integral, etc.

Un atac per força bruta consisteix en provar exhaustivament totes les claus possibles. Per exemple, en el cas del RC5-32/12/9, es tenen 2^{72} claus possibles per provar. A la pràctica, s'espera trobar la clau provant la meitat de les claus de l'espai de claus, 2^{71} en aquest cas. Com s'ha vist a la introducció, el cas que ocupa aquest projecte és un xic especial, ja que es disposa de part del text desxifrat, i del vector d'inicialització. Usualment la única informació que es té per realitzar atacs per força bruta és el text xifrat i la informació sobre l'algorisme de xifrat, segons la suposició de Kerckhoffs.

Llavors, per poder realitzar atacs de força bruta, es necessita alguna manera d'analitzar el text desxifrat a cada intent, com per exemple un reconeixedor de text[11].

[11]Es podrien utilitzar tècniques d'Intel·ligència Artificial. Si es sap que el text és en anglès, per exemple, es podria usar un classificador de Bayes.

3. Càlcul distribuït per a la realització d'atacs

Abans d'explicar com s'aplica el càlcul distribuït per a la realització d'atacs de força bruta a sistemes de xifrat, cal explicar què són els propis sistemes distribuïts i el càlcul distribuït.

3.1. Els sistemes distribuïts i el càlcul distribuït

Un sistema distribuït es pot definir segons dos punts de vista [7]:

- Des d'un punt de vista físic, un sistema distribuït és un conjunt de processadors, possiblement heterogenis, sense memòria ni rellotge comú, que es troben connectats a través d'una xarxa.
- Des del punt de vista lògic, un sistema distribuït és un conjunt de processos que s'executen en una o més computadores i que col·laboren i es comuniquen entre ells mitjançant l'intercanvi de missatges.

El càlcul distribuït es defineix com el mètode en que diverses parts d'un programa s'executen simultàniament en dos o més ordinadors que es comuniquen per la xarxa [8]. Aquest tipus de càlcul requereix que es tingui en compte que els diferents ordinadors que formen el sistema distribuït poden tenir arquitectures i sistemes operatius diferents. Es diu que el càlcul distribuït és el resultat d'utilitzar les xarxes per permetre als ordinadors comunicar-se eficientment.

Les aplicacions que utilitzen càlcul distribuït tenen com a característica principal la resolució de problemes que necessiten grans quantitats de càlcul per a ser resolts. Aquestes aplicacions solen dividir el problema en petites parts que són enviades a les diferents màquines que composen el sistema distribuït, i posteriorment, combinant les solucions parcials, s'obté la solució al problema. Aquests tipus de problemes necessiten tanta potència computacional que són impossibles de resoldre per una sola màquina, o un conjunt reduït de màquines amb una quantitat de temps raonable i que, per tant, necessiten les avantatges que el càlcul distribuït ofereix.

Un aspecte molt important dels sistemes distribuïts és la interacció entre les diferents màquines. Per aconseguir la màxima compatibilitat, el canal de comunicació no ha de contenir informació que no pugui ser entesa per algunes de les màquines. Un altre aspecte molt important és la manera en que es porta el software entre dues màquines. En cas d'utilitzar arquitectures diferents, no sempre és possible fer una portabilitat directa i en ocasions pot ser necessari utilitzar tècniques de compilació creuada del software, o portar el software manualment.

Avantatges i desavantatges

Els sistemes de càlcul distribuït presenten un seguit d'avantatges i desavantatges que es descriuen a continuació.

Entre els avantatges dels sistemes distribuïts es troba la transparència amb la que es poden connectar els usuaris i els seus recursos. A més, aquest tipus de sistemes solen ser sistemes oberts i escalables i, idealment, tenen molta tolerància a errors. Aquesta propietat de ser sistemes oberts, però, implica que poden interactuar contínuament amb altres subsistemes i també té implicacions en l'estabilitat de la connectivitat. L'escalabilitat del sistema implica que poden haver-hi canvis en el nombre d'usuaris i recursos, de manera que el sistema es pot adaptar, per exemple, a un increment notable en el nombre d'usuaris, sense que el rendiment es vegi afectat.

Com tot sistema, el càlcul distribuït també té els seus desavantatges. Si no es planifica bé, un sistema distribuït pot causar una baixada del rendiment dels càlculs si la indisponibilitat d'un node arriba a causar la interrupció del treball d'altres nodes. Una altre gran desavantatge d'aquest tipus de sistema és la dificultat de diagnosticar i localitzar els problemes i averies que puguin ocórrer. Aquest fet és causat per la necessitat d'analitzar les comunicacions entre nodes o inspeccionar els nodes remots on es pugin produir aquestes fallades. Un altre possible problema apareix en el cas que les aplicacions tinguin requisits importants per comunicar i sincronitzar els diferents nodes. Mai es pot suposar que l'ample de banda és infinit, o que les connexions a la xarxa són permanents alhora de programar aplicacions per aquest tipus de càlcul. Aquest últim

desavantatge pot fer que els beneficis del càlcul distribuït no siguin tals i siguin preferibles els sistemes no distribuïts.

Arquitectura

A baix nivell es parla d'interconnectar diverses CPUs a través d'algun tipus de xarxa, mentre que a més alt nivell es fa l'abstracció de la connexió de processos entre aquestes CPUs amb algun tipus de sistema de comunicació. La programació distribuïda es pot classificar dins d'algun dels següents models: client-servidor, arquitectures de 3 o més capes, objectes distribuïts, dèbilment o forta acoblats i *peer-to-peer* [8].

El paradigma client-servidor és el típic en que la màquina client contacta amb el servidor per a que aquest li proporcioni dades i software per realitzar els càlculs pertinents i un cop finalitzat el càlcul el client li retorna el resultat. Com es veurà més endavant, BOINC és del tipus client-servidor. Les arquitectures de tres o més capes mouen la intel·ligència del client a una capa intermèdia, moltes aplicacions web són d'aquest tipus. Es diferencien de l'arquitectura client-servidor en que la interfície d'usuari, la lògica de procés, l'emmagatzemament de dades i l'accés a dades es mantenen en mòduls separats. En els sistemes dèbilment i forta acoblats es parla de la integració de les diferents parts que formen el sistema. Un exemple de sistemes fortament acoblats serien els clústers. Finalment, en l'esquema *peer-to-peer* es parla d'una arquitectura en que cap màquina gestiona els recursos, sinó que les responsabilitats es divideixen entre totes les màquines que formen el sistema.

Càlcul amb recursos públics

Al llarg d'aquest apartat s'ha parlat sobre sistemes distribuïts però no s'ha esmentat d'on provenen els recursos computacionals que s'utilitzen. Tant pel cas que interessa a aquest projecte, com per tots els projectes BOINC en general, la majoria de recursos solen ser cedits voluntàriament per les persones que hi participen. A aquest tipus de càlcul se'l sol anomenar *Public Resource Computing* (PRC).

PRC descriu un model computacional introduït pel projecte *distribuited.net*[12], del qual se'n parlarà més endavant. La idea consisteix en aconseguir que gent amb connexió a Internet i recursos computacionals, com qualsevol ordinador de sobretaula, clústers o superordinadors, cedeixi desinteressadament cicles de CPU del seu ordinador per algun projecte de computació distribuïda [9]. L'ús dels recursos computacionals que d'altra manera serien desaprofitats s'anomena "collita de cicles".

Les plataformes PRC presenten diversos problemes seriosos. D'una banda, en un sistema PRC en principi no es disposa dels recursos dedicats habituals (supercomputadors, clústers, etc) i per tant, els dispositius de que es disposa solen estar connectats a través de mòdems només una fracció del temps i per tant no es pot assumir que aquests recursos estan sempre disponibles. A més, s'ha de suposar que molts recursos poden estar en una xarxa local (possiblement utilitzant NAT i sense una adreça IP pública vàlida) i per tant no es pot assumir que aquests recursos estan llestos per acceptar connexions entrants del servidor del sistema distribuït.

D'altra banda, amb la PRC apareix una qüestió de confiança dels recursos. I és que mentre que els recursos dedicats solen ser administrats per la pròpia organització, o per alguna organització de confiança, els recursos públics poden estar situats a qualsevol llar i per tant sense supervisió dels responsables dels projectes. Per tant, el resultat de l'execució de la tasca pot ésser assumit com a vàlid o no, podent arribar a executar una mateixa tasca diverses vegades, per comparar els resultats. La confiança, però, juga un paper en les dues bandes, ja que el client ha de confiar en que el projecte no inclourà en les seves tasques cap mena de codi maliciós que, per exemple, pugui robar contrasenyes o destruir dades de l'ordinador. Per garantir la seguretat del recurs, molts PRCs utilitzen una tècnica anomenada *sandbox*. Aquesta tècnica aïlla el software que s'executa al client de la resta de processos que s'hi executen. D'aquesta manera es preveu qualsevol risc de danyar tot el que quedi fora de la *sandbox*. Habitualment s'intercepten les crides al sistema, afegint una capa entre l'aplicació i el sistema operatiu, decidint en cada moment quan una crida al sistema és segura o no. En el capítol en el que es descriu la BOINC es veurà com es fa ús d'aquesta tècnica.

[12] També es coneix PRC com "volunteer computing", originalment fou introduït pel projecte Great Internet Mersenne Prime Search però atribuït a distribuited.net. El terme va ser encunyat per Luis F. G. Sarmenta [9].

Un altre dels principals problemes de la PRC és que els clients poden necessitar una raó per cedir els seus recursos computacionals sobrants. Segons el projecte, es pot arribar a pagar pel servei prestat, tot i que la majoria no ho fan. Altres raons per col·laborar en aquests tipus de projectes de computació distribuïda són la motivació per ajudar a la ciència, com per exemple en càlculs que permeten desenvolupar cures contra malalties.

3.2. Avantatges del càlcul distribuït per a la realització d'atacs

S'ha vist que el càlcul distribuït és molt útil per resoldre problemes que tenen grans necessitats de realitzar càlculs. El tipus de problema amb el que s'enfronta aquest projecte, el de realitzar un atac per força bruta a un criptosistema, és precisament un d'aquests tipus de problemes. No es requereixen grans quantitats de dades, doncs de fet, amb dues paraules de 32 bits i el conjunt de claus n'hi ha suficient per realitzar aquest tipus d'atac.

Si bé és cert que hi ha una gran quantitat de claus per provar, 2^{72} claus en total, aquest fet no suposa un problema alhora de dividir el problema en trossos petits, ja que cada tros necessitarà conèixer només un rang de claus a provar. El problema està en que, tot i que l'algorisme sembli d'allò més senzill, com s'ha vist a l'apartat 2.1.2, provar un gran volum de claus acaba sent una tasca molt feixuga per una sola màquina.

Per fer-se una idea, només per provar 2^{32} claus, es necessita de l'ordre de 3.400 segons en un PC amb dos processadors a 3.0 GHz[13], utilitzant el sistema operatiu GNU/Linux[14], i deixant aquest PC dedicat únicament a aquest càlcul. Fent la divisió en paquets de 2^{32} claus es té un total de 2^{40} paquets, que processats en sèrie es tardaria de l'ordre de 122 milers anys. Per tant, la conclusió que es pot extreure d'intentar fer un atac per força bruta provant totes les claus en sèrie seria que l'intent és del tot fútil.

Aleshores, utilitzant el càlcul distribuït, si s'envien paquets de claus a cada màquina i aquestes es posen a processar-los en paral·lel, el temps total per trencar el criptosistema

[13] Les dades sobre la memòria són irrellevants per aquest problema.
[14] Més endavant es veurà que aquesta dada sí té molta rellevància.

es pot arribar a reduir dràsticament. A més a més, la comunicació entre els diferents paquets de treball és nul·la, per tant els únics moments en que es necessitaria utilitzar la xarxa seria per descarregar-se la informació d'entrada i per enviar els resultats. Aquest fet facilita la tasca del propi sistema distribuït, que cal recordar que quanta més comunicació i sincronització necessita més disminueix el rendiment que ofereix. Per tant, una altre avantatge d'usar sistemes distribuïts per atacs per força bruta és el d'aprofitar el màxim rendiment que un sistema d'aquest tipus pot oferir.

Cal fixar-se però, que tot i haver anomenat que el gran avantatge del càlcul distribuït és el de reduir el temps de processat del conjunt de claus, s'ha de recordar que per aconseguir-lo reduir sensiblement, es poden arribar a necessitar molts milions d'instruccions per segon (MIPS) de potència de processat. Fent unes senzilles estimacions, si el PC utilitzat anteriorment té una potència de 6000 MIPS per processador i es necessitarien 122 milers d'anys aproximadament per trencar el criptosistema, si es necessités trencar-lo en, com a màxim, un any, es necessitarien uns 1.464.000.000.000 MIPS en potència de càlcul com a mínim. Si es suposa que de mitjana un processador té 6.000 MIPS, es necessitarien 244.000.000 processadors.

En aquest punt és on entra el concepte de PRC. Es poden arribar a aconseguir molts recursos si se li dóna una mica de publicitat al projecte a través d'Internet. Fins al punt de poder arribar a aconseguir fins a 100 o 150 TeraFLOPS com els projectes de SETI@home o Folding@home, respectivament. Tot i que aquests projectes tinguin una causa força més noble que la de trencar criptosistemes.

3.3. Eines de càlcul distribuït existents

Actualment no es troben gaires eines de càlcul distribuït, ja que aquest tipus d'eina no és d'ús comú, sinó que només se sol utilitzar per un tipus de projectes amb moltes necessitats de càlcul. L'eina més famosa de PRC és, evidentment, la Berkeley Open Infrastructure for Network Computing (BOINC), de la que se'n parlarà extensament a l'apartat 4. Tanmateix, com a referència històrica, es pot dir que va ser creada per la Universitat de Berkeley per estudiar les mostres recollides en un observatori en busca

d'intel·ligència extraterrestre. Al principi de ser creada, al 1999, només funcionava per a aquest projecte, ja que estava dissenyada específicament per les seves necessitats. No va ser fins al 2003 que es va alliberar la versió 2.0 de la BOINC i poc després l'equip de SETI@home va començar-lo a utilitzar, substituint la versió anomenada *"classic"*.

Entre les altres eines més conegudes es troba la Distributed Computing Environment (DCE) [10], desenvolupada a principis del 1990 per un consorci que incloïa l'Apollo Computer, IBM i Digital Equipment Corporation entre altres. La DCE proveeix d'un *framework* per a desenvolupar aplicacions client-servidor, que inclou un mecanisme de crides a procediments remots (RPC) anomenat DCE/RPC, un servei de directori de noms, un servei de temps, un d'autenticació, un d'autorització, i un sistema distribuït de fitxers, conegut com DCE/DFS.

Un altra de les grans eines de càlcul distribuït és la Globus Toolkit [11], desenvolupada per la Globus Alliance, i actualment en la seva quarta versió. Tot i no ser estrictament per càlcul distribuït, sinó per Grids, cal mencionar-la en aquest treball, ja que és un conjunt d'eines de codi lliure que és considerat pràcticament un estàndard. Cal remarcar que no és una infraestructura Grid complerta però els seus elements es poden identificar com blocs bàsics per la construcció d'un Grid. Les eines existents de la Globus Toolkit es divideixen en les categories de seguretat, gestió de dades, gestió de l'execució, serveis d'informació i rutines d'execució.

També per Grids i de codi lliure, existeix una eina anomenada Condor [12], que és un *framework* per a la paralel·lització distribuïda de tasques computacionalment intensives. La desenvolupa la Universitat de Wisconsin-Madison i està dissenyada per gestionar la càrrega de treball en clústers dedicats i en recursos no dedicats com ordinadors personals. Per exemple, a les instal·lacions de Supercomputació Avançada de la NASA hi tenen una piscina de màquines amb 350 *workstations* SGI i Sun.

Una altra eina per Grids, però comercial, és la Frontier® Grid Platform [13], desenvolupada per la companyia Parabon Computation. La seva primera versió data del 2000, i aquesta companyia ha desenvolupat un conjunt d'eines centrades en aquest producte. També comercial, existeix una eina anomenada Grid MP [14], de la companyia United Devices, que té com a última versió la 5.5. Les característiques de

Grid MP són la planificació de tasques amb prioritització, les restriccions en seguretat per l'usuari, l'exclusió selectiva d'aplicacions, la detecció d'activitat de l'usuari i el control de temps en l'execució. Aquesta eina pot usar-se per gestionar PCs corporatius, servidors departamentals i nodes de clústers dedicats.

Més com una curiositat, anomenar també una extensió pels navegadors de la barra de Google anomenada Google Compute, que permet participar en projectes de càlcul distribuït com Folding@home [15] Cal puntualitzar, però, que ja no es troba disponible.

3.4. Projectes semblants a Internet

Els projectes semblants al que es presenta en aquesta memòria es poden dividir en dos grans grups: els projectes que tracten sobre criptografia i els que utilitzen BOINC com a eina de càlcul distribuït. Entre els primers s'hi troben projectes com els de la pàgina web *distribuited.net*. Des de *distribuited.net* també intenten trencar el mateix criptosistema proposat per RSA Laboratories. El mateix equip també participa en el repte de la regla de Golomb òptima. En el segon grup es trobarien projectes com SETI@home, Folding@Home o Einstein@home.

Projectes que utilitzin BOINC n'hi ha molts, però que hagin assolit una fama i renom que es tradueixi en una gran potència de càlcul no tants. El més famós de tots, SETI@home, desenvolupador, a més, de la pròpia infraestructura BOINC, tracta de buscar intel·ligència extraterrestre a l'espai exterior. De tots els projectes d'aquest tipus, és el que més potència de càlcul té, tot i que a dia d'avui no ha aconseguit cap resultat concloent que permeti afirmar l'existència de vida extraterrestre intel·ligent. Per ordre d'importància es troba, en segon lloc, el projecte Folding@home, dedicat a la recerca en biomedicina. Tracta de simular el plegament de proteïnes, per intentar comprendre el desenvolupament de moltes malalties. També s'ha fet famós per ser el primer en aprofitar la potència de càlcul del processador CELL de la PlayStation® 3. Finalment, val la pena mencionar també el projecte Einstein@home, que tracta d'analitzar ones gravitacionals en dades recollides pels observatoris LIGO. A la Taula 3 es pot veure una

comparativa del nombre de hosts dels majors projectes que utilitzen BOINC i la potència de càlcul en FLOPS[15] [16].

Projecte	Àrea de recerca	#hosts	Potència de càlcul
SETI@home	Intel·ligència extraterrestre	850.000	100 TeraFLOPS
Folding@home	biomedicina	200.000	120 TeraFLOPS
Einstein@home	astrofísica	200.000	80.5 TeraFLOPS
Rosetta@home	Biologia	100.000	38 TeraFLOPS
LHC@home	Física	60.000	-

Taula 3: Comparativa de hosts i potència de càlcul dels principals projectes en BOINC

Finalment, projectes sobre criptografia famosos se'n troben més aviat pocs, ja que lògicament no són tan interessants per a la gent com projectes de recerca d'intel·ligència extraterrestre o medicina, i la majoria es desenvolupen en entorns militars i sota secret. El més famós de tots és el de l'equip de *distribuited.net* [17]. Van començar l'any 1997 a intentar trencar un dels desafiaments d'RSA Laboratories, concretament el que tenia com a algorisme l'RC5-32/12/7, o sigui, amb una clau secreta de 56 bits. Després de 212 dies i un total del 47.03% de l'espai de claus provat van aconseguir trencar el criptosistema. A continuació van començar amb el desafiament immediatament superior, amb clau de 64 bits. En aquesta ocasió varen necessitar 1757 dies, provant un total de més del 63% de l'espai de claus. Mentre participaven en el desafiament de l'RC5-32/12/8 també van participar en el desafiament DES-II-1 i DES-II-2, tot i que només van acabar a temps en el primer d'ells. Altres desafiaments en que participen, tot i que no d'RSA Laboratories, són sobre les regles òptimes de Golomb[16].

Al acabar el desafiament de l'RC5-32/12/8 van començar amb el mateix que ocupa el projecte presentat en aquesta memòria, l'RC5-32/12/9. El dia d'inici va ser el 3 de desembre de 2002 i, tanmateix, a dia d'avui només porten un 0.420% de l'espai de claus completat, que es tradueix en un total de 19.811.569.592.434.687.000 claus provades, amb un total de 76.145 participants inscrits des de l'inici del projecte i 6.181 usuaris actius. Aquestes dades, com s'ha vist a la introducció, treuen qualsevol expectativa d'aquest projecte en completar el repte d'RSA Laboratories.

[15] Dades segons la Wikipèdia anglesa.
[16] Que tracten de trobar marques sobre una regla imaginària de manera que cap parell de marques tingui la mateixa distància. Una regla de Golomb es diu que és perfecta si es mesuren totes les distàncies fins la seva llargada, i és òptima si no existeix una regla de Golomb del mateix ordre més curta.

4. El sistema distribuït BOINC

4.1. Introducció a la BOINC

La Berkeley Open Infrastructure for Network Computing (BOINC) és una plataforma oberta middleware de càlcul distribuït que utilitza recursos computacionals cedits voluntàriament per altra gent [18], Public Resource Computing (PRC). La desenvolupa l'equip de SETI@Home, de la Universitat de Berkeley, i actualment la fan servir nombrosos projectes.

La BOINC es pot utilitzar en projectes de molts tipus, i cada projecte és independent i es realitza amb els seus propis servidors i bases de dades. La gent que cedeix el seu ordinador voluntàriament pot participar en els projectes que desitgi, i pot distribuir el temps i recursos de que disposa segons les seves preferències. Si un dels projectes en que els voluntaris participen està fora de servei o no té tasques pendents per repartir, els recursos dels voluntaris es divideixen entre els altres projectes en que estigui participant.

La BOINC proveeix certes característiques que simplifiquen la creació i execució dels projectes. Les aplicacions ja existents programades en llenguatges comuns poden funcionar en BOINC amb petites modificacions o fins i tot sense cap modificació[17]. La BOINC també protegeix als voluntaris contra força tipus d'atacs i, a més a més, utilitza un esquema de signatura digital de protecció contra la distribució de virus.

La BOINC també es pot utilitzar en aplicacions que necessiten produir moltes dades o utilitzar molts recursos de memòria. Els voluntaris, però, podran decidir quin ample de banda i ús de disc dur pot utilitzar l'aplicació, de tal manera que el servidor ha de decidir a quins voluntaris pot enviar cada tasca segons si la poden gestionar o no.

L'ús d'una plataforma hardware o sistema operatiu concret no sol ser impediment pels participants d'un projecte ja que BOINC està disponible en les plataformes més

[17] Es consideren modificacions a nivell algorítmic, no de codi, que sempre seran necessàries.

comuns. A més a més, BOINC proporciona un conjunt d'eines web per poder-se registrar al projecte, editar les preferències (com la d'ample de banda i disc dur disponibles abans comentades) i per mostrar l'estat del participant. Si un participant disposa de diverses màquines, l'edició de les seves preferències s'aplicarà per totes elles.

Per minimitzar les connexions al servidor del projecte, el client de BOINC pot proporcionar, segons les preferències del participant, la suficient quantitat de treball per mantenir ocupada la màquina mentre el projecte estigui fora de servei o el client no es pugui connectar.

A més a més, els participants es poden organitzar en grups de treball, que poden arribar a competir per oferir el màxim del seus recursos disponibles a cert o certs projectes. Els grups representen un al·licient més pels participants per fomentar la participació en projectes basats en PRC.

Característiques de les aplicacions

Les aplicacions que es vulguin executar per mitjà de la BOINC han de complir certes característiques per tal que l'esforç de posar en marxa la infraestructura estigui justificat.

En primer lloc, les aplicacions han de tenir unes necessitats de càlcul o d'emmagatzemament molt grans. Un inconvenient del càlcul distribuït mitjançant recursos públics és que les aplicacions han de poder acceptar certa tolerància a errades, ja que els resultats proporcionats pels participants del projecte no es poden considerar absolutament fiables. Més endavant es veuran de quins mecanismes disposa la BOINC per solucionar aquest problema.

Una altra característica necessària és que els càlculs de l'aplicació es puguin dividir en tasques paral·lelitzables i amb poques o nul·les dependències de dades. Tot i que un projecte en BOINC pot ésser creat per necessitar molta capacitat d'emmagatzemament, un inconvenient a aquest fet és que en moltes ocasions les connexions a Internet dels

voluntaris poden ser de cobrament segons el volum de dades descarregat i/o enviat, la qual cosa pot resultar força negativa pel projecte.

Per últim, és important mencionar que el principal inconvenient que poden tenir moltes aplicacions basades en PRC és que no siguin prou atractives per a que els voluntaris s'afegeixin al projecte i cedeixin els seus recursos.

4.2. Estructura i components de la BOINC

Com s'ha esmentat en l'apartat anterior, la BOINC és un sistema distribuït del tipus client-servidor. La BOINC proporciona un client que és compartit per tots els projectes BOINC, un servidor específic i altres aplicacions dependents del projecte en sí. La Figura 8 mostra una vista sobre com s'estructura la plataforma.

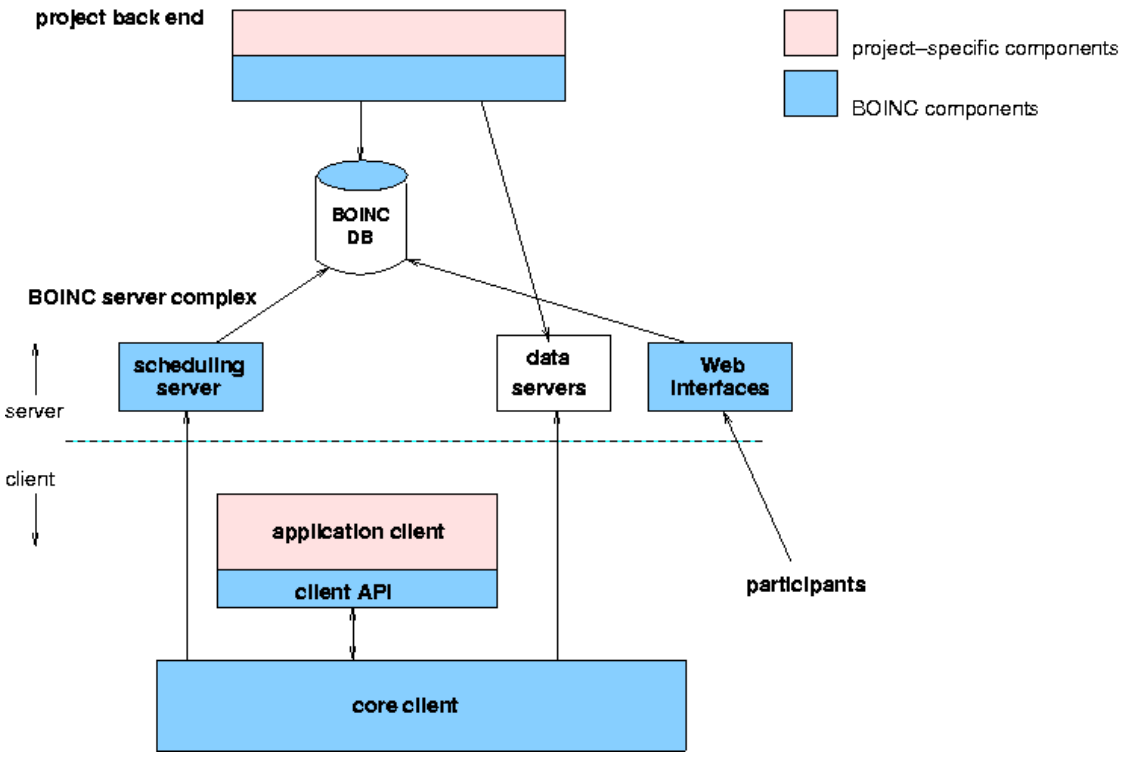

Figura 8: Estructura de la BOINC

Com es pot veure, a l'entorn del client es disposa del propi client, la seva API, i l'aplicació client, que el servidor enviarà a aquest quan es connecti al projecte.

A l'entorn del servidor, es disposa d'un servidor de planificació, on l'aplicació client es connecta per demanar tasques, uns servidors de dades, des d'on es descarreguen les aplicacions i les dades de cada tasca. També es disposa d'un servidor web, on el client pot consultar tota la informació disponible del projecte. La base de dades és el pilar fonamental de tot projecte, ja que emmagatzema tota la informació necessària pel seu correcte funcionament: les tasques, els clients registrats, etc. Per tal de gestionar la base de dades i processar tota la informació del projecte hi ha dimonis i aplicacions servidor. Alguns d'aquests dimonis i aplicacions servidor són genèrics a tots els projectes i alguns s'han de dissenyar i implementar pels propis creadors del projecte, ja que s'hauran d'ajustar a les necessitats de cada aplicació client en concret.

4.2.1. El Client

Els usuaris que volen ser voluntaris en algun projecte BOINC s'han de descarregar i instal·lar el client, anomenat *core client*. El client es pot descarregar de la pròpia pàgina web del projecte o bé de la pàgina oficial de la BOINC. Després d'instal·lar-lo el voluntari s'ha de registrar a la pàgina web del projecte o des del mateix client com a usuari. Si el registre és correcte, l'usuari rep un correu electrònic amb el seu identificador d'usuari. Amb aquest nombre d'identificació i la URL del projecte l'usuari pot identificar-se al client i començar a cedir els seus recursos computacionals.

Per protegir els recursos del client de l'execució de codi maliciós, el client executa les aplicacions en un entorn anomenat de "*sandbox*". Aquest entorn intercepta totes les crides al sistema de l'aplicació i comprova que no posin en perill el sistema. D'aquesta manera, les aplicacions s'executen en un entorn aïllat.

El client permet identificar-se al voluntari en diversos projectes i decidir el percentatge de temps i recursos que vol dedicar a cada projecte. L'usuari no necessita fer res més ja que el client s'encarrega automàticament de descarregar-se les dades necessàries i l'aplicació del projecte, així com d'enviar les dades resultants un cop duta a terme la tasca assignada. En resum, el client automatitza tota la part de negociació amb el servidor, de manera totalment transparent a l'usuari.

A través de les preferències de l'aplicació client, l'usuari pot controlar moltes opcions, tan generals com específiques del projecte. Algunes de les opcions són:

- especificar el moment en que s'executaran les tasques de cada projecte: mentre l'usuari està actiu al seu ordinador o després d'un temps d'inactivitat prefixat.
- definir el mínim i màxim de treball que s'ha de mantenir a l'ordinador per cada projecte.

Si la quantitat de treball disponible és inferior al mínim, el client contacta amb el projecte per demanar la quantitat suficient de treball per arribar al màxim definit. També s'aprofita la connexió per publicar els resultats de les tasques dutes a terme des de la última connexió i per comprovar si hi ha noves versions de l'aplicació. Els resultats es gestionen en dos passos. Primer, tan aviat com el resultat està disponible i es disposa de connexió a Internet s'envia. Després, durant la sessió de planificació (*scheduling*) el resultat és publicat. Aquest esquema comporta el problema que un resultat podria passar dies en el propi client o en el servidor web sense que el planificador en tingui notícies.

Plataformes

En l'entorn de la BOINC, es considera una plataforma com una combinació d'arquitectura CPU i sistema operatiu determinats. Donat que la plataforma que pot tenir un voluntari per executar el client de la BOINC pot tenir una configuració determinada, la BOINC permet que els clients s'executin en un nombre variat de configuracions diferents. En concret, es recomana utilitzar una de les següents plataformes [19]:

nom	Descripció
windows_intelx86	Microsoft Windows (98 o posterior) sobre Intel x86 o CPU compatible
Windows_x86_64	Microsoft Windows sobre AMD x86_64 o Intel CPU EM64T
i686-pc-linux-gnu	Linux sobre Intel x86 o CPU compatible
x86_64-pc-linux-gnu	Linux sobre AMD x86_64 o Intel CPU EM64T
ppc64-linux-gnu	Linux sobre un processador PowerPC de 64 bits
powerpc-apple-darwin	Mac OS X 10.3 o posterior sobre Motorola PowerPC
i686-apple-darwin	Mac OS 10.4 o posterior sobre Intel
sparc-sun-solaris2.7	Solaris 2.7 sobre SPARC o CPU compatible
sparc-sun-solaris	Solaris 2.8 o posterior sobre SPARC- o CPU compatible
sparc64-sun-solaris	Solaris 2.8 o posterior sobre SPARC amb CPU de 64-bits

Taula 4: Plataformes en BOINC

El nom de la plataforma es guarda al client, que informarà al servidor de planificació sobre aquestes dades per a que aquest li faciliti la versió d'aplicació corresponent, compatible amb la plataforma. Les plataformes suportades per un projecte es defineixen al fitxer anomenat *project.xml,* que més endavant es veurà quina informació conté. No és necessari que un projecte suporti totes aquestes plataformes, ja que podria ser que els creadors d'un projecte no disposin de les eines per compilar les seves aplicacions client en alguna de les plataformes concretes. També es pot donar el cas en que una aplicació estigui suportada en un subconjunt de les plataformes que apareixen al projecte, pel motiu que sigui. Aquest fet, no suposa cap inconvenient ja que el planificador sap en tot moment en quines plataformes es té una versió de l'aplicació corresponent.

4.2.2. El Servidor

El servidor de la BOINC consisteix, com a mínim, en un servidor web i una base de dades. També es necessiten cinc dimonis que comproven periòdicament l'estat de la base de dades i duen a terme qualsevol tasca associada a la seva responsabilitat. Aquests programes poden funcionar sobre la mateixa màquina o poden estar distribuïts en diferents servidors, per raons de rendiment.

Un projecte BOINC té una estructura de directoris genèrica. Sota el directori arrel del projecte es troben un seguit de directoris per emmagatzemar les aplicacions que s'executen al client, les aplicacions que s'executen al servidor, la plana web del projecte, els directoris on s'emmagatzemen les dades d'entrada i sortida de les tasques i directoris amb els *logs* de les diferents aplicacions del projecte i plantilles per a la creació d'unitats de treball i resultats.

En el directori arrel del projecte també s'hi emmagatzemen els fitxers de configuració, que bàsicament són dos. El fitxer *project.xml*, on s'emmagatzema la informació sobre les plataformes en les que les aplicacions client del projecte estan disponibles i la informació de les pròpies aplicacions. L'altre gran fitxer de configuració s'anomena *config.xml*, i conté informació específica sobre la configuració del projecte, com ara el nom del projecte, la URL d'accés, les aplicacions accessibles pel client o els diferents directoris de treball. També inclou els dimonis que es criden a l'iniciar el projecte, informació sobre el màxim d'unitats de treball a enviar, etc.

Els cinc dimonis principals de tot projecte són els següents[18]:

- el transicionador (*transitioner*),
- el verificador (*validator*),
- l'assimilador (*assimilator*),
- l'eliminador de fitxers (*file deleter*),
- l'alimentador (*feeder*)

També hi ha dues aplicacions més: el planificador i gestor de pujades. (*scheduler* i *file upload handler* en terminologia BOINC). La principal diferència entre els dimonis i aquestes dues aplicacions és que, mentre que les últimes s'executen sempre que un client es connecta al servidor, els dimonis s'executen periòdicament amb una periodicitat indicada en el fitxer *config.xml*. En el temps en que no estan "activats", els processos estan "adormits"[19].

[18] Entre parèntesi s'hi inclou el terme anglès que s'usa en la terminologia BOINC.
[19] Tècnicament es tradueix amb una crida al sistema operatiu anomenada "sleep", per forçar que el procés passi a estat suspès fins que transcorri el temps entre activacions.

En la Figura 8 es pot veure com s'estructura el servidor o servidors de tot projecte basat en BOINC. Tot i que es mostrin com diverses màquines, es poden tenir totes les aplicacions en una mateixa màquina. Entre els anomenats *BOINC components* es troben el planificador, el gestor de pujades i els dimonis transicionador, eliminador de fitxers i alimentador. Entre els *project-specific components* s'hi troben el verificador i l'assimilador, que estaran lligats, a més a més, a cada aplicació client del projecte. Aquestes també entren en aquesta categoria, tot i trobar-les també al costat del client, ja que es considera que estan desenvolupades pels creadors del projecte i són distribuïdes des del servidor. La interfície o interfícies web són accedides a través del servidor web Apache, també necessari per executar les peticions dels clients al planificador i al gestor de pujades. Donat que les peticions a aquestes aplicacions es fan a través d'HTTP, es necessita el servidor Apache per a que executi aquestes aplicacions CGI.

4.3. Unitats de treball i resultats

Abans de veure els dimonis dels que es disposen i el funcionament precís de la infraestructura cal explicar detalladament que són les unitats de treball i els resultats (*workunits* i *results* respectivament en terminologia de la BOINC[20]). Com es veurà, és fonamental entendre aquests dos conceptes, ja que són els pilars en tot projecte basat en aquesta infraestructura.

En la secció 4.1 on s'ha introduït la BOINC, s'ha apuntat que entre les característiques de les aplicacions, és necessari que aquestes siguin altament paral·lelitzables. Al dividir una gran tasca (en aquest PFC, la prova de 2^{72} claus de xifrat) es creen paquets de tasques sense dependència de dades entre elles, o amb una dependència mínima i controlable. Aquestes tasques s'anomenen unitats de treball.

Donat que els problemes que aborda el càlcul distribuït poden necessitar de moltes tasques i/o moltes dades, es poden arribar a necessitar moltes unitats de treball i els seus respectius resultats. Aquests dos conceptes tenen la seva pròpia taula a la base de dades, i solen necessitar emmagatzemar molta informació, pel que poden ocupar molt d'espai.

[20] Com amb el cas dels dimonis, es prefereix utilitzar la seva traducció.

Unitats de treball

Una unitat de treball descriu un càlcul a realitzar [20]. El conjunt de les unitats de treball formen una divisió en tasques el màxim d'independents, per assolir l'objectiu o objectius del projecte. Les dependències habituals entre tasques poden venir donades, per exemple, pel fet que les dades calculades en una tasca siguin dades d'entrada en una o diverses tasques posteriors.

Per cada unitat de treball es poden crear diversos resultats, on el mínim de resultats a crear serà el quòrum desitjat. Cada unitat de treball té dos estats, segons si ha estat assimilada i si s'han esborrat els fitxers d'entrada i sortida. També es guarda, a la base de dades, a quina aplicació pertany (només pot ser una), si necessita validació i el nombre identificatiu del seu resultat canònic (en cas que existeixi).

Una altra informació guardada també és el nombre màxim de resultats que es volen per aquella unitat de treball, i el nombre màxim de resultats vàlids i erronis. Per últim, existeixen camps on s'especificaran les estimacions d'operacions en coma flotant que poden arribar a ser necessàries per realitzar el càlcul, i el màxim d'operacions permès. També s'especifiquen els límits de memòria i de disc a utilitzar.

Les unitats de treball es representen amb un registre a la taula de *workunits* de la base de dades i tenen atributs estàtics, d'estimacions de recursos i de redundància i planificació. Els atributs estàtics d'una unitat de treball són els següents:

- **name:** És el nom, representat amb una cadena de caràcters, ha de ser únic.
- **application:** L'aplicació que realitza el càlcul. No es detalla la versió o rang de versions. Si les dades d'entrada de la unitat de treball tenen formats diferents en diferents versions, s'han de crear aplicacions diferents.
- **input files:** El llistat de fitxers d'entrada, el nom i el nom amb que l'aplicació s'hi referirà.
- **priority:** La prioritat de la unitat de treball. Un valor major implica tenir més prioritat.
- **batch:** Un enter usat per cancel·lar, canviar la prioritat, etc, en grups d'unitats de treball.

Els atributs sobre estimacions dels recursos i límits són:

- **rsc_fpops_est:** L'estimació de la mitjana del nombre d'operacions en coma flotant necessàries per realitzar el càlcul. S'usa per calcular quan poden tardar els càlculs en un host.
- **rsc_fpops_bound:** El límit d'operacions en coma flotant necessàries per completar el càlcul. En cas d'excedir-se aquest límit, l'aplicació s'avorta.
- **rsc_memory_bound:** Una estimació de la memòria necessària per executar l'aplicació. Només s'envien unitats de treball a clients que cedeixin suficient memòria.
- **rsc_disk_bound:** El límit en espai de disc utilitzat per l'aplicació, incloent els fitxers d'entrada, sortida i temporals.

Els atributs de redundància i planificació són:

- **delay_bound:** El límit superior de temps mesurat en segons entre que s'envia un resultat a un client i aquest contesta. Si un client no contesta entre els límits de temps, el servidor descarta el resultat i en genera un altre que assigna a un altre client.
- **min_quorum:** El nombre mínim de consens. El verificador s'activa quan s'assoleix aquest nombre. S'accepta el resultat que la majoria hagi generat. Aquest valor és superior o igual a dos si es vol redundància en el càlcul.
- **target_nresults:** Quants resultats crear inicialment. Com a mínim n'hi ha d'haver *min_quorum*.
- **max_error_results:** Si el nombre d'errors de clients excedeix aquesta xifra, es declara que la unitat de treball té un error.
- **max_total_results:** Si el nombre total de resultats excedeix aquesta xifra, també és declara que la unitat de treball té un error.
- **max_success_results:** Si el nombre de resultats correctes excedeix aquesta xifra, significa que no s'ha arribat al consens i es declara que la unitat de treball té un error.

Les unitats de treball tenen un atribut dinàmic, que és:

- **error mask:** Una màscara de bits amb diverses condicions d'error:
  - **WU_ERROR_COULDNT_SEND_RESULT:** El planificador ha estat

incapaç d'enviar la unitat de treball a 100 o més clients. Si es dóna, BOINC descarta aquesta unitat de treball

- **WU_ERROR_TOO_MANY_SUCCESS_RESULTS:** S'han retornat masses resultats correctes sense arribar al consens.
- **WU_ERROR_TOO_MANY_TOTAL_RESULTS:** S'han rebut masses resultats per la unitat de treball.

Resultats

En el context de la BOINC, un resultat s'entén com una descripció d'una instància d'un càlcul, ja sigui sense començar, en progrés o completat [21]. El càlcul és la unitat de treball associada a un resultat. Com és lògic, un resultat només pot estar associat a una única unitat de treball, mai a diverses. En canvi, una unitat de treball, pot tenir associats diversos resultats, segons si es vol redundància en el càlcul.

Mentre que una unitat de treball té un únic fitxer o fitxers d'entrada per tots els seus resultats, cada resultat té el seu o seus propis fitxers de sortida. Com es veurà, encara que dos resultats d'una mateixa unitat de treball sembla que hagin de ser exactament igual, pot no ser així, ja que es poden donar discrepàncies numèriques. Una de les opcions del projecte, que no s'ha comentat fins al moment, és la possibilitat d'enviar a un mateix host resultats d'una mateixa unitat de treball. Tot i que sembli il·lògic, es podria fer, afectant al quòrum d'aquella unitat de treball, i al referent a les discrepàncies numèriques.

A la base de dades dels resultats s'hi emmagatzema informació relativa a la unitat de treball a la qual pertany, a l'estat del resultat en el servidor i en el client i el de validació. Respecte a la validació, pot passar que un resultat es validi com a correcte o que el format del fitxer o fitxers de sortida sigui incorrecte i per tant no es validi com correcte. També es guarda el límit de temps que té el client per informar sobre el resultat, la versió de l'aplicació que està utilitzant el client per computar-lo, així com el crèdit que s'ha reclamat des del client per aquell resultat i el crèdit que el verificador ha concedit.

Cada resultat està descrit amb un registre a la taula de resultats de la base de dades. Els atributs són els següents:

- **name:** El nom, representat amb una cadena de caràcters, únic entre tots els noms de resultats.
- **workunit name:** Identifica la unitat de treball associada.
- **output files:** La llista de noms dels fitxers de sortida i els noms pels quals les aplicacions s'hi referiran.

Un resultat també té un atribut dinàmic:

- **server state:** els valors poden ser inactiu, no enviat, en progrés, acabat, *timed out*, fet amb error o no necessari.

Un cop el resultat està completat s'omplen els següents atributs:

- **host:** El host que ha realitzat el còmput.
- **exit status:** L'estat (0 si finalitzat amb èxit).
- **CPU time:** El temps de CPU utilitzat.
- **output file info:** Les mides i els *checksums* dels fitxers de sortida.
- **stderr:** La sortida *stderr* del càlcul.
- **received time:** La data de recepció del resultat.

Redundància i errors

El servidor BOINC envia els resultats com una abstracció d'un càlcul al client i aquest respon amb el resultat després de realitzar el càlcul. Es poden donar diverses situacions [22]:

- El client calcula el resultat correctament o incorrecta i el retorna.
- El client pot fallar en l'intent de descarregar o enviar els fitxers necessaris.
- L'aplicació pot generar errors en el client.
- El client no retorna cap resultat perquè s'espatlla la màquina de l'usuari o bé aquest deixa d'utilitzar el client de la BOINC.
- El planificador és incapaç d'enviar la unitat de treball degut a que necessita més recursos dels que qualsevol dels voluntaris disponibles disposa.

A més a més, per assegurar la veracitat dels resultats, es pot enviar una mateixa unitat de treball a diversos clients. Els resultats obtinguts seran comparats i s'accepta el resultat que els consens indiqui com a correcte. Aquest procés és automàtic, però es necessita la intervenció humana en dos casos. El desenvolupador ha de definir com es fa la validació dels resultats, decidint el nombre de resultats correctes mitjançant el consens. Quan s'aconsegueix el consens, es determina el resultat com a "canònic". Qualsevol resultat que arribi després d'aconseguir el consens és comparat amb aquest.

La segona tasca que necessita ser definida pel desenvolupador és l'assimilació. Mitjançant aquest procés s'informa al projecte que una determinada unitat de treball s'ha completat. En cas que el resultat sigui correcte, la funció proporcionada per processar els resultats s'encarrega de tractar-los. En cas contrari, aquesta funcionalitat informa de l'error.

4.4. Emmagatzematge de dades

El model d'emmagatzemament de la BOINC està basat en fitxers, que poden ser fitxers d'entrada o sortida, components de l'aplicació, com els executables, etc. Els fitxers emmagatzemats al servidor de dades es transfereixen al client per HTTP. Un cop s'ha creat un fitxer, aquest resta immodificable, i si es canvia el contingut del fitxer, s'ha de guardar com un fitxer diferent, amb nom diferent.

Les propietats d'un fitxer s'especifiquen en format *xml*, guardant diversos camps, com el nom, la mida màxima o el resum MD5, utilitzat, per exemple, al client per validar que durant la transferència no ha estat modificat. També es guarden en *xml* les referències al fitxer, com per exemple el nom amb els que els programes l'invoquen. Quan un fitxer no es necessita més, la BOINC l'elimina. Per exemple, els fitxers d'entrada d'una unitat de treball s'eliminen quan cap unitat de treball fa referència a ells.

La base de dades

BOINC utilitza una base de dades MySQL que emmagatzema tota la informació rellevant del projecte, incloent informació dels usuaris registrats i els servidors associats, sobre les aplicacions i les seves versions, dels *core clients*, i de les unitats de treball i els resultats associats a cadascuna.

Més específicament, la base de dades consta de vint-i-dues taules, tot i que el creador d'un projecte pot afegir-hi les que consideri necessàries pel seu projecte. Entre elles hi ha les que emmagatzemen la informació de les aplicacions, definides al fitxer *project.xml*, i una altra taula per les diverses versions de cada aplicació. Referent també a les aplicacions, existeix una altra taula amb informació sobre les plataformes sobre les quals poden funcionar les aplicacions.

Sobre els participants, es guarda la informació més rellevant en una taula, i en una altra s'hi guarda informació sobre els grups de participants. L'últim grup de taules més important són les que emmagatzemen tota la informació sobre les unitats de treball i els seus resultats. Com s'ha vist a l'apartat anterior, aquestes són les taules que més espai ocupen dintre la base de dades, i les que requeriran ser netejades periòdicament per evitar ocupar més de l'espai disponible a disc.

Com s'ha comentat, la base de dades és el component central i més delicat del projecte, ja que sense aquesta, cap projecte podria funcionar. És de vital importància que aquesta estigui protegida contra atacs, i que només sigui accessible pels dimonis del projecte. La capacitat d'emmagatzematge és un factor molt important, per tant, s'ha de netejar periòdicament la base de dades per no desbordar la capacitat de la màquina on s'allotja.

És d'obligat comentari en aquest punt dir que, tot i que en la Figura 8 es mostra com si el planificador és connectés directament a la base de dades, aquest fet és totalment erroni en l'obtenció de resultats per enviar al client. Com ja s'ha apuntat, el planificador llegeix els resultats disponibles d'un espai de memòria compartida creat per l'alimentador. Quan sí que es connectarà a la base de dades és al rebre el resultat calculat per part del client.

4.5. Dimonis i aplicacions

El transicionador

S'encarrega de gestionar l'estat de les transicions de les unitats de treball i dels resultats. Comprova aquest estat a la base de dades i actualitza els camps apropiats de la base de dades quan la unitat de treball està preparada per la transició d'estat. Aquest procés és complicat donat que una unitat de treball té un conjunt de subestats i no un estat total. Si una unitat de treball està llesta per validar-se i hi ha suficients resultats per garantir el consens, aleshores es canvia a l'estat anomenat "llest per validar-se". El transicionador és un dels programes que necessita més ús de processador, però es pot dividir en diversos processos distribuïts en diversos servidors, cadascun ocupant-se d'un subconjunt d'unitats de treball.

El verificador

S'encarrega de verificar si els resultats obtinguts pels clients són vàlids, comprovant a la base de dades si hi ha nous resultats i aplicant la funció específica per a la comparació de resultats. Aquesta funció ha d'extreure un resultat canònic del conjunt de resultats disponible.

És obligat parlar en aquest punt sobre les discrepàncies numèriques. Depenent de l'aplicació que es programi, en diferents arquitectures, sistemes operatius i fins i tot en diferents compiladors ens podem trobar que es generen resultats diferents. El grau de diferència dels resultats pot variar bastant, per això, alhora de verificar-los s'ha de tenir en compte per fer una comparació difusa que permeti desviacions amb un percentatge en el resultat. També es pot trobar que els resultats siguin completament diferents, ja que la desviació ens els càlculs parcials s'acumuli i per tant es generin resultats molt diferents[21]. En aquest cas la comparació de resultats és molt més complicada, i la comparació difusa ja no sol servir.

Per intentar pal·liar aquest problema, la BOINC proporciona una característica

[21] Aquests casos se solen donar en càlculs en coma flotant, on la precisió de les operacions i la longitud en bits dels operands pot variar entre arquitectures.

anomenada redundància homogènia. Amb aquest mecanisme activat el planificador només enviarà una unitat de treball a clients amb la mateixa plataforma.

L' assimilador

S'encarrega de comprovar regularment si hi ha unitats de treball llestes per l'assimilació. L'administrador ha de proporcionar una funció que determini que s'ha de fer amb els resultats canònics, com per exemple notificar a les persones interessades o postprocessar les dades per emmagatzemar-les. Quan una unitat de treball és assimilada, es marcarà com a completada.

L'eliminador de fitxers

Comprova les unitats de treball completades que s'han acumulat des de l'últim cop que s'ha llençat aquest procés. Si n'hi ha, neteja els fitxers d'entrada i sortida del servidor web relacionats amb les unitats de treball completades. Només s'eliminen fitxers, mai registres de la base de dades, d'aquesta manera sempre està disponible la informació necessària de les unitats de treball.

L'alimentador

Carrega els resultats no enviats de la base de dades a un segment de memòria compartida per tal d'augmentar el rendiment del sistema limitant el nombre de peticions a la base de dades. Especialment es millora el rendiment del planificador, ja que només ha d'accedir al segment de memòria compartida per enviar els resultats als clients. Els resultats es carreguen en ordre de creació de la unitat de treball.

Sobre la memòria compartida, s'ha de dir que té una mida limitada, i configurable a través del fitxer *config.xml*, i que a mesura que es va buidant, cada vegada que un client agafa un resultat, l'alimentador la va reomplint per mantenir sempre el màxim de resultats disponible.

El planificador

És un programa CGI que s'executa cada cop que un client es connecta al projecte per demanar tasques a realitzar. Com ja s'ha comentat, no fa peticions a la base de dades alhora d'enviar resultats a clients, sinó que accedeix a l'espai de memòria compartida. Els planificador envia resultats segons les preferències del client, sempre respectant els recursos màxims utilitzables especificats pel client. Durant la sessió de planificació, el client rep la llista de resultats assignats per processar i una llista d'URLs d'on descarregar-se els fitxers necessaris per executar les tasques. En canvi, quan un client es connecta per notificar que s'ha acabat de processar un resultat, aleshores el planificador sí que accedirà a la base de dades, per indicar que el resultat ha estat completat.

El gestor de pujades

Quan un client ha finalitzat el càlcul d'una tasca i ha informat al planificador, llavors ha d'enviar el fitxer o fitxers de resultats al servidor. En aquest punt entra aquesta aplicació, que calcularà un *hash* sobre el nom de la unitat de treball per decidir a quin subdirectori s'emmagatzemen els resultats. Aquesta aplicació, igual com el planificador, també és un programa CGI.

A més a més d'aquests components fonamentals del servidor, existeixen tot un seguit d'utilitats per facilitar la gestió a l'administrador del projecte. Entre les utilitats més importants hi ha un seguit d'aplicacions per actualitzar la base de dades en la part referent a les aplicacions i les seves versions. Hi ha utilitats per la generació de tasques, per l'eliminació de tasques antigues a la base de dades, i per posar en marxa, veure l'estat i parar el projecte.

Referent a la posada en marxa del servidor, el que es fa es invocar els diferents dimonis i crear l'espai de memòria compartida. Per saber l'estat del projecte, al posar en funcionament el servidor, es creen uns fitxers de *lock* i de *pid*[22], per indicar quins dimonis estan en aquell moment en funcionament. En els últims d'aquests fitxers, també s'hi indica la ID del procés per tenir-los controlats. Alhora de parar el servidor

[22] Process ID.

s'utilitzen aquests fitxers de *pid* per saber a quins processos se'ls hi ha d'enviar la senyal de parada.

Aplicacions i versions

Una aplicació consta d'un programa, que pot tenir versions per diferents plataformes, i un conjunt d'unitats de treball i resultats. Un projecte pot tenir diverses aplicacions, aquestes són guardades a la taula *application* de la base de dades i poden ser creades mitjançant la utilitat anomenada *xadd*.

Una aplicació pot tenir diverses versions, i una versió d'una aplicació és una versió concreta compilada per una plataforma concreta. Una versió consisteix en diversos fitxers, a part del programa principal també pot tenir *scripts*, programes de preprocessament o postprocessament, etc. El nombre de versió és un enter, i tot i que aquest nombre pot variar segons la plataforma, es recomana mantenir la consistència i que un nombre concret de versió l'aplicació hauria de ser computacionalment idèntica independentment de la plataforma.

Cada aplicació té un nombre mínim de versió, i en cas que un client demani rebre tasques d'una aplicació, se li enviarà, apart de les dades, la versió més recent de l'aplicació sempre que aquesta sigui superior o igual a aquest nombre mínim. Les versions es poden crear utilitzant la utilitat anomenada *update_versions*. La informació de cada versió de l'aplicació es guarda a la taula *app_version* de la base de dades.

4.6. Seguretat en BOINC

Fins ara s'ha parlat dels components principals de la BOINC, però no s'ha fet referència a la seguretat dels clients o del propi servidor. Per seguretat, en aquest cas, s'entén com evitar que els clients executin codi que no sigui el propi dels projectes en que participi. També, a la banda del servidor, s'entendrà que no es puguin produir atacs que permetin a gent aliena al projecte entrar a la màquina servidora per aconseguir modificar, per exemple, les aplicacions del projecte.

En tot cas, aquest apartat està més centrat en la seguretat dels usuaris dels projectes, en l'enviament de les aplicacions i fitxers d'entrada i sortida entre client i servidor. És de vital importància evitar qualsevol modificació del codi de les aplicacions en el trajecte d'aquestes entre el servidor i els clients, ja que la imatge del projecte, i en general, de la PRC quedaria debilitada.

Cal recordar en aquest punt, que tant les comunicacions com les transferències de fitxers entre client i servidor es realitzen mitjançant el protocol HTTP. Aquest protocol no ofereix cap tipus de seguretat com si l'ofereix el seu homòleg segur, anomenat HTTPS. Per pal·liar la manca de seguretat d'HTTP, però sense utilitzar HTTPS, s'implementa un esquema de signatura digital. No és l'objectiu d'aquest projecte veure en detall com s'implementen els esquemes de signatura digital, ja que BOINC proporciona aquesta opció de forma totalment transparent, tant a l'usuari com a l'administrador.

Al crear el projecte, automàticament es generen un parell de claus de xifrat que s'utilitzen per obtenir la signatura digital de totes les aplicacions client. Quan un client es descarrega una aplicació, rep també el fitxer amb la signatura digital d'aquest, i amb la clau pública del projecte, que també té disponible, comprova si l'aplicació és idèntica a la del projecte. Tant si pel trajecte entre servidor i client hi ha errors de transmissió com si s'ha substituït el codi, el client de la BOINC detecta aquesta anomalia i impedeix l'execució de l'aplicació.

Respecte la seguretat de la màquina o màquines servidor, s'ha de comentar que els creadors de BOINC recomanen mantenir aquestes màquines rere *firewalls* degudament configurats. També que s'ha de tenir el sistema operatiu del servidor actualitzat, ja que, per exemple, en GNU/Linux surten actualitzacions de seguretat casi bé a diari.

4.7. El cicle d'una unitat de treball

Un cop es té la infraestructura de la BOINC en funcionament[23], només queda la creació de les tasques i els resultats corresponents per poder-los enviar als voluntaris. El procés

[23] El procés de posada en funcionament es veurà al punt 5.2.

es pot veure en la figura 9 i es detalla a continuació. En aquesta figura, les línies contínues indiquen que hi ha una interacció directa i les discontínues una interacció indirecta, en el sentit que un cop es completi aquell pas, el següent pas es durà a terme en la pròxima activació del dimoni corresponent.

Distribució de tasques

La BOINC distribueix el treball de manera que es mantingui als clients el màxim ocupats, però minimitzant la possibilitat que un client retorni resultats amb errors i la possibilitat de participants maliciosos que intentin obtenir múltiples resultats per la mateixa unitat de treball.

Les restriccions alhora de distribuir les tasques són que una tasca només s'envia a un client si la versió de l'aplicació està disponible per la plataforma requerida i es compleix el nombre mínim de versió. Una tasca no s'enviarà si el client no compleix els requeriments de memòria o disc, si s'especifica que només s'accepta un únic resultat per tasca i client i aquest ha tornat a rebre la mateixa tasca, si un client ha sobrepassat la quota de resultat màxima per dia, etc; i tampoc s'accepta si està activada la redundància homogènia i s'ha rebut un resultat de la tasca per una plataforma diferent [23].

El cicle de vida

Primer, es creen les tasques, ja sigui amb la utilitat que ens proporciona BOINC (d'una en una en aquest cas), o amb alguna utilitat desenvolupada en el propi projecte. En la creació de les unitats de treball també es creen els fitxers d'entrada i els *xml* de la unitat de treball i resultats, que es dipositen al directori corresponent. Alhora, s'insereix a la base de dades els registres corresponents a la taula anomenada "*workunit*". En la següent activació del transicionador, es generen tots els resultats a partir del fitxer generat al pas anterior (i emmagatzemat en un camp de la taula "*workunit*"). Aleshores, l'últim pas abans que els clients puguin començar a descarregar-se resultats per computar, és que l'alimentador (en la següent activació, com al cas anterior) llegeixi la base de dades i ompli l'espai de memòria compartida.

En aquest punt, es tenen creades i disponibles les unitats de treball que els clients es

descarreguen al connectar-se al planificador. Així doncs, quan un client es connecta al planificador, en els diversos passos que té la comunicació, el client li proporciona informació referent a sí mateix. Aquesta informació inclou la plataforma en que aquest s'executa, els resultats de l'execució d'uns benchmarks per determinar els gigaFLOPS del host, i els límits de memòria i disc a utilitzar. El planificador determina si se li pot enviar algun resultat o no. En cas que no es pugui enviar cap resultat, el client suspèn les comunicacions amb el servidor durant un temps determinat. Si aquesta comprovació és correcta, el client es descarrega l'aplicació corresponent, en cas que no l'hagi descarregada anteriorment, i els fitxers d'entrada del resultat. Si no està deshabilitada la comprovació del *checksum* dels fitxers, es realitza aquesta comprovació. Si per algun motiu aquests fitxers estan corromputs, es genera un error i s'informa d'aquest al servidor. El resultat queda, per tant, com erroni. En canvi, si es validen correctament els fitxers (sol ser el cas habitual), el client executa la unitat de treball. Com abans, pot ser que aquesta es calculi amb algun error o que es calculi presumiblement de forma correcta. Entre els errors en el càlcul es pot donar que no s'aconsegueixi obrir els fitxers necessaris (entrada, sortida o de *checkpoint*), que hi hagi un desbordament, o alguna violació de segment. També s'envia l'error ocorregut al servidor.

Si la tasca s'executa correctament, al finalitzar-la, el client informa al planificador i envia al gestor de pujades el fitxer resultant amb els resultats del càlcul. El planificador actualitza la base de dades. Si s'ha arribat al quòrum per aquella tasca, en la pròxima execució del verificador, aquest examina els resultats que no tinguin cap error i valida, segons la funció que s'implementi, els fitxers de sortida. Si aquests són correctes, el resultat passa a tenir estat "validat correctament". En cas contrari, s'invalida el resultat. Si s'executa el verificador sense haver-hi resultat calculats, aquest es torna a "adormir".

Com en els casos anteriors, en la següent activació de l'assimilador, es comprova si alguna de les unitats de treball existents té suficient quòrum. Si és així, la unitat corresponent passa a estat "assimilat". L'assimilador, a més a més, segons la funcionalitat de que se'l doti, té la capacitat d'informar als administradors via correu electrònic, actualitzar taules pròpies del projecte, etc.

Finalment, l'últim pas que es du a terme és l'eliminació dels fitxers, en cas que aquesta sigui permesa. El dimoni eliminador de fitxers s'activa i comprova si la unitat de treball

té l'estat anomenat "llest per eliminar fitxers", que prèviament ha actualitzat l'assimilador.

En el diagrama es veu que tant el verificador com l'assimilador accedeixen als fitxers de sortida, tot i que aquesta relació s'hauria d'indicar com a condicional, ja que depèn de la funcionalitat amb la que se'ls doti.

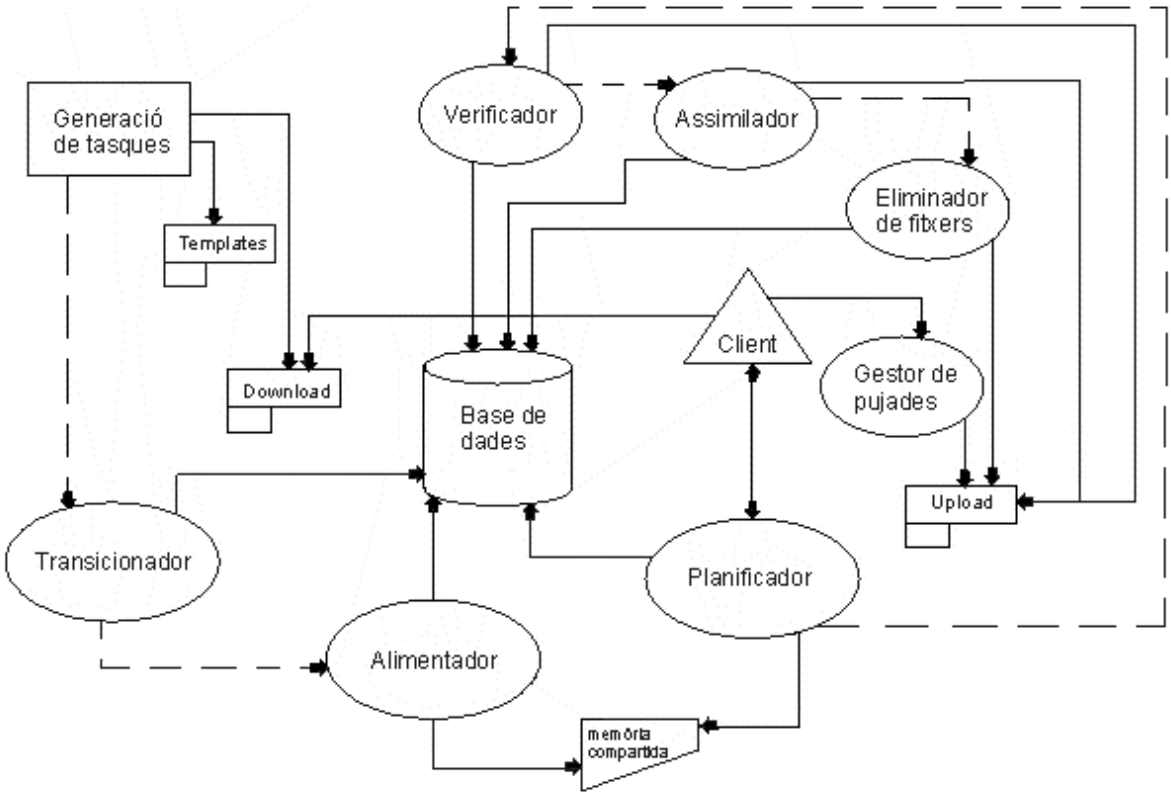

Figura 9: Interacció dels components de la BOINC durant el cicle de vida d'una unitat de treball

5. Desenvolupament de la infraestructura pel trencament de l'RC5-32/12/9

El desenvolupament dut a terme per habilitar la infraestructura pel trencament de l'RC5 ha comportat dues etapes. La primera etapa compren la implementació de l'RC5 (o bé l'adaptació d'una implementació existent) a les necessitats del projecte. En un principi, quan es va planificar l'enviament de les tasques als ordinadors client, es dubtava sobre si enviar 2^{30} ó 2^{32} claus per a que aquest les provés. El dubte radicava en la durada de les tasques, ja que entre una decisió i l'altra els temps, com es veurà, s'arribaven a multiplicar per quatre. Per tant, es va desenvolupar un programa per mesurar el temps en mitjana que un ordinador estàndard de sobretaula necessita per provar aquest nombre de claus. També, evidentment, aquesta tasca va permetre validar que la implementació de l'RC5 obtinguda és correcta, tot utilitzant el pseudo test que RSA Laboratories posa a disposició dels participants.

En la segona etapa, la més llarga del desenvolupament, s'ha habilitat el servidor BOINC. Com es veurà en el següent apartat, per dur a terme aquesta tasca s'han hagut de desenvolupar els dimonis necessaris i s'ha hagut d'adaptar la implementació de l'RC5 de l'etapa anterior per a que utilitzi l'API de la BOINC per interaccionar amb el client. A més a més, també ha estat necessari modificar la base de dades base que la BOINC proveeix tot afegint-hi taules noves per guardar-hi informació específica del projecte.

5.1. Aplicacions desenvolupades

La implementació de l'algorisme de xifrat RC5 triada és una adaptació de la que els propis creadors de l'algorisme varen adjuntar amb el document on exposaven l'algorisme [3]. La seva implementació era d'un RC5-32/12/16, escrita en llenguatge C. En aquest projecte, a més d'adaptar-la a una implementació amb clau de 72 bits, en comptes dels 128 bits de la versió proposada, també s'han canviat parts del codi per fer que la implementació final fos el més eficient possible, intentant evitar operacions innecessàries i intentant reduir, en la mesura del possible, el temps de còmput.

L'aplicació final només desxifra dues paraules[24] del text xifrat proposat al desafiament. Amb dues paraules n'hi ha suficient, tot i que de fet només es coneix el text pla per les primeres quatre paraules, corresponents al començament de la frase "*The unknown message is:* ". La part de l'algorisme corresponent al xifrat no ha estat necessari utilitzar-la.

En aquesta primera etapa del desenvolupament, es van programar les següents aplicacions:

- *rc5ValidaTest*
- *rc5TestClaus*
- *rc5BOINC*

Aquestes aplicacions han estat desenvolupades per executar-se sobre els sistemes operatius Windows i GNU/Linux. En el cas de GNU/Linux s'ha adaptat, a més a més, per utilitzar totes les CPUs de la màquina, ja que els fabricants de plaques mare i CPU estan traient al mercat models amb múltiples processadors, com els CoreDuo d'Intel i els AM2 d'AMD. Mitjançant una crida al sistema de la llibreria *unistd.h* s'esbrina el nombre de CPUs i, a partir d'aquesta informació, es creen els *threads* necessaris i es reparteix el rang de claus disponible equitativament a cada CPU.

La primera aplicació, *rc5ValidaTest*, serveix per comprovar el correcte funcionament de l'algorisme, sobre el pseudo test corresponent. Aquesta és l'única d'aquestes tres aplicacions que treballa sobre el text complert.

La segona de les aplicacions, *rc5TestClaus*, realitza un test de rendiment de la màquina que l'executa. Aquesta és l'aplicació utilitzada per decidir quantes claus es proven per unitat de treball. Els testos de rendiment serveixen per mesurar el temps necessari per calcular-se un rang de claus de 2^5, 2^{10}, 2^{15}, 2^{20}, 2^{25}, 2^{30} i 2^{32} en segons. A l'apartat 6 i a l'annex D es veuen exemples d'aquests temps a partir de 2^{20} claus[25].

Per últim, l'aplicació *rc5BOINC* tenia la finalitat de simular una unitat de treball en BOINC, agafant el rang de claus des de fitxer.

[24] Cal recordar que RC5 és un xifrador en bloc que necessita de dues paraules per realitzar el xifrat.
[25] Per rangs de claus inferiors a aquesta mida els temps són despreciables.

Aplicacions pròpies de la infraestructura

Les aplicacions que es descriuen a continuació corresponen a la segona etapa del desenvolupament i fan referència als dimonis i CGIs que la BOINC necessita per gestionar les dades de l'aplicació. Tal i com s'ha descrit en l'apartat 4, la BOINC proporciona tres dimonis que són comuns a tots els projectes, tot i que òbviament cada projecte se'ls pot adaptar a les seves necessitats. Aquests dimonis són l'alimentador, que crea un espai de memòria compartida amb les tasques pendents que hi ha a la taula de resultats de la base de dades; el transicionador, que gestiona les transicions de les diferents unitats de treball i el seus resultats; i l'eliminador, que s'encarrega d'eliminar els fitxers d'entrada i sortida que ja no són necessaris. Finalment, també hi ha dues aplicacions CGI, que són el planificador, que és l'aplicació a la qual els clients es connecten per demanar tasques i el gestor de pujades, al qual els clients, un cop acabada la unitat de treball, s'hi connecten per enviar el fitxer de resultats.

A més dels tres dimonis descrits anteriorment, n'hi ha dos més anomenats verificador i assimilador que són dependents de l'aplicació. Aquests dos dimonis s'encarreguen de validar que els resultats d'una unitat de treball generats per un participant del projecte siguin vàlids, per exemple validant que la sintaxis del fitxer sigui la correcta. Un cop els resultats han estat validats, llavors entra en joc l'assimilador, que s'encarrega d'assimilar el resultats, és a dir, processar-los per incorporar-los a la base de dades, generar més unitats de treball, informar a qui pertoqui, etc. Donat que el procés dels resultats depèn de l'aplicació que els genera, s'ha de dissenyar i programar aquest parell de dimonis per cada aplicació diferent que es tingui.

A més a més, a banda d'aquests dimonis, també es pot tenir un generador de tasques, que és l'aplicació que s'encarrega de generar unitats de treball automàticament, emulant a l'aplicació que dóna la BOINC per crear tasques manualment. Aquesta aplicació es pot considerar un dimoni o no depenent de com es concebi. Per exemple, si cada cop que s'assimilen un conjunt de resultats d'una unitat de treball des de l'assimilador es crida al generador de tasques per a que generi més tasques, no es podria considerar un dimoni. En canvi, si s'afegeix al fitxer de configuració del projecte per a que s'executi periòdicament per comprovar el nombre d'unitats de treball a la base de dades i generar-

ne les necessàries per mantenir ocupats als participants del projecte, llavors sí que el es pot considerar com un dimoni.

Donat que aquest projecte només treballa amb una aplicació, que és l'encarregada de provar un conjunt de 2^{32} claus, només cal desenvolupar un únic verificador i un únic assimilador. Pel que fa al generador de tasques, es pot desenvolupar com un programa en C/C++ o com un *script* de Linux que invoca a l'aplicació corresponent de la BOINC. Per desenvolupar aquests dimonis, així com per desenvolupar les aplicacions del projecte, la BOINC proporciona unes APIs amb tot el conjunt de funcions necessàries per accedir a la base de dades, entre altres. Tot i estar tot ordenat en directoris segons la funcionalitat, alhora d'incloure les diferents llibreries no cal ficar el directori relatiu per cadascuna, sinó que el fitxer de *Makefile* s'encarrega de resoldre totes les dependències amb les llibreries pròpies de l'API quan s'executa la comanda *make* per compilar les aplicacions i dimonis del projecte. Per no haver de modificar cap fitxer de *Makefile* BOINC proporciona uns fitxers de contenidor (*placeholder*) per posar la funcionalitat d'aquests dimonis i compilar sense necessitar modificar cap *Makefile*.

A banda dels dimonis i el generador de les unitats de treball que s'han desenvolupat pel projecte, també s'han afegit a la base de dades dues taules per condensar la informació necessària pel desenvolupament del projecte.

Taules noves a la base de dades

Les dues noves taules creades s'anomenen *tested_keys* i *project_status*. Per cadascuna de les taules s'han afegit les estructures i classes en C++ necessàries a les APIs de la BOINC, en aquest cas, declarades al fitxer *db/boinc_db.h* i implementades al fitxer *db/boinc_db.C*.

La primera de les taules, *tested_keys*, serveix per reduir la mida de la base de dades. Sense aquesta nova taula, es necessitaria un registre de la taula *result* i un de la taula *workunit* per emmagatzemar la informació de cada tasca[26]. Ara bé, donades les característiques del projecte, un cop una tasca sigui calculada i es tingui el seu resultat,

[26] Donat que no s'inclou redundància en el càlcul de les unitats de treball, ja que es considera suficient el sistema de validació de resultats i introduir-la faria baixar la potència de càlcul del projecte a la meitat o més.

la única informació que es necessita emmagatzemar és que aquell rang de 2^{32} claus de la tasca ha estat calculat, i si conté la clau buscada. Per tant, es pot esborrar el resultat i la unitat de treball i emmagatzemar, en aquesta nova taula, que aquest rang de claus ja ha estat provat. Amb això, s'obté un gran estalvi d'espai a la base de dades, ja que cada registre de les taules *workunit* i *result* emmagatzema molta informació innecessària un cop computada la tasca.

La descripció que s'acaba de fer genera un total de 2^{40} registres en aquesta nova taula. Ara bé, es pot reduir l'espai necessari si es generen conjunts de tasques[27] de, per exemple, 1024 unitats de treball. Quan es creen aquestes tasques, s'insereix un registre a la taula *tested_keys* que conté la informació necessària sobre aquestes 1024 unitats de treball. Cada cop que una d'aquestes unitats de treball s'ha computat correctament, s'actualitza el registre corresponent per indicar-ho. Un cop el conjunt complert de tasques han estat computades, s'esborren tots els registres de la taula *result* i *workunit* de la base de dades i s'actualitza el registre a la taula *tested_keys* creat prèviament per indicar la finalització d'aquest conjunt de tasques.

La taula creada per a poder realitzar les reduccions d'espai que s'acaba de descriure conté els següents camps:

- **Id:** La clau primària.
- **Ini_rang1:** Els 4 bytes inferiors del rang.
- **Ini_rang2:** El byte superior del rang[28].
- **Long_rang:** La longitud del rang, expressada en nombre de tasques.
- **Status:** L'estat del paquet. Pren els valors '*O*' i '*F*', que signifiquen "*ongoing"* i "*finished"* respectivament.

La creació de cada registre de la taula la fa la utilitat *rc5_make_work* (que es veurà més endavant). L'actualització de la taula la realitzarà l'assimilador, l'explicació del qual es veurà quan s'expliqui la funcionalitat desenvolupada en aquest dimoni.

La segona de les taules, anomenada *project_status*, s'utilitza per guardar l'estat del projecte i consultar-lo via web. En aquest cas, l'objectiu és oferir als participants del

[27] Veure explicació de la utilitat *rc5_make_work*.
[28] Si la clan és de 72 bits, els quatre bytes inferiors no formen part del rang, per tant en queden 5 bytes.

projecte una pàgina web on puguin veure el seu estat. La informació que es pot veure inclou el nombre de claus que s'han provat, el tant per cent al que s'ha arribat i, en cas que s'hagi assolit l'objectiu, la clau resultant, el text complert desxifrat i l'usuari que ha processat la unitat de treball corresponent.

La taula té els següents camps:

- **Id:** La clau primària.
- **Data:** El dia en curs.
- **Key_exists:** L'indicador de si s'ha trobat la clau del criptosistema.
- **Key_value1, 2 i 3:** La clau del criptosistema (en cas que s'hagi trobat).
- **Num_computed_wu:** El nombre d'unitats de treball computades fins al moment.
- **Deciphered_text:** El text complet desxifrat (en cas que s'hagi trobat la clau).
- **User_found_key:** L'*id* de l'usuari que ha calculat la unitat de treball (en cas que s'hagi trobat la clau). El valor de l'*id* es troba a la taula d'usuaris.

Inicialment, al crear la taula s'insereix un registre inicial amb la data d'inici del projecte, el nombre d'unitats de treball computades a zero i la resta de camps també a zero. Aquest registre inicial és necessari ja que, com es veurà, l'assimilador actualitza aquesta taula utilitzant l'últim registre existent.

Com en el cas de la taula *tested_keys*, l'actualització la realitza l'assimilador, tal i com es veurà més endavant. Per visualitzar l'estat del projecte, a la pàgina web d'aquest es pot trobar l'enllaç a l'apartat *Stadistics*/*Stadistics*. Es mostra com una taula on cada registre és un dia. Per tant, les dades mostrades corresponen al progrés diari del projecte. A l'apartat 6.4 i a l'annex E es pot veure un resum de les estadístiques fins a dia de tancament d'aquesta memòria.

Un cop vistes les modificacions de la base de dades del projecte, es passa a descriure el funcionament dels dos dimonis específics del projecte, el verificador i l'assimilador.

El verificador

En aquest projecte, el verificador només comprova que els resultats llegits del fitxer de sortida[29] segueixin la sintaxi esperada (dit d'altra manera, estiguin realment generats per l'aplicació corresponent). La BOINC proporciona per aquest dimoni un conjunt de fitxers: *validator.C validate_util.C, validate_util2.C i validator_placeholder.C.*

El primer conté el *main* del verificador i fa unes comprovacions bàsiques sobre la crida d'aquest, i comprova també que les unitats de treball i resultats associats existeixin i el seu fitxer de sortida estigui present a disc. Els fitxers *validate_util.C* i *validate_util2.C* contenen utilitats per obrir els fitxers de sortida de cada aplicació o calcular el crèdit a concedir per cada resultat. També contenen funcions per tal que la funcionalitat a dissenyar sigui executada pel conjunt de resultats de cada unitat de treball, per assumir un resultat canònic i validar els resultats posteriors contra el resultat canònic.

Finalment, el fitxer de "*placeholder"* anomenat *validator_placeholder.C* incorpora el codi amb la funcionalitat de validació. Abans de descriure la funcionalitat de validació, cal recordar que en aquest projecte cada unitat de treball només té un únic resultat associat i, donat que les funcions proporcionades són per comprovar dos resultats o un conjunt de resultats, s'ha hagut de modificar el codi per a que funcioni amb un únic resultat. La funcionalitat de validació simplement examina si el fitxer de sortida segueix la sintaxi correcta, en cas contrari es genera l'error corresponent i es descarta.

La sintaxi del fitxer de sortida pot ser una de les dues següents, depenent de si s'ha trobat la clau o no[30]:

Keys test *FEE1D4C0 85* Time: *6500.000000* Result: *1* Win Key: *53030CC9 FEE1D4C0 85*	Keys test *CE15D4F7 52* Time: *6500.000000* Result: *0*

[29] Pel cas d'aquest projecte cada *result* només genera un únic fitxer de sortida
[30] Les dades mostrades corresponen a claus provades amb el pseudo test.

Per fer la validació, al fitxer de "*placeholder*" s'han de realitzar tres funcions: inicialitzar, netejar un resultat i validar-lo. El verificador està pensat per validar dos resultats per comparació d'aquests, en canvi en aquest cas es valida cada resultat un a un, comprovant si el fitxer indica que s'ha trobat la clau que trenca el criptosistema o no. En cas d'haver-se trencat, verifica que la clau té el format correcte. A més a més, també es comprova que el temps de càlcul sigui superior a zero, indicant un error en cas contrari. Addicionalment, en cas de trobar la clau, s'ha afegit la funció de provar-la, intentant desxifrar les dues paraules inicials del text i comparant-les amb les esperades. En el supòsit que la clau trobada no fos vàlida, es generaria un error indicant aquesta anomalia i es descartaria el resultat.

Tot i que el projecte no contempla cap comparació de resultats dos a dos, si es volgués tenir redundància en el càlcul, un cop es tingués un resultat canònic, per validar els següents només caldria comparar el contingut dels dos fitxers. Per aquest motiu s'ha mantingut la funció de validació dos a dos, i s'ha afegit una de nova per validar el fitxer de sortida d'un sol resultat.

L'assimilador

La BOINC també proporciona un conjunt de fitxers per implementar l'assimilació com en el cas del verificador, però més reduït. Com en el cas anterior, s'ha d'ubicar tota la funcionalitat específica en el fitxer de "*placeholder*" (anomenat *assimilator_placeholder.C*). També es disposa d'un fitxer amb la funcionalitat bàsica que, en aquest cas, no cal modificar.

Alhora d'assimilar un resultat sempre es queda amb el canònic, en cas que n'hi hagués més d'un. Per tant, primer es comprova si existeix i s'obre el fitxer de sortida corresponent (sempre reportant un error si no existeix), i se'n llegeix el contingut. En aquest punt, segons el que es llegeixi d'aquest fitxer poden donar-se dues situacions. Evidentment, en aquest punt no cal comprovar la sintaxi, ja que ja ha d'haver estat comprovada pel verificador[31]. La primera de les situacions, i la més probable, és que no s'hagi trobat la clau, aleshores, per tant, s'assimilarà la unitat de treball i s'actualitzaran

[31] A l'apartat 4.5 es va veure que l'assimilador examina un resultat sempre i quan aquest hagi estat validat prèviament pel verificador.

les taules pròpies d'aquest projecte (*tested_keys* i *project_status*). El procés d'actualització s'explica seguidament. En cas que s'hagi trobat la clau (ja validada), es procedeix a desxifrar el text sencer i assimilar la unitat de treball.

En el primer dels dos casos anteriors, s'actualitzarà la taula *tested_keys* incrementant el nombre d'unitats de treball computades per aquell paquet. En cas que el nou nombre sigui igual al nombre total de unitats de treball d'aquell paquet, s'actualitza l'estat del paquet marcant l'atribut *status* amb el valor '*F*'. La taula *project_status* s'actualitza de manera que, si el resultat anterior a aquest té la mateixa data, s'actualitza l'últim registre de la taula incrementant el nombre de unitats de treball computades en 1. En cas contrari es crea un nou registre, on també s'incrementa el nombre d'unitats de treball computats respecte l'anterior. En cas que s'hagués trobat la clau anteriorment, es mantenen els valors dels camps corresponents.

En el segon dels casos, la taula *tested_keys* s'actualitza amb el mateix procediment que el cas anterior. La taula *project_status*, en canvi, s'actualitza, també com abans, però afegint els valors de la clau, el text desxifrat prèviament calculat i la *id* de l'usuari. En el cas improbable de que s'hagués trobat una clau amb anterioritat, aquests valors s'actualitzen amb els nous, de manera que es pogués veure la situació d'excepcionalitat existent.

Generador de tasques

Com a última aplicació desenvolupada hi ha el generador de tasques, que en aquest cas no és cap dimoni. Tot i que l'entorn BOINC proporciona un programa anomenat *create_work* per a generar tasques, aquest serveix per crear unitats de treball a l'hora de provar el funcionament general de l'aplicació però és del tot insuficient per crear paquets de moltes unitats de treball, ja que s'han de crear manualment els fitxers de plantilla per cada unitat i resultat.

Per aquest motiu s'ha creat un generador automàtic de tasques, que permet crear de cop fins a 2^{32} tasques. Per invocar-lo se li ha de passar els 40 bits superiors de la clau inicial del rang i, opcionalment, els 40 bits superiors de la clau final de rang. Sinó se li passa aquest segon paràmetre llavors només es crea una única unitat de treball. També

opcionalment, se li pot passar el nombre de resultats per cada unitat de treball, per si es volgués incloure redundància en el còmput. També se li ha d'indicar de quina de les aplicacions es vol crear unitats de treball.

El programa s'ha desenvolupat seguint la següent estructura de directoris. Els fitxers *xml* generats per cada unitat de treball i cada resultat es guardaran al directori local del projecte anomenat *workunits/rc5_wus*. El codi dels fitxers *xml* generats està *hardcoded* en el codi font del programa, per simplificar la generació d'aquests. Els fitxers d'entrada de cada unitat de treball, per defecte s'han de guardar en el directori *download*.

L'aplicació també s'encarrega d'inserir a la base de dades els registres necessaris, que només són les de cada unitat de treball[32]. La inserció es fa a partir de la crida que proporciona l'API de la BOINC per tal efecte.

5.2. Posada en funcionament

La posada en funcionament de la infraestructura de la BOINC compren la instal·lació de tot el software necessari al servidor. Per fer aquest procés s'ha seguit la guia anomenada *Project creation cookbook*, que proporciona la pròpia pàgina de creació de projectes de la BOINC. Algunes modificacions han estat necessàries, ja que la guia no inclou la solució a alguns problemes trobats durant la posada en funcionament de la infraestructura. A l'annex C s'inclou una guia de referència ràpida, amb les comandes necessàries per la posada en funcionament del projecte. També s'explica en aquest apartat la instal·lació del client en els ordinadors de sobretaula.

[32] Com s'ha vist en l'apartat de la BOINC, és el transicionador qui s'encarrega de generar els registres de la taula de resultats a la base de dades.

Instal·lació del software de la BOINC

El servidor:

La instal·lació del servidor es fa sobre el sistema operatiu Fedora GNU/Linux amb les següents aplicacions instal·lades [24]:

- Les eines GNU següents: *make 3.79+, m4 1.4+, libtool 1.4+, pkg-config 0.15+, autoconf 2.58+, automake 1.8+* i *GCC 3.0.4+,*
- *Python 2.2+*, el mòdul *MySQLdb 0.9.2+* i el mòdul *xml* per *Python*,
- *MySQL 4.0+* o *4.1+* i el client de *MySQL*,
- *Apache* amb *mod_ssl* i *PHP 5+,*
- *OpenSSL 0.9.8+*,
- i *libcurl 7.15.5*.

D'altra banda, es necessita tenir instal·lat el sistema de control de versions CVS[33] (el client) per descarregar-se el codi font del servidor de la BOINC. Addicionalment, es pot descarregar les aplicacions client d'exemple. D'aquesta manera s'obtenen en el directori local, entre altres, el codi font del servidor i del client (aquest és opcional d'instal·lar en l'ordinador servidor), les APIs, la documentació i una interfície web molt bàsica pel projecte.

Pel correcte funcionament del servidor, cal comprovar que la memòria compartida estigui activada i tingui un segment de dades de com a mínim 32 megabytes (tot i dependre del sistema operatiu). A més a més, es pot passar un test de comprovació recomanat per la BOINC per assegurar-se que MySQL funciona i es tenen els permisos configurats correctament.

També s'ha de configurar bé la base de dades MySQL per permetre accedir-hi a l'Apache (que se sol executar com usuari *nobody*). A més a més, s'ha de modificar l'arxiu *httpd.conf* del servidor web Apache per indicar el tipus per defecte MIME

[33] Quan es va començar amb la instal·lació aquest sistema era l'únic disponible per descarregar-se tot el codi font necessari. En el transcurs de la posada en funcionament del servidor han canviat el sistema CVS pel SVN i, a més a més, es pot descarregar el codi font mitjançant la web que han habilitat en format *.tar.gz.*

application/octet-stream. Pel que fa al PHP, s'ha d'habilitar les *magic quotes* al fitxer *php.ini.*

El client:

Segons el sistema operatiu utilitzat pel voluntari, es necessiten certs prerequisits de llibreries. Si es vol instal·lar el client de la BOINC a Windows, tan sols cal descarregar-se el client de la pàgina web oficial i instal·lar-lo seguint els passos necessaris indicats pel instal·lador.

En el cas de GNU/Linux, per instal·lar el *Core Client* i el *BOINC Manager* es necessita, apart de les eines GNU anteriorment indicades i les llibreries *OpenSSL* i *libcurl*, els *WxWidgets 2.8.3* i les llibreries *jpeglib* per X11.

Mentre que en Windows executant l'enllaç que apareix al menú d'inici és suficient perquè el client comenci a treballa, en GNU/Linux cal, en primer lloc, cridar al *core client* des de línia de comandes, que es pot fer amb l'*script run_client*. Per a que no molesti amb missatges per consola, es pot invocar el client amb l'opció *–daemon* per a que s'executi en segon pla, sense missatges molestos. Aleshores cal invocar al *boincmgr* que és la interfície gràfica utilitzada per gestionar els projectes dels quals es formi part de forma còmoda. Per poder treballar directament amb el client, es pot utilitzar l'aplicació *boinc_cmd*, que permet enviar ordres directament al c*ore client.*

Creació i posada en funcionament del projecte

Un cop instal·lat el servidor de la BOINC i arrancat el MySQL s'han de seguir un conjunt de passos per crear i personalitzar el nostre projecte. Primer, cal situar-se al directori on hi ha les fonts de la BOINC i executar l'*script make_project* de la següent manera:

```
$./tools/make_project
-project_root           /boinc-projects/rc5
-url_base         http://ccd-pr7.uab.cat
--delete_prev_inst
--drop_db_first
--db_name rc5
-v
rc5
```

S'ha de configurar el servidor web *Apache* per donar accés a la pàgina del projecte, incloent al final del fitxer *httpd.conf* el contingut del fitxer *rc5.httpd.conf* situat al directori del projecte. Aquest últim, inclou la configuració necessària per a que l'Apache accedeixi al directori *html* per mostrar la web del projecte. Per executar les tasques periòdiques del projecte, es poden definir al fitxer *config.xml*.

Cal que el fitxer *project.xml* estigui en el directori del projecte. Aquest fitxer es troba en el directori de les fonts de la BOINC i cal editar-lo per reflectir la realitat d'aquest projecte respecte a les plataformes suportades per les aplicacions d'aquest i també sobre quines aplicacions disposa el projecte. En aquest projecte, les plataformes suportades són GNU/Linux i Windows sobre arquitectures de 32 bits *x86*, per tant s'han d'eliminar les entrades corresponents, per exemple, PowerPC i SPARC. Pel que fa a les aplicacions, momentàniament no cal modificar-ho, en futurs passos s'afegirà una aplicació de prova per comprovar que el servidor funciona.

Per a que els clients puguin connectar-se al planificador del projecte, cal editar el fitxer *schedulers.txt* situat al subdirectori *html/user* per incloure-hi la URL d'accés al planificador. En aquest punt, és important ressaltar que per cada planificador que disposi el projecte s'ha d'incloure una nova línia amb el següent format:

```
<scheduler>http://ccd-pr7.uab.cat/rc5_cgi/cgi</scheduler>
<link       rel="boinc_scheduler"       href="http://ccd-
pr7.uab.cat/rc5_cgi/cgi">
```

Cal destacar que cada entrada d'aquest fitxer ha d'anar en una sola línia ja que del contrari, si un participant intenta afegir-se al projecte rebrà un error indicant que la URL del planificador no conté cap informació sobre el projecte.

Per la seguretat del projecte, cal configurar l'Apache per a que no permeti el llistat de directoris i s'ha de protegir el subdirectori *html/ops* amb contrasenya mitjançant els fitxers *.htaccess* i *.htpasswd* que proporciona el servidor web per aquest propòsit.

Respecte al servidor web, cal assegurar-se que aquest tindrà permisos d'accés als fitxers de la web. Si no és així cal afegir la configuració necessària als fitxers de gestió d'usuaris i grups.

L'altre fitxer de configuració del projecte a tenir en compte és el *config.xml*. S'ha d'editar aquest fitxer per habilitar la creació de comptes d'usuari, inicialment deshabilitada, per fer-ho cal editar el tag *disable_account_creation* indicant-hi el valor 0.

La configuració dels dimonis també cal modificar-la. Tot i que inicialment no sigui necessari, quan el servidor de la BOINC ja es troba funcionant correctament cal canviar el paràmetre *–d* pel valor 1, per tal que no es generin uns fitxers de *log* que desbordin l'espai a disc. Aquest paràmetre regula el què es guarda als fitxers de *log* i inicialment aquest està a nivell *debug* per poder comprovar que el sistema funciona degudament. Per evitar-ho al inici, es pot afegir una o diverses tasques periòdiques en el fitxer de configuració afegint, per exemple, la següent opció:

```
<task>
<cmd> cat /dev/null </cmd>
<output> log_USER/feeder.log </output>
<period> 1 week </period>
</task>
```

El que es fa és buidar el fitxer de log del *feeder* (aquest és el que més s'omple) cada setmana.

Seguint amb altres fitxers de configuració, s'han de modificar (tot i no ésser necessari) els paràmetres *PROJECT*, *COPYRIGHT_HOLDER* i *SYS_ADMIN_EMAIL* del fitxer *html/project/project.inc* per adequar-los a la realitat del projecte.

Per afegir funcionalitat al menú del client de la BOINC s'ha de crear el fitxer *gui_urls.xml* al directori principal del projecte i posar-hi els enllaços desitjats (p.e. la pàgina del compte de l'usuari).

Finalment, existeix un fitxer *php* que mostra l'estat dels diferents components del projecte (p.e. base de dades, dimonis, planificador, etc). Aquest fitxer està al subdirectori */html/user* i s'anomena *server_status.php*, i es pot modificar per incloure-hi els dimonis com l'assimilador i el verificador.

Abans de començar a afegir aplicacions i unitats de treball és important posar en marxa els dimonis del projecte, ja que sinó al crear unitats de treball no es creen els resultats associats a la base de dades.

Per comprovar el correcte funcionament del projecte es pot utilitzar una de les aplicacions de prova. En aquest cas, s'ha triat l'aplicació *uppercase* que donat un fitxer d'entrada amb un text retorna un fitxer de sortida amb el mateix text en majúscules.

S'afegeix l'aplicació al fitxer *project.xml*, tal com s'ha comentat anteriorment. Seguidament es copia l'aplicació del directori amb els fonts del servidor BOINC al directori *apps* del projecte. Es crea un subdirectori amb el nom de l'aplicació on s'hi ha de copiar l'aplicació canviant-li el nom, seguint aquest format:

```
nom_versió_plataforma[.extensió]
```

Per actualitzar la base de dades (concretament la taula *app_version*) s'executa l'*script update_versions*, des del directori principal del projecte. Es creen els fitxers *xml* per definir una o diverses unitats de treball i resultats a partir de les plantilles situades al subdirectori *templates*. També cal crear un fitxer anomenat *in* amb un text de prova i guardar-lo al directori *downloads*. Finalment es crida el programa anomenat *create_work* per afegir noves unitats de treball a la base de dades. Es pot comprovar que s'ha afegit correctament la unitat de treball i el resultat consultant la base de dades. També cal comprovar el *log* del *feeder* per veure que aquest ha afegit la unitat de treball al segment de memòria compartida. També es pot executar el programa *show_shmem*

que indicarà quines unitats de treball estan llestes per ser enviades als usuaris que es connectin i en quin ordre.

Per comprovar que el servidor i les aplicacions funcionen degudament cal connectar-se amb el client de la BOINC i registrar-se al projecte, creant un usuari[34], per descarregar-se les dades per les tasques de prova. Es poden seguir els *logs* emmagatzemats al servidor i consultar la base de dades per verificar que s'han processat bé les tasques de prova i que s'han rebut els resultats correctament.

Compilació de les aplicacions

Tot i ser habitualment un procés molt simple de realitzar, en aquest cas és força més complicat ja que les aplicacions de la BOINC necessiten de força llibreries pròpies d'aquesta. Les APIs de BOINC estan estructurades de manera que les llibreries amb funcionalitats semblants estan en un mateix subdirectori, igual que les aplicacions. Per exemple, les llibreries d'accés a la base de dades del projecte i les utilitats del projecte. Per les aplicacions pròpies del projecte es pot crear un directori nou o utilitzar el mateix que les aplicacions de prova proporcionades. En qualsevol dels dos casos s'han de modificar o crear fitxers de *Makefile* per a resoldre les dependències de llibreries específiques de la BOINC. La tasca de crear o modificar aquests fitxers és força feixuga en aquest cas, ja que es poden trobar fitxers de *Makefile* amb centenars o milers de línies de codi.

Per tant, si s'utilitza el mateix subdirectori per compilar les aplicacions del projecte, com ha estat el cas, és necessari modificar el fitxer de *Makefile* per afegir-hi les aplicacions desenvolupades. Una segona modificació molt necessària ha estat causada per les llibreries dinàmiques. Al compilar una aplicació client de la BOINC en GNU/Linux aquesta ha d'executar-se en diferents distribucions, per tant, s'ha de fer de manera que s'utilitzi el mínim de llibreries de forma dinàmica, ja que es poden donar molts problemes que fan que l'aplicació causi un error. Aquest fet és degut a que el carregador del sistema operatiu va a buscar les llibreries dinàmiques al directori on l'executable indica que les té *linkades* i, si no les troba, aleshores succeeix un error.

[34] Consultar l'apartat 6.1 per realitzar-ho

Doncs, també es necessita modificar els fitxers de *Makefile* que s'utilitzen per modificar les aplicacions client per a que el *linkatge* de totes les llibreries es faci de forma estàtica. Aquest fet implica que el codi necessari de les llibreries s'afegeix directament a l'executable, incrementant la mida d'aquest considerablement.

Pel cas de Windows, s'ha utilitzat el Visual Express per compilar les aplicacions client. La BOINC proporciona un projecte ja creat, amb totes les llibreries incloses per poder realitzar la compilació de les aplicacions. Tant sols cal substituir l'aplicació *upper_case* de mostra per l'aplicació del projecte.

Pel que respecta als dimonis específics del projecte, aquests es poden compilar utilitzant el fitxers de *Makefile* que també s'utilitzen per compilar el planificadors, el gestor de pujades i els altres dimonis. No cal modificar-lo ja que es poden utilitzar els fitxers de *placeholder* que s'han comentat a l'apartat 5.1. Finalment, pel cas de les aplicacions genèriques no es necessita realitzar cap acció.

6. Funcionament del projecte

El funcionament del projecte, des del punt de vista de l'usuari que cedeix els seus recursos computacionals, es pot dividir en dues parts: el registre i el càlcul d'unitats de treball. Pel que respecta al registre d'usuaris, l'apartat 6.1 descriu pas a pas com ho han de fer aquests per registrar-se. El càlcul d'unitats de treball, és automàtic un cop l'usuari està registrat, tot i que aquest pot triar en quin moment vol calcular unitats de treball i el tant per cent de recursos a destinar-hi. Finalment es veu com s'adapta el procés vist al punt 4.7, sobre el cicle de vida de les unitats de treball pel cas concret d'aquest projecte, des del punt de vista de l'administrador.

6.1. Registre dels usuaris al projecte

El registre es pot fer de dues maneres:

- via un formulari web
- mitjançant el client de la BOINC.

Si s'opta per la primera opció, sols cal accedir a la web del projecte i omplir el formulari amb les dades demanades (correu electrònic, nom, contrasenya i, opcionalment, el país i el codi postal). L'adreça per accedir a aquest projecte és la següent:

http://ccd-pr7.uab.cat/rc5

La Figura 10 mostra el formulari de registre al projecte, anomenat "create an account", que es troba accessible des de l'enllaç mencionat.

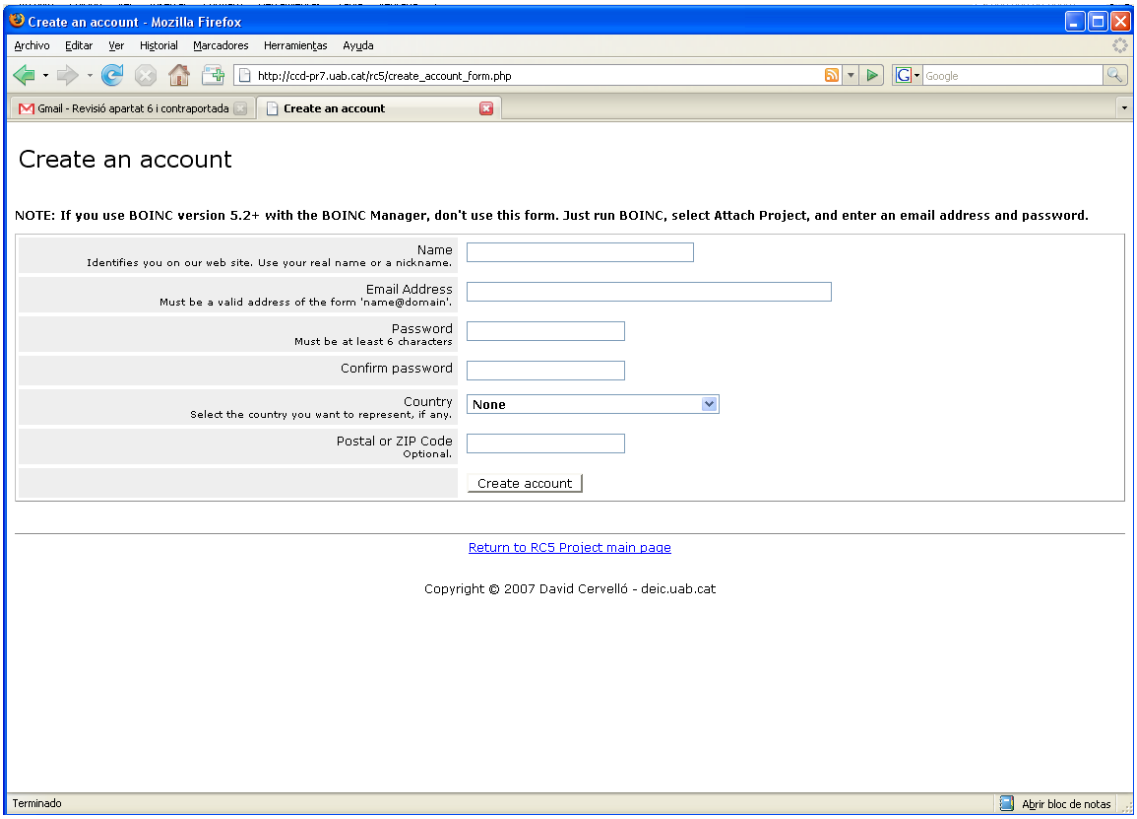

Figura 10: Creació d'un compte a través de l'interfície web

La segona forma de registrar-se al projecte és a través del client de la BOINC. El client de la BOINC disposa de dues interfícies, la interfície bàsica i l'avançada, segons el nivell de control que es vulgui tenir sobre aquest. El registre es fa alhora d'adjuntar-se al projecte, a l'enllaç "Add project", indicant que es vol crear un compte nou i introduint el correu electrònic i la contrasenya. Les figures 11, 12 i 13 mostren el procés utilitzant la interfície bàsica del client de la BOINC.

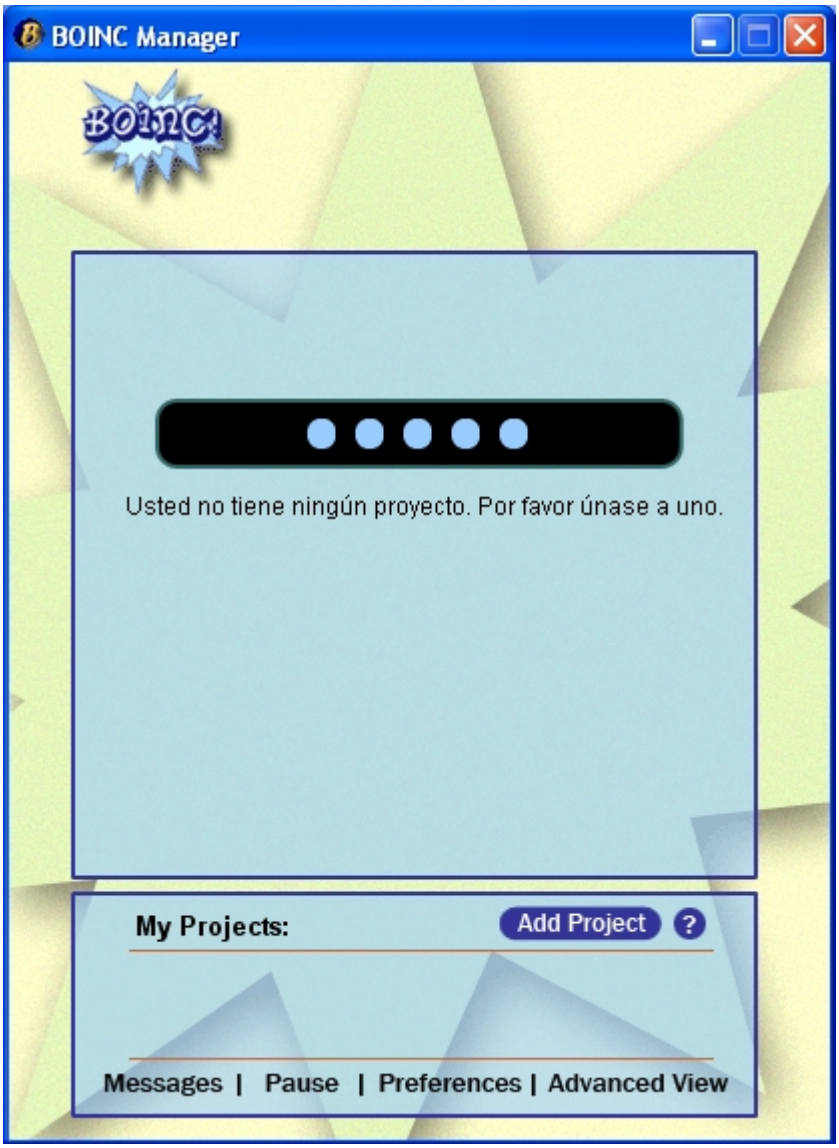

Figura 11: Interfície bàsica del client de la BOINC

Figura 12: Creació d'un compte mitjançant el client de la BOINC (pas 1)

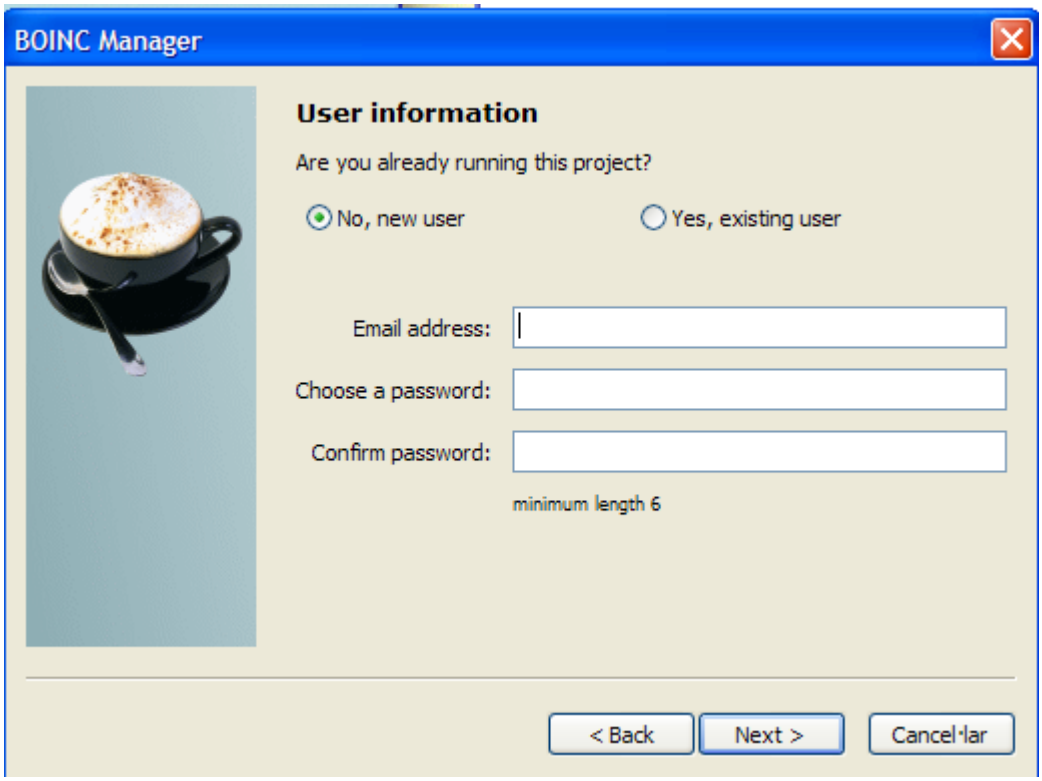

Figura 13: Creació d'un compte mitjançant el client de la BOINC (pas 2)

En canvi, si es tria la interfície avançada del client per registrar-se a un projecte, cal anar a “Eines”, “Attach to a project” i la resta del procés consisteix en els mateixos quadres de diàleg.

El registre a través de formulari web té l’avantatge de poder modificar les preferències de l’usuari (per exemple, l’ús de la màquina) abans d’afegir-se al projecte. Un cop registrat amb el formulari, s’utilitza el client BOINC per afegir-se al projecte. Concretament cal marcar l’opció “Yes, existing user” de la pantalla que apareix a la Figura 13 i introduir les dades de registre subministrades al formulari web.

Si es fa el registre a través del client BOINC, en canvi, l’usuari s’adjunta al projecte sense fer cap més pas.

6.2. Interfície del client de la BOINC pel càlcul de resultats

Si s’utilitza l’interfície bàsica, no hi ha gaires opcions, ja que tota la informació relativa als projecte en els que es participa i les tasques que s’estan executant apareix condensada en la mateixa finestra (veure Figura 14).

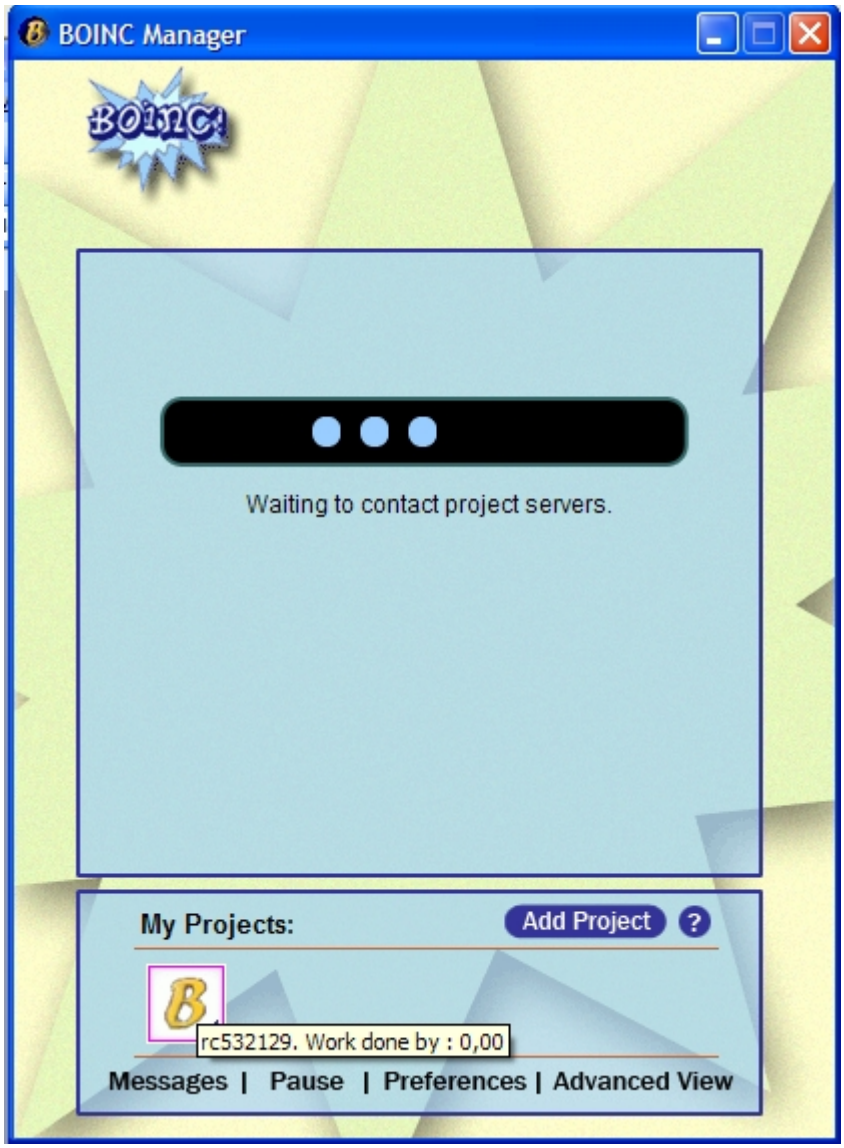

Figura 14: Interfície bàsica de la BOINC amb projectes afegits

Com a informació addicional es poden consultar els missatges que s'envien entre el client i el servidor, clicant el botó anomenat "Messages". Es pot pausar l'execució mitjançant el botó "Pause" i es poden modificar les preferències clicant al botó "Preferences". Finalment, es pot passar a la interfície avançada mitjançant el botó anomenat "Advanced View".

La interfície avançada ofereix més control i més informació sobre els projectes en els que es participa (veure Figura 15). Es disposa de sis pestanyes, que són les de:

- Projectes
- Tasques
- Transferències

- Missatges
- Estadístiques
- Disc

En la pestanya "Projectes" es pot veure en quins projectes es participa, i també la informació referent a la quantitat de feina feta, la distribució de recursos i l'estat del client respecte a aquell projecte. L'estat pot variar entre estar pendent d'enviar una petició al servidor i que aquest la contesti, estar calculant algun resultat, etc. També es pot controlar si es vol pausar la comunicació amb el servidor i l'execució de tasques d'aquell projecte, si s'accepten més tasques per calcular i es pot forçar la comunicació amb el servidor. A més a més, si el projecte incorpora enllaços a pàgines concretes de la seva web o altres pàgines, també es poden veure en aquesta pestanya. La figura 15 mostra el contingut d'aquesta pestanya.

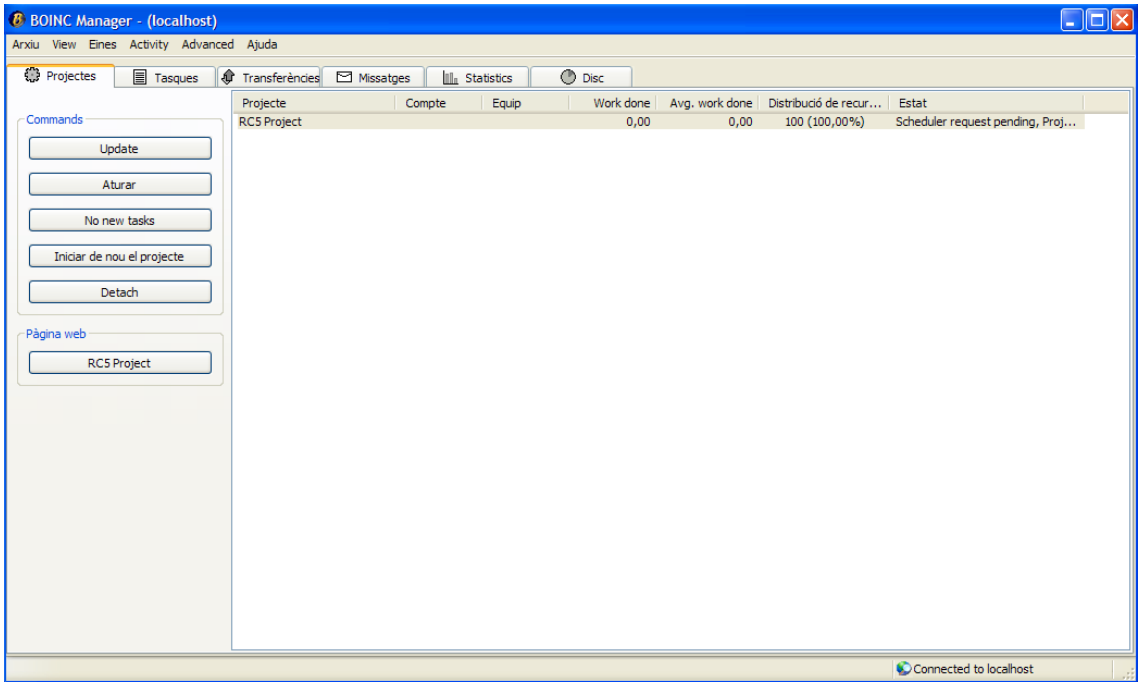

Figura 15: Pestanya principal de l'interfície avançada del client de la BOINC

En la pestanya "Tasques" es pot veure la informació sobre el progrés de l'execució de tasques en cada moment. Aquesta informació inclou el projecte al que pertanyen, el nom de l'aplicació i de la tasca, el temps de CPU utilitzat, el tant per cent de progrés, el temps que falta per completar-se, el límit per informar i l'estat. Els camps del tant per cent de progrés i el temps restant per completar-se només es veuen actualitzats si

l'aplicació del projecte els calcula. A més, també es poden aturar i abortar des d'aquesta pestanya. En la Figura 16 es veu aquesta pestanya, amb una tasca d'aquest projecte calculant-se.

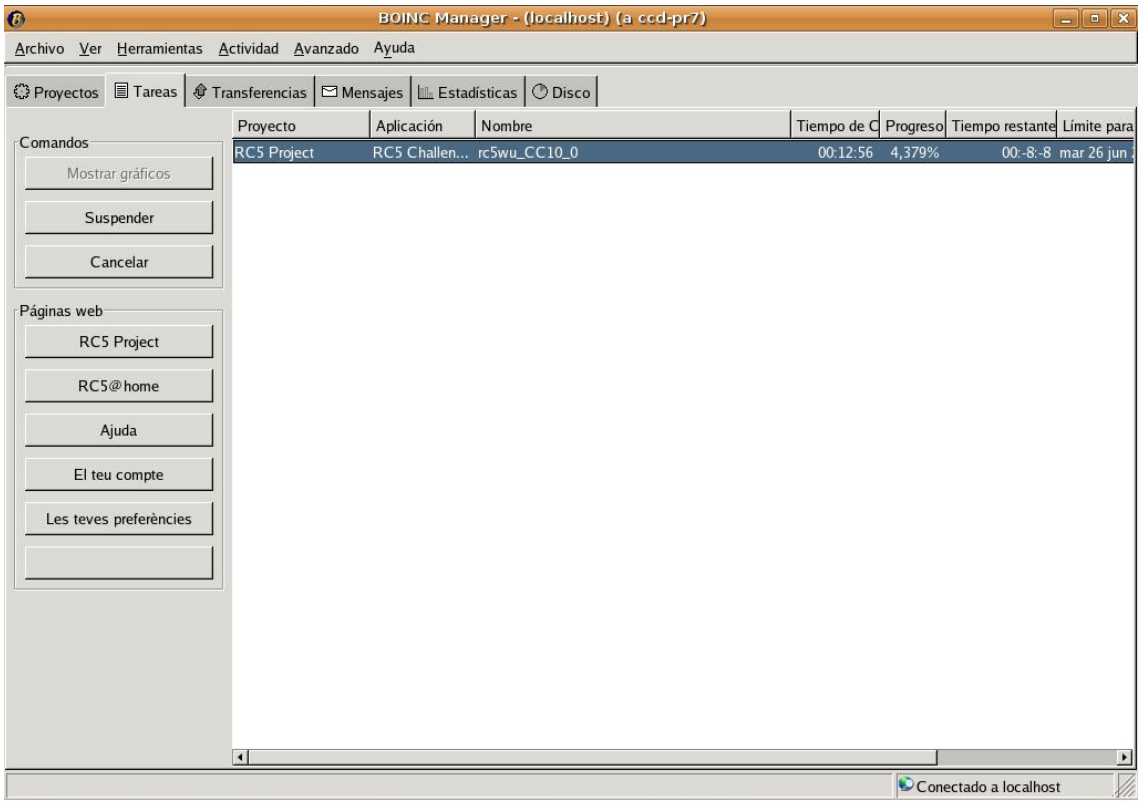

Figura 16: Pestanya de tasques de l'interfície avançada del client de la BOINC

En la pestanya de "Transferències" es pot veure l'estat de les transferències de fitxers entrants i sortints així com aturar aquestes transferències. En la pestanya de "Missatges" es pot consultar amb detall la "conversa" que mantenen el client i el servidor, els possibles errors que es produeixin a l'intentar accedir al servidor, o errors al costat del client. Entre aquests errors es pot trobar que el servidor estigui parat per manteniment, problemes amb la memòria compartida del servidor, problemes amb la signatura digital de les aplicacions i fitxers, etc. També es poden consultar en aquesta pestanya els resultats dels *benchmarks* si s'han realitzat des de l'últim cop que s'ha iniciat el client. En la pestanya "Estadístiques" es mostren les estadístiques globals de l'usuari, per cada màquina i per projecte. Finalment, a la pestanya "Disc" es pot consultar el percentatge de disc usat entre tots els projectes als que l'usuari està inscrit en aquella màquina, i l'ús per projecte.

6.3. Creació, càlcul i validació d'unitats de treball

La posada en funcionament del projecte es pot descriure en base al cicle de vida de les unitats de treball, que s'ha descrit en l'apartat 4.7, i que es mostra de forma gràfica en la Figura 9.

El cicle de vida de tota unitat de treball comença amb la generació dels fitxers de plantilla per la pròpia unitat de treball i pel resultat. Per aquest projecte no cal editar manualment aquests fitxers, ja que la utilitat *rc5_make_work,* desenvolupada específicament per a aquest projecte, els crea de forma automàtica. Sols es necessita el rang de claus a provar, sense comptar els primers 32 bits. A partir d'aquest rang s'extreuen el nombre d'unitats de treball que cal generar, el nom de les unitats de treball i dels resultats[35], els fitxers de plantilla i els fitxers d'entrada de cada unitat de treball. Aquesta aplicació també introdueix els registres necessaris a la taula de *workunit*. El transicionador, com ja s'ha comentat anteriorment, genera tots els registres a la taula *result* a la propera activació. Per tant és important, en aquest punt, no esborrar els fitxers de plantilla generats per l'*rc5_make_work*, ja que del contrari, el transicionador en la seva propera activació fallaria a l'intentar generar els resultats corresponents.

Quan l'alimentador s'activa després d'haver-se generat els resultats, aquest omple l'espai de memòria compartida amb els resultats pendents de processar pel client. Quan els clients es connecten, a través del planificador, aquest consulta l'espai de memòria compartida habilitat per l'alimentador i si hi ha resultats els envia, d'un en un. El que realment els hi envia el planificador és el contingut dels fitxers *xml* i, en cas que sigui la primera vegada que s'adjunta al projecte, també se'ls envia l'aplicació client.

Aleshores els clients s'han de descarregar el fitxer d'entrada a partir de la URL dels fitxers *xml.* Un cop totes les dades necessàries estan descarregades, el client executa l'aplicació client en segon pla. Sigui quin sigui el resultat, a l'acabar d'executar l'aplicació client del projecte, el client de la BOINC envia el fitxers de sortida a través del gestor de pujades, si aquest existeix. El proper cop que el client es connecti al planificador, aquest enviarà tots els resultats pendents, incloent l'estat de finalització de

[35] Cal recordar que aquests noms han de ser únics entre tots els noms d'unitats de treball i resultats de la base de dades.

l'aplicació client i la sortida d'error del programa. L'alimentador, cada cop que s'activi comprovarà si hi ha espais buits a la memòria compartida i si existeixen resultats a la base de dades que encara no s'hi hagin afegit. En cas que es doni aquesta situació, afegeix els resultats necessaris per mantenir plena la memòria compartida.

En cas que el resultat falli, es podrà consultar a la base de dades la sortida d'error, que ajuda a depurar els possibles errors de l'aplicació, o determinar si ha estat culpa del client. Quan es disposi de totes les dades necessàries per validar el resultat i es torni a activar el verificador, aquest comprova la sintaxi del fitxer de sortida. La validació, com ja s'ha comentar a l'apartat 5.1, dóna com a correcte un resultat si la sintaxi és correcta i en cas que inclogui la clau guanyadora, si aquesta realment desxifra el criptosistema. En cas que el resultat generi un error, el transicionador en genera un de nou, sempre i quan no s'hagi superat el nombre màxim de resultats erronis. En aquest últim cas, la unitat de treball es dóna per perduda.

Per acabar el procés, entra en funcionament l'assimilador, que comprova si el resultat conté la clau guanyadora i actualitza les taules pròpies d'aquest projecte. A més a més, si s'ha arribat al quòrum necessari per la unitat de treball corresponent, aquesta passa a l'estat de completada. Si l'assimilador ha marcat els resultats i la unitat de treball per a que s'eliminin els fitxers d'entrada i sortida, durant la propera activació de l'eliminador de fitxers, aquest els esborra de disc.

6.4. Proves realitzades

En l'apartat de desenvolupament s'ha vist com el projecte es dividia en dues etapes, corresponents al desenvolupament de l'aplicació de desxifrat per l'RC5 i a la implantació del servidor de la BOINC. Cadascuna ha tingut les seves proves.

Per la primera de les etapes, les proves realitzades corresponen a les diferents aplicacions desenvolupades que s'han vist a l'apartat 5.1. Primer, les proves de desxifrat correcte de les dades del pseudo test d'RSA Laboratories, que es comprova amb l'aplicació *rc5ValidaTest*. L'exemple del pseudo test és el següent:

Clau	IV	Text xifrat (en hexadecimal)
c9 0c 03 53 c0 d4 e1 fe 85	07 ce 59 1f 86 14 9a 41	5a 28 2d 56 2a 85 b7 2f ef ae 30 ba 93 54 da 15 ac ba 79 1a a1 cb 0e 85 9a c3 31 a5 36 19 85 e3 77 df dd 07 34 3b fa ed 84 84 41 90 9d 7b 6f 17 2d 51 ea b2 1c 5e 45 1c e9 6a 34 de a5 f5 b9 56 22 16 e2 60 15 e1 7e b2

Taula 5: Prova de xifrat de l'aplicació *rc5ValidaTest*

Després de desxifrar-lo amb l'aplicació anterior s'obté:

Clau	IV	Text en clar (en hexadecimal)
c9 0c 03 53 c0 d4 e1 fe 85	07 ce 59 1f 86 14 9a 41	54 68 65 20 75 6e 6b 6e 6f 77 6e 20 6d 65 73 73 61 67 65 20 69 73 3a 20 54 68 65 20 52 43 35 2d 33 32 2f 31 32 2f 39 2d 74 65 73 74 20 63 6f 6e 74 65 73 74 27 73 20 70 6c 61 69 6e 74 65 78 74 08 08 08 08 08 08 08 08

Taula 6: Prova de desxifrat de l'aplicació *rc5ValidaTest*

L'aplicació *rc5TestClaus* és la més interessant de totes, ja que, com s'ha explicat, ha servit per decidir si crear unitats de treball de 2^{30} ó de 2^{32} claus. Aquesta aplicació calcula els temps que es necessita per provar diferents rangs de claus, entre ells els de 2^{30} i 2^{32} claus. Per tenir una idea aproximada de perquè dóna un temps o un altre, al passar els testos en diverses màquines s'ha omplert un formulari amb les dades de la màquina i què s'està fent en el moment d'executar l'aplicació. El formulari és el següent:

Dades de l'ordinador:

- Sistema Operatiu:
- Arquitectura:
- Tipus Placa Base:
- Tipus de Processador:
- Velocitat del Processador:
- Memòria RAM:

Altres dades:

- Us de l'ordinador durant el test:

Es demanen dades sobre el hardware de la màquina, per veure si els temps varien molt entre diferents processadors i plaques base. Una de les dades més rellevant a l'hora de determinar el temps de càlcul és el sistema operatiu utilitzat ja que entre Windows i GNU/Linux[36] hi ha una diferència notòria. Mentre que en GNU/Linux es compila amb el GCC i aquest compilador disposa d'opcions de compilació que aconsegueixen reduir el temps de l'aplicació notablement, en Windows s'ha utilitzat el Visual Express i aquest no optimitza el codi tant eficaçment com GCC. La Taula 7 mostra una comparativa de temps per diversos rangs de claus pels dos sistemes operatius

S.O. \ Claus	2^{20}	2^{25}	2^{30}	2^{32}
GNU/Linux	2s	50s	1895s	6482s
Windows	2s	61s	2042s	7761s

Taula 7: Comparativa de temps per diversos rangs de claus i sistemes operatius

Les dades d'aquesta taula s'han obtingut realitzant les proves en una mateixa màquina, les característiques de la qual es mostren a continuació:

- Sistema Operatiu: *GNU/Linux - Windows*
- Arquitectura: *AMD 3*2 bits
- Tipus Placa Base: K7
- Tipus de Processador: *Duron*
- Velocitat del Processador: *1800 MHz*
- Memòria RAM: *1024 Mb*

Altres dades:

- Us d l'ordinador durant el test: *sense usar*

En l'annex D es detallen els temps d'execució d'aquesta aplicació en diverses màquines.

Per altra banda, les proves a les que s'ha dedicat més temps corresponen a la posada en funcionament de la infraestructura BOINC. En aquest punt és on s'han trobat més problemes i alguns defectes en l'aplicació client. Per citar algun dels problemes de

[36] Cal recordar són els únics sistemes operatius sobre els quals s'ha compilat l'aplicació.

l'aplicació client, s'ha vist que si s'executava en mode *standalone*[37] funcionava degudament però en mode client el *thread* principal acabava sense esperar als altres *threads*. Aquest fet provocava que mai es tingués en compte el resultat de la cerca de la clau alhora d'escriure els resultats al fitxer de sortida. El defectes en l'aplicació client han estat solucionats i per tant ara sí es té en compte el resultat de provar el rang de claus corresponent. Les proves que s'han comentat s'han realitzat amb una aplicació client pel pseudo test, en les que sí es coneix la clau que trenca el criptosistema.

L'aplicació client del pseudo test també ha servit per realitzar proves d'enviament dels fitxers de l'aplicació i els d'entrada i sortida entre client i servidor i comprovar que la signatura digital del codi que proveeix la BOINC funciona correctament. Igualment, aquesta aplicació ha servit per comprovar qualsevol possible error alhora d'adaptar l'aplicació original de la primera etapa del projecte per utilitzar les APIs de la BOINC. S'han trobat molts problemes alhora de resoldre els noms dels fitxers d'entrada i sortida, ja que aquests depenen de l'atribut corresponent de les unitats de treball i dels resultats i s'ha trobat una manca important de documentació sobre aquest tema.

Per comprovar el correcte funcionament del verificador i l'assimilador també s'ha utilitzat l'aplicació del pseudo test, però, en aquest cas, limitada a provar 1000 claus. De tal manera que si se li passa els 40 bits superiors de la clau corresponents a la clau del criptosistema, la clau guanyadora estigui en aquest rang. S'ha fet així per estalviar-se temps alhora de realitzar aquestes proves, ja que de l'altra manera, es necessitava més d'una hora per provar qualsevol canvi.

Les proves s'han realitzat per verificar el correcte funcionament en els dos casos possibles, en que es troba la clau i en que no. En cas que es troba la clau també s'ha alterat el fitxer de sortida canviant la clau abans d'ésser verificada per comprovar que la funcionalitat de detectar possibles alteracions del fitxer de sortida funciona correctament. Igualment, per l'assimilador s'ha comprovat que si es rep la clau que desxifra el text, aquest és desxifrat i emmagatzemat correctament, eliminant, a més a més, els bits de farciment que pugui incloure el text en clar.

[37] El mode *standalone* és un mode d'execució de les aplicacions client per provar-ne el correcte funcionament, tot i que, irònicament, en aquest cas el funcionament en el mode client (quan l'aplicació s'executa sota el *core client* de la BOINC) i en el mode *standalone* diferien considerablement.

En aquest punt, un cop verificada tota la funcionalitat tant de l'aplicació client, com dels dimonis propis del projecte com de les taules afegides a la base de dades, ja es pot posar en funcionament el servidor de la BOINC per intentar trencar el criptosistema del desafiament. A data d'impressió d'aquesta memòria, es porta aproximadament un mes en execució, amb les màquines del laboratori Kleinrock del dEIC calculant unitats de treball durant les nits i els caps de setmana. Es porten 7.692.286.427.136 claus provades, que corresponen a 1791 unitats de treball.

7. Conclusions

Aquest projecte ha desenvolupat la infraestructura de càlcul distribuït necessària per iniciar el desafiament proposat per la companyia RSA Laboratories per trencar el criptosistema RC5-32-12-9 i en l'actualitat es troba en funcionament utilitzant les màquines dels laboratoris del dEIC testejant unitats de treball amb rangs de 2^{32} claus.

El projecte va començar amb l'estudi de l'algorisme de xifrat RC5, per tal d'entendre el seu funcionament i posteriorment es va procedir a buscar una implementació que s'adaptés a les necessitats del projecte. Aquesta implementació s'ha utilitzat, en primer lloc, per obtenir el temps de processat de paquets de 2^{30} i 2^{32} claus que han permès fixar la mida dels paquets a processar. S'ha triat una mida fixa per facilitar el funcionament en general del projecte, ja que una mida variable complicaria tota la lògica de validació i control del nombre total de claus calculades.

El procés d'aquest projecte segueix amb l'estudi de la plataforma de càlcul distribuït BOINC. Durant el transcurs d'aquest projecte els creadors de la BOINC varen realitzar diversos canvis tant en el client com en el servidor. A més a més, aquests canvis van arribar amb documentació en part nova i força més complerta. Alguns dels problemes amb els que s'ha topat durant la realització del projecte haurien estat solucionats de forma més ràpida amb aquesta nova documentació. Per tant, s'haurien pogut dur a terme algunes de les millores que s'exposen al següent apartat.

La realització del projecte segueix, doncs, amb la posada en funcionament del servidor de la BOINC, en una de les màquines del laboratori Kleinrock, duent tot un seguit de proves amb l'aplicació de mostra anomenada *uppercase*. Un cop s'ha comprovat que el servidor de la BOINC està degudament configurat, s'ha pogut provar l'aplicació client del propi projecte i corregir-ne les errades trobades, com s'ha vist a l'apartat 6.4. Finalment, s'han desenvolupat les aplicacions per validar els resultats i assimilar-los a la base de dades i s'han provat amb l'aplicació del pseudo test. També com a aplicació final, s'ha desenvolupat un programa per crear automàticament paquets d'unitats de treball.

Tot i que no estava entre els objectius inicials del projecte, com es pot deduir, no s'ha aconseguit trencat el criptosistema. Ni tant sols s'ha arribat a provar un nombre de claus significatiu, ja que no s'ha disposat de la potència de càlcul de projectes com el de *distribuited.net* (que tampoc ha aconseguit trencar el criptosistema en el temps que porta funcionant) o SETI@home. L'objectiu principal d'aquest projecte era posar en funcionament la infraestructura necessària per possibilitar el trencament de la clau, i aquest ha estat aconseguit.

El segon dels objectius d'aquest projecte és fer una estimació del temps necessari per aconseguir trencar el criptosistema. Aquesta estimació es força difícil d'aconseguir, ja que els possibles recursos són molt heterogenis. Per exemple, com s'ha vist, no és el mateix tenir una màquina amb un sistema operatiu Windows que una altra amb GNU/Linux, ja que, com s'ha pogut veure en els temps de càlcul, a causa de les optimitzacions dels compiladors corresponents, el temps de processat de les 2^{32} claus difereix molt. Tot i aquestes dificultats, s'ha donat un temps aproximat per un PC amb dos nuclis amb un sistema operatiu GNU/Linux. D'aquesta manera es pot estimar que si, per exemple, es tingués un nombre de màquines similar al de SETI@home, unes 850.000, això implica una potència de càlcul de 5.100.000.000 MIPS i d'aquesta manera es necessitarien 287 anys aproximadament per aconseguir trencar el criptosistema proposat per RSA Laboratories. Amb aquests càlculs es pot concloure que la seguretat de l'algorisme RC5 amb paràmetres *w=32, r=12* i *b=9* és molt alta.

L'últim dels objectius era el d'aconseguir que el màxim de gent s'afegís al projecte i col·laborés, cedint voluntàriament els seus recursos computacionals per intentar provar quantes més claus fos possible. Aquest objectiu no ha estat assolit, ja que ha faltat força temps al final, per poder invitar a la gent a afegir-s'hi. Es necessitava un període de prova del servidor funcionant amb un conjunt de màquines controlades pel propi projecte per veure si aquest respon bé i els seus components interactuen degudament. Aquest període necessari tot just conclou just amb la finalització d'aquest projecte. Per tant no es poden fer mesures de temps de càlcul en un nombre de màquines diferents més gran.

7.1. Possibles millores del projecte

A continuació presentem un seguit de modificacions del projecte que permetrien la seva millora però que no s'han pogut implementar per restriccions temporals.

En primer lloc, un dels punts a millorar de la present implementació del projecte fa referència a la creació d'unitats de treball. Tal i com s'ha desenvolupat el sistema, actualment no poden coexistir a la vegada dos paquets d'unitats de treball, ja que l'assimilador, al rebre un resultat i assimilar-lo, sempre l'adjudica a l'últim dels paquets creats. A més a més, pot deixar sense feina momentàniament als participants Per tant, una millora en aquest punt consisteix en poder tenir diversos paquets d'unitats de treball coexistint a la vegada, de manera que es puguin provar diversos rangs de claus. Aquesta millora possibilitaria no haver de provar seqüencialment tot l'espai de claus. A més a més, es pot millorar la creació de tasques de manera que a l'acabar-se un paquet de treball, automàticament l'assimilador en creés un altre paquet.

Una altra possible millora fa referència a la reutilització de tota la infraestructura per a altres aplicacions criptogràfiques. És a dir, en comptes de realitzar les aplicacions de forma tan específica pel projecte basades únicament en l'algorisme RC5-32-12-9, es podria haver dissenyat l'aplicació client, i els dimonis verificador i assimilador de manera que acceptessin qualsevol de les modalitats del desafiament de RSA Laboratories. Aquesta millora seria realment útil si el temps per trencar el criptosistema de les diferents competicions, a partir de 72 bits de clau, no fos tant elevat.

La última de les millores que es proposen, i la més innovadora de totes, és la d'adaptar l'aplicació client per aprofitar la potència de càlcul de la nova PlayStation® 3. La idea, descartada per falta de temps, consisteix en instal·lar a la PlayStation® 3 el sistema operatiu Fedora Core 6 i el client de la BOINC. D'aquesta manera l'aplicació client podria aprofitar la potència d'aquesta màquina al màxim, és a dir, utilitzar els 8 nuclis dels que disposa el CELL[38]. Com s'ha vist a l'apartat 3.3, la PlayStation® 3 ha donat al projecte Folding@home una gran potència de càlcul extra.

[38] Un PPE i set SPEs. Es pot consultar més informació a http://en.wikipedia.org/wiki/Cell_microprocessor.

Concloent, aquest ha estat un projecte força llarg de dur a terme, ja que, tan l'algorisme de xifrat RC5 com la plataforma de càlcul distribuït BOINC eren noves per l'alumne i s'ha necessitat estudiar-ne molt els detalls. Sobretot en el cas de la BOINC i en general el model de càlcul distribuït que implementa, basat en PRC. Tot i les dificultats trobades, el projecte ha estat molt profitós i, com en tot projecte informàtic amb un temps limitat, hom es queda amb moltes idees i millores per dur a terme.

8. Bibliografia

[1] Criptografia de Wikipèdia: http://es.wikipedia.org/wiki/Criptografía [Data de consulta: 7 de juny de 2007]

[2] Protocol criptogràfic de Wikipedia: http://en.wikipedia.org/wiki/Cryptographic_protocol [Data de consulta: 7 de juny de 2007]

[3] Internet Security Glossary: http://tools.ietf.org/html/rfc2828 [Data de consulta: 7 de juny de 2007]

[4] Modes d'operació en xifrador de bloc de Wikipèdia: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation [Data de consulta: 7 de juny de 2007]

[5] R.L. Rivest, "The RC5 Encryption Algorithm", MIT Laboratory for Computer Science, 1997.

[6] A. Kerckhoffs, "La cryptographie militaire", Journal des sciences militaires, vol. IX, pp. 5–83, Jan. 1883, pp. 161–191, Feb. 1883.

[7] J. Carretero Pérez, P. De Miguel Anasagasti, "Sistemas operativos: Una visión aplicada", McGraw Hill, p. 562, 2001.

[8] Distributed computing de Wikipèdia: http://en.wikipedia.org/wiki/Distributed_computing [Data de consulta: 7 de juny de 2007]

[9] Public Resource Computing de Wikipèdia: http://en.wikipedia.org/wiki/Public_Resource_Computing [Data de consulta: 7 de juny de 2007]

[10] Distributed Computing Environment de Wikipèdia: http://en.wikipedia.org/wiki/Distributed_computing_environment [Data de consulta: 6 de juny de 2007]

[11] Globus Toolkit de Wikipèdia: http://en.wikipedia.org/wiki/Globus_Toolkit [Data de consulta: 6 de juny de 2007]

[12] Condor de Wikipèdia: http://en.wikipedia.org/wiki/Condor_cycle_scavenger [Data de consulta: 6 de juny de 2007]

[13] Frontier® Grid Platform de Parabon Computation: http://www.parabon.com/ [Data de consulta: 6 de juny de 2007]

[14] Grid MP de Wikipèdia: http://en.wikipedia.org/wiki/Grid_MP [Data de consulta: 6 de juny de 2007]

[15] Google Toolbar de Wikipèdia: http://en.wikipedia.org/wiki/Google_Toolbar [Data de consulta: 22 de maig de 2007]

[16] Potència de càlcul de Wikipèdia: http://en.wikipedia.org/wiki/List_of_distributed_computing_projects [Data de consulta: 6 de juny de 2007]

[17] Distributed.net de Wikipèdia: http://en.wikipedia.org/wiki/Distributed.net [Data de consulta: 6 de juny de 2007]

[18] C.U. SØttrup, J.G. Pedersen, "Developing Distributed Computing Solutions Combining Grid Computing and Public Computing", M.Sc. Thesis, Department of Computer Science - University of Copenhagen, 2005.

[19] Plataformes en BOINC de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/BoincPlatforms [Data de consulta: 7 de juny de 2007]

[20] Unitats de treball de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/JobIn [Data de consulta: 7 de juny de 2007]
[21] Resultats de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/JobOut [Data de consulta: 7 de juny de 2007]
[22] Redundància i errors de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/JobReplication [Data de consulta: 7 de juny de 2007]
[23] Distribució de tasques de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/WorkDistribution [Data de consulta: 7 de juny de 2007]
[24] Prerequisits de software de la BOINC de Creating BOINC Projects: http://boinc.berkeley.edu/trac/wiki/SoftwarePrereqsUnix [Data de consulta: 9 de juny de 2007]

Annexes

A. Planificació del projecte

A continuació es mostra la planificació inicial del projecte i tot seguit el calendari real:

Planificació inicial

Pla de treball

Tasques:

- Tasca 1: Estudi de l'algorisme de xifrat RC5 i els modes de xifrat en bloc (posant èmfasi en el CBC).
- Tasca 2: Entendre com funciona la Berkeley Open Infrastructure for Network Computing, els conceptes bàsics (projects, applications, workunits, results, etc.).
- Tasca 3: Implementar o obtenir una implementació de l'RC5 amb el mode de xifrat CBC i entendre-la per estudiar-ne una millora de l'eficiència (adaptada a les necessitats del PFC).
- Tasca 4: Comprobar que la implementació funciona i calcular els temps de processat (en PCs amb diferents configuracions) per fer un estudi del nombre de claus que podria rebre com a client, l'estratègia de les proves en base al text xifrat i el temps necessari en funció dels ordinadors utilitzats.
- Tasca 5: Identificar cada un dels conceptes bàsics de la BOINC amb el nostre PFC concret i els requisits específics del PFC que no formen part de la BOINC.
- Tasca 6: Estudiar i definir les necessitats del servidor pel que fa a la informació que ha d'emmagatzemar. A més a més, estudiar la manera d'empaquetar la informació per enviar al client.
- Tasca 7: Buscar un ordinador i montar, configurar i posar-hi en marxa el servidor de la BOINC.
- Tasca 8: Adaptar la implementació del RC5 amb mode de xifrat CBC a l'aplicació client de la BOINC i testejar-ne el correcte funcionament.

- Tasca 9: Fer un seguiment del procés de trencament del criptosistema i solucionar qualsevol possible mal funcionament.
- Tasca 10: A partir de tota la documentació que s'haurà anat generant durant les tasques anteriors elaborar la memòria final del PFC.

Durada de les tasques (en setmanes)[39]:

Tasques	*T1*	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*	*T8*	*T9*	*T10*
Durada	2	2	3	2-3	3-4	1-2	4	2	5	3

Taula 8: Durada inicial prevista de les tasques del projecte

Paquets de treball:

- Estudis sobre el projecte: tasques 1, 2 i 4,
- Infraestructura del servidor: tasques 5, 6, 7 i 9.
- Infraestructura del client: tasques 3 i 8.
- Documentació: tasca 10 (les altres tasques tindran una part de documentació associada).

Temporització:

Primer semestre:

	Octubre	Novembre				Desembre				Gener					Febrer			
Setmanes	30	6	13	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26
Tasca 1																		
Tasca 2																		
Tasca 3																		
Tasca 4																		
Tasca 5																		
...																		
Tasca 10																		

Taula 9: Planificació inicial prevista de les tasques del projecte (primer semestre)

[39] Els dos valors en algunes tasques signifiquen que es deixa un marge de temps. En el diagrama de Gantt que es mostra a continuació els marge es mostra en color roig.

Segon semestre:

	Març				Abril				Maig				
Setmanes	5	12	19	26	2	9	16	23	1	7	14	21	28
...													
Tasca 6													
Tasca 7													
Tasca 8													
Tasca 9													
Tasca 10													

Taula 10: Planificació inicial prevista de les tasques del projecte (segon semestre)

Calendari real

Pla de treball

Tasques:

- Tasca 1: Estudi de l'algorisme de xifrat RC5 i els modes de xifrat en bloc (posant èmfasi en el CBC).
- Tasca 2: Entendre com funciona la Berkeley Open Infrastructure for Network Computing, els conceptes bàsics (*projects, applications, workunits, results*, etc
- Tasca 3: Implementar o obtenir una implementació de l'RC5 amb el mode de xifrat CBC i entendre-la per estudiar-ne una millora de l'eficiència (adaptada a les necessitats del PFC).
- Tasca 4: Comprovar que la implementació funciona i calcular els temps de processat (en PCs amb diferents configuracions) per fer un estudi del nombre de claus que podria rebre com a client, l'estratègia de les proves en base al text xifrat i el temps necessari en funció dels ordinadors utilitzats
- Tasca 5: Identificar cada un dels conceptes bàsics de la BOINC amb el nostre PFC concret i els requisits específics del PFC que no formen part de la BOINC.
- Tasca 6: Estudiar i definir les necessitats del servidor pel que fa a la informació que ha d'emmagatzemar. A més a més, estudiar la manera d'empaquetar la informació per enviar al client.
- Tasca 7: Buscar un ordinador per muntar, configurar i posar-hi en marxa el servidor de la BOINC.
- Tasca 8: Adaptar la implementació del RC5 amb mode de xifrat CBC a l'aplicació client de la BOINC i testejar-ne el correcte funcionament.
- Tasca 9: Fer un seguiment del procés de trencament del criptosistema i solucionar qualsevol possible mal funcionament.
- Tasca 10: A partir de tota la documentació que s'haurà anat generant durant les tasques anteriors elaborar la memòria final del PFC.
- Tasca 11: Implementar els dimonis assimilador i verificador i un generador de tasques automàtic.
- Tasca 12: Adaptar la pàgina web del projecte que ve per defecte a les necessitats del projecte.

- Tasca 13: Dur a terme idees de millora que sorgeixin durant la realització del projecte[40].

Paquets de treball:

- Estudis sobre el projecte: tasques 1, 2 i 4,
- Infraestructura del servidor: tasques 5, 6, 7, 9, 11, 12 i 13.
- Infraestructura del client: tasques 3 i 8.
- Documentació: tasca 10 (les altres tasques tindran una part de documentació associada).

Temporització:

Primer semestre:

	Octubre	Novembre				Desembre				Gener					Febrer			
Setmanes	30	6	13	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26
Tasca 1																		
Tasca 2																		
Tasca 3																		
Tasca 4																		
Tasca 5																		
Tasca 6																		
Tasca 7									1	1							1	1
Tasca 8																		
Tasca 9																		
Tasca 10																		
Tasca 11																		
Tasca 12																		
Tasca 13																		

Taula 11: Calendari real de les tasques del projecte (primer semestre)

[40] No es fa una planificació específica per aquesta tasca

Segon semestre:

	Març				Abril				Maig				
Setmanes	5	12	19	26	2	9	16	23	1	7	14	21	28
...													
Tasca 5													
Tasca 6													
Tasca 7													
Tasca 8													
Tasca 9													
Tasca 10													
Tasca 11													
Tasca 12													
Tasca 13													

Taula 12: Calendari real de les tasques del projecte (segon semestre)

B. Exemple del procés de xifrat i desxifrat en RC5

A continuació es detallen exemples sobre les operacions bàsiques de l'RC5 i sobre el procés de xifrat i desxifrat sobre un RC5-32/12/9. S'utilitza també la clau de xifrat *D6 3C 96 E2 FF 1A BE A8 A2*.

Operacions bàsiques:

Operació	***A***	***B***	***R***
+	*01011010*	*00101110*	*10001000*
-	*01011010*	*00101110*	*00101100*
⊕	*01011010*	*00101110*	*01110100*
<<<	*01011010*	*3 bits*	*11010010*
>>>	*01011010*	*3 bits*	*01001011*

Taula 13: Operacions bàsiques de l'algorisme RC5 i exemples

Expansió de la clau:

Els valors de les constants màgiques per una mida de paraula *w* de 16, 32 i 64 bits tenen els següents valors en hexadecimal:

w	***Valor hexadecimal***	
	P_w	Q_w
16	*B7 E1*	*9E 37*
32	*B7 E1 51 63*	*9E 37 79 B9*
64	*B7 E1 51 62 8A ED 2A 6B*	*9E 37 79 B9 7F 4A 7C 15*

Taula 14: Càlcul dels valors de les constants màgiques de l'algorisme RC5

Convertir la clau secreta de bytes a paraules:

Es té *K[0] = D6, K[1] = 3C, ..., K[8] = A2.* Els valors de *u* i *c* són *4* i *3* respectivament. Els valors de la taula *L* són els següents:

L[0]	*E2 96 3C D6*
L[1]	*A8 BE 1A FF*
L[2]	*00 00 00 A2*

Taula 15: Valors de la taula *L* per un exemple de xifrat en RC5

Inicialitzar el vector *S*:

S[0]	*B7E15163*	*S[13]*	*C0B27FC8*
S[1]	*5618CB1C*	*S[14]*	*5EE9F981*
S[2]	*F45044D5*	*S[15]*	*FD21733A*
S[3]	*9287BE8E*	*S[16]*	*9B58ECF3*
S[4]	*30BF3847*	*S[17]*	*399066AC*
S[5]	*CEF6B200*	*S[18]*	*D7C7E065*
S[6]	*6D2E2BB9*	*S[19]*	*75FF5A1E*
S[7]	*B65A572*	*S[20]*	*1436D3D7*
S[8]	*A99D1F2B*	*S[21]*	*B26E4D90*
S[9]	*47D498E4*	*S[22]*	*50A5C749*
S[10]	*E60C129D*	*S[23]*	*EEDD4102*
S[11]	*84438C56*	*S[24]*	*8D14BABB*
S[12]	*227B060F*	*S[25]*	*2B4C3474*

Taula 16: Valors del vector *S* per un exemple de xifrat en RC5

Introducció de la clau *K*. Es mostren per algunes iteracions el valor corresponent de *A* i *B* (o *S[i]* i *L[j]*):

k	***i***	***j***	***A = S[i]***	***B = L[j]***
0	*0*	*0*	*BF0A8B1D*	*743418FE*
1	*1*	*1*	*4ABB79BC*	*E59EB6B6*
2	*2*	*2*	*2553AA39*	*30C88579*
3	*3*	*0*	*451F7207*	*EA1C107E*
...	...	...	...	...
25	*25*	*1*	*4B836925*	*52C7C76B*
26	*0*	*2*	*EAADDD6A*	*3D08003*
27	*1*	*0*	*C9CEB949*	*6D9A1532*
...	...	...	...	...
75	*23*	*0*	*AFBF6A69*	*A95AE9DF*
76	*24*	*1*	*BCDC4A74*	*D01F7D07*
77	*25*	*2*	*41AD0542*	*6A22C1C1*

Taula 17: Valors per algunes iteracions de la introducció de la clau de xifrat *K*

Xifrat:

S'han de xifrar dues paraules de 32 bits, 8 bytes en total. Es suposa el text *"provapfc"* que passat a ASCII i agrupat en dues paraules de 32 bits en hexadecimal queda com *0x70726F76 i 0x61706663.*

A la següent taula es mostren per cada passada (round) del xifrat els valors de *A* i *B*:

r	***A***	***B***
0	*70726F76*	*61706663*
1	*348EC1EC*	*D61D8453*
2	*78000314*	*8431B2F1*
3	*4B8B94A0*	*3B96D46A*
4	*448B189D*	*9E8C6ACE*
5	*CC003B5D*	*1A55E3E3*
6	*1DDF848C*	*B7DD89BE*
7	*9FAD8A7B*	*E857660F*
8	*7700FC74*	*A5E82DE9*
9	*74C642D3*	*99BB46CE*
10	*3F80A53*	*51634D57*
11	*2852000B*	*3A2A4E32*
12	*F5C09455*	*8EA7029D*

Taula 18: Valors de les iteracions pel xifrat del text "*provapfc*"

Finalment els valors de *A* i *B* xifrats són els de la última iteració.

Desxifrat:

r	***A***	***B***
12	*F5C09455*	*8EA7029D*
11	*2852000B*	*3A2A4E32*
10	*3F80A53*	*51634D57*
9	*74C642D3*	*99BB46CE*
8	*7700FC74*	*A5E82DE9*
7	*9FAD8A7B*	*E857660F*
6	*1DDF848C*	*B7DD89BE*
5	*CC003B5D*	*1A55E3E3*

4	*448B189D*	*9E8C6ACE*
3	*4B8B94A0*	*3B96D46A*
2	*78000314*	*8431B2F1*
1	*348EC1EC*	*D61D8453*
0	*70726F76*	*61706663*

Taula 19: Valors de les iteracions del desxifrat del text xifrat *F5C09455 8EA7029D*

Es pot veure que, com és lògic, el procés de desxifrat ens dóna els mateixos valors que el procés de xifrat però en ordre invertit. Els valors en la última iteració ($r = 0$) són els valors originals que s'han xifrat.

C. Procés de creació del projecte

Instal·lació del software de la BOINC

Servidor:

Per a la instal·lació del servidor s'utilitza el sistema operatiu GNU/Linux, intentant tenir una distribució força recent, ja que es necessiten certes llibreries software instal·lades. Primer, s'han d'acomplir els prerequisits de software, o sigui, les llibreries que s'han esmentat, entre altres. Els requisits són els següent:

- Les eines GNU següents: *make 3.79+, m4 1.4+, libtool 1.4+, pkg-config 0.15+, autoconf 2.58+, automake 1.8+* i *GCC 3.0.4+,*
- *Python 2.2+*, el mòdul *MySQLdb 0.9.2+* i el mòdul *xml* per *Python*,
- *MySQL 4.0+ o 4.1+* i el client de *MySQL*,
- *Apache* amb *mod_ssl* i *PHP 4.0.6+*,
- *OpenSSL 0.9.8+*,
- i *libcurl 7.15.5*.

També es necessita tenir instal·lat el sistema de control de versions SVN (el client) per descarregar-se el codi font del servidor de la BOINC. Per fer-ho s'executa l'ordre:

```
$svn co http://boinc.berkeley.edu/svn/trunk/boinc
```

I per descarregar-se addicionalment els fitxers d'exemple s'executa l'ordre:

```
$svn co http://boinc.berkeley.edu/svn/trunk/boinc_samples
```

Havent seguit aquests dos passos es disposa al directori local, entre altres, del codi font del servidor i del client (aquest és opcional d'instal·lar en l'ordinador servidor), les APIs, la documentació i una interfície web molt bàsica pel projecte.

Seguidament, s'ha de compilar tot el codi font necessari, que inclou els dimonis i aplicacions del servidor i les llibreries que aquests utilitzen:

```
$./_autosetup
$./configure --disable-client
$make
```

Pel correcte funcionament del servidor, s'ha de comprovar que la memòria compartida estigui activada i tingui un segment de dades màxim de com a mínim 32 megabytes (tot i dependre del sistema operatiu).

També s'ha de configurar bé la base de dades MySQL per permetre a l'Apache (que se sol executar com usuari *nobody*) accedir a la base de dades. Per fer-ho s'executaren les següents comandes:

```
$mysql -u root
    grant all on *.* to yourname@localhost identified by 'password';
    grant all on *.* to yourname identified by 'password';
    grant all on *.* to nobody@localhost identified by 'password';
    grant all on *.* to nobody identified by 'password';
```

També s'ha de modificar l'arxiu *httpd.conf* del servidor web Apache per ficar el tipus per defecte MIME de la següent manera:

```
DefaultType application/octet-stream
```

De PHP s'ha de configurar per habilitat les *magic quotes* al fitxer *php.ini* com s'indica a continuació:

```
magic_quotes_gpc = On
```

Client:

Segons el sistema operatiu utilitzat pel voluntari, es necessitaren certs prerequisits de llibreries o no. Si es vol instal·lar el client de la BOINC en Windows, tan sols cal descarregar-se el client de la pàgina web oficial i instal·lar-lo seguint els passos necessaris indicats per l'instal·lador.

En el cas de GNU/Linux, per instal·lar el Core Client i el BOINC Manager es necessitaren, a part de les eines GNU anteriorment indicades i les llibreries *OpenSSL* i *libcurl*, els *WxWidgets 2.6.3* i les llibreries *jpeglib* per *X11*.

Creació del projecte i posada en funcionament del mateix:

Un cop instal·lat el servidor de la BOINC i de que el MySQL està funcionant correctament s'han seguit les següents passes per crear el projecte RC5[41]:

- Situar-se al directori on es disposa dels fonts de la BOINC.
- Executar l'script *make_project* tal com segueix:

```
$./tools/make_project
      -project_root           $HOME/boinc/rc5
      -url_base         http://ccd-pr7.uab.cat/
      --delete_prev_inst
      --drop_db_first
      rc5
```

- Afegir la configuració necessària a l'Apache per donar accés a la pàgina del projecte:

```
$cat rc5.httpd.conf >> /etc/apache/httpd.conf
```

- Copiar el fitxer *project.xml* al directori del projecte i editar-lo per reflectir la realitat del projecte (p.e. les aplicacions i plataformes). Executar la comanda *bin/xadd* per reflectir els canvis a la base de dades.
- Editar el fitxer *html/project/project.inc* per canviar la *URL master* i el posseïdor del copyright.
- Protegir el directori *html/ops* mitjançant els fitxers *.htaccess* i *.htpasswd* per a l'Apache.
- Modificar el fitxer *html/user/schedulers.txt* per afegir-hi *http://ccd-pr7.uab.cat/rc5_cgi/cgi* entre *tags* <*scheduler*> </*scheduler*> (fer-ho en una sola línia per evitar problemes).

[41] Per crear un projecte esquelet es poden seguir les instruccions de http://boinc.berkeley.edu/project_cookbook.php

- Canviar l'usuari i grup del directori del projecte pels de l'administrador del servidor de la BOINC, per exemple, usuari *boincadm* i grup *boinc*. També cal canviar els permisos, ja que sinó l'Apache no pot funcionar degudament amb la pàgina web del projecte.

```
$chown boincadm:boinc rc5/ -R
$chmod 777 rc5/ -R
```

- Permetre la creació de comptes editant el fitxer *config.xml* i ficant un 0 al *tag disable_account_creation*.
- Seguir editant aquest fitxer, afegir una nova tasca amb el següent codi:

```
<task>
      <cmd> cat /dev/null </cmd>
      <output> log_USER/feeder.log </output>
      <period> 1 month </period>
</task>
```

- Ara es modifica el fitxer *config.xml* per canviar els paràmetres amb els que es criden els *daemons*, i reduir-ne la freqüència d'escriptura al *log* al mínim. S'ha de canviar el paràmetre *–d* amb valor 1.
- Modificar els paràmetres del fitxer *$HOME/html/project/project.inc* anomenats *PROJECT*, *COPYRIGHT_HOLDER* i *SYS_ADMIN_EMAIL* amb els paràmetres corresponents.
- També pot ser interessant modificar altres paràmetres d'altres fitxers d'aquest directori.
- Per afegir funcionalitat al menú del client de la BOINC es pot afegir el fitxer *gui_urls.xml* al directori principal del fitxer i editar-lo[42] per afegir els enllaços desitjats.
- Es pot modificar el fitxer *html/user/server_status.php* per afegir tots els dimonis i la base de dades al llistat de programes.
- Es pot afegir la línia *$project_has_beta = true;* al fitxer *html/project/project_specific_prefs.inc* i a l'afegir aplicacions de test (beta), afegir la línia *<allow_beta_work>1</allow_beta_work>* al fitxer *project.xml*.

[42] Es pot veure la sintaxis d'aquest fitxer a la següent pàgina web:
http://boinc-wiki.ath.cx/index.php?title=Add_Project-Specific_Links_to_the_Client_GUI

- Afegir els dimonis de validació (*validator*) i d'assimilació (*assimilator*) a l'arxiu *config.xml*. Prèviament s'han creat, compilat i copiat aquests dimonis al subdirectori *bin*.

Per comprovar el correcte funcionament del projecte es pot afegir una de les aplicacions de prova. S'ha triat l'aplicació *uppercase* que donat un fitxer d'entrada amb un text retorna un fitxer de sortida amb el mateix text en majúscules.

- Normalment s'afegiria l'aplicació al fitxer *project.xml*, però en aquest cas ja ve per defecte.
- Copiar l'aplicació del directori amb els fonts del servidor de la BOINC al directori *apps* del projecte.
- Crear-se els fitxers *xml* que defineixin la unitat de treball i el resultat a partir de les plantilles del directori corresponent. Els podem guardar a un directori nou anomenat *workunits*.
- Executar el programa per afegir noves unitats de treball:

```
./bin/create_work –appname uppercase –wu_name ucwu_2 –
wu_template workunits/uc_workunit.xml –result_template
workunits/uc_result.xml in
```

Amb aquests passos es té un resultat llest per enviar-se a qualsevol client Linux.

D. Informe dels resultats de les proves realitzades

Per la primera etapa del projecte, corresponent a la implementació de l'algorisme RC5, les proves realitzades han consistit en tres aplicacions:

- *rc5ValidaTest*
- *rc5TestClaus*
- *rc5BOINC*

La primera, *rc5ValidaTest*, consisteix en comprovar que, pel pseudo test d'RSA Laboratories, es desxifra correctament el text. La sortida d'aquest programa, usant la clau correcta és la següent:

```
Aplicació RC5-32-12-9 optimitzada pel Challenge (RC5-ValidaTest)
Carreguem la clau de xifrat:
C9 C 3 53 C0 D4 E1 FE 85
Convertint la clau de bytes a paraules...
L: 53030CC9 FEE1D4C0 85
Inicialitzant l'array S
Introduint la clau K
Vectors d'inicialització: ivA: 1F59CE07 - ivB: 419A1486

Text original:
20656854    6E6B6E75    206E776F    7373656D
20656761    203A7369    20656854    2D354352
312F3233    2D392F32    74736574    6E6F6320
74736574    70207327    6E69616C    74786574
08080808    08080808

Text xifrat:
562D285A    2FB7852A    BA30AEEF    15DA5493
1A79BAAC    850ECBA1    A531C39A    E3851936
7DDDF77     EDFA3B34    90418484    176F7B9D
B2EA512D    1C455E1C    DE346AE9    56B9F5A5
60E21622    B27EE115

Text desxifrat:
20656854    6E6B6E75    206E776F    7373656D
20656761    203A7369    20656854    2D354352
312F3233    2D392F32    74736574    6E6F6320
74736574    70207327    6E69616C    74786574
08080808    08080808

Comparem els textos...
Tot correcte
```

La segona, com s'ha comentat a l'apartat 5.1, és la més interessant de les tres, ja que s'obtenen temps d'execució per diverses mides de rangs de claus. La taula 20 mostra una comparativa de temps amb diverses màquines.

Dades de la màquina	Temps
Dades de l'ordinador: • Sistema Operatiu: Gentoo Linux • Arquitectura: Intel 32 bits • Tipus Placa Base: 845 • Tipus de Processador: Pentium 4 • Velocitat del Processador: 2.266 GHz • Memòria RAM: 512 Mb Altres dades: • Us d l'ordinador durant el test: Normal	2^{20} claus: 3s 2^{25} claus: 76s 2^{30} claus: 2.802s 2^{32} claus: (sense dades)
Dades de l'ordinador: • Sistema Operatiu: Ubuntu LiveCD • Arquitectura: AMD 32 bits • Tipus Placa Base: K7 • Tipus de Processador: Duron • Velocitat del Processador: 1.8 GHz • Memòria RAM: 1024 Mb Altres dades: • Us d l'ordinador durant el test: Nul	2^{20} claus: 2s 2^{25} claus: 52s 2^{30} claus: 1895s 2^{32} claus: 6.482s
Dades de l'ordinador: • Sistema Operatiu: Windows XP • Arquitectura: AMD 32 bits • Tipus Placa Base: K7 • Tipus de Processador: Duron • Velocitat del Processador: 1.8 GHz • Memòria RAM: 1024 Mb Altres dades: • Us d l'ordinador durant el test: Nul	2^{20} claus: 2s 2^{25} claus: 61s 2^{30} claus: 5.815s 2^{32} claus: ****7.761s
Dades de l'ordinador: • Sistema Operatiu: Windows XP • Arquitectura: AMD 64 bits • Tipus Placa Base: K8 • Tipus de Processador: Athlon X2 3600+ • Velocitat del Processador: 2.01 GHz • Memòria RAM: 1024 Mb Altres dades: • Us d l'ordinador durant el test: Normal	2^{20} claus: 5s 2^{25} claus: 158s 2^{30} claus: 5.063s 2^{32} claus: (sense dades)
Dades de l'ordinador: • Sistema Operatiu: Fedora	2^{20} claus: 1s 2^{25} claus: 39s

• Arquitectura: Intel 32 bits • Tipus Placa Base: (desconegut) • Tipus de Processador: Pentium 4 (x2) • Velocitat del Processador: 3.0 GHz • Memòria RAM: 1024 Mb Altres dades: • Us d l'ordinador durant el test: Nul	2^{30} claus: (desconegut) 2^{32} claus: 3.381s
Dades de l'ordinador: • Sistema Operatiu: Windows XP • Arquitectura: Intel 32 bits • Tipus Placa Base: (desconegut) • Tipus de Processador: Pentium 4 • Velocitat del Processador: 2.4 GHz • Memòria RAM: 256 Mb Altres dades: • Us d l'ordinador durant el test: (desconegut)	2^{20} claus: 3s 2^{25} claus: 88s 2^{30} claus: (sense dades) 2^{32} claus: 11.642s
Dades de l'ordinador: • Sistema Operatiu: Linux 2.6.15 • Arquitectura: Intel 32 bits • Tipus Placa Base: (desconegut) • Tipus de Processador: Xeon • Velocitat del Processador: 3.2 GHz • Memòria RAM: 4 Gb Altres dades: • Us d l'ordinador durant el test: (desconegut)	2^{20} claus: 3s 2^{25} claus: 97s 2^{30} claus: (sense dades) 2^{32} claus: 12.403s
Dades de l'ordinador: • Sistema Operatiu: Linux 2.6.17 • Arquitectura: Intel 32 bits • Tipus Placa Base: (desconegut) • Tipus de Processador: Xeon • Velocitat del Processador: 3.2 GHz • Memòria RAM: 2 Gb Altres dades: Us d l'ordinador durant el test: (desconegut)	2^{20} claus: 1s 2^{25} claus: 37s 2^{30} claus: (sense dades) 2^{32} claus: 4.698s

Taula 20: Temps d'execució per diversos rangs de claus en diverses màquines

L'última de les aplicacions, *rc5BOINC,* tan sols difereix de la primera en seguir l'estructura d'una aplicació de la BOINC, però sense utilitzar les APIs d'aquesta. Per tant, no se'n mostren resultats.

E. Informe de l'evolució del desafiament

A data d'impressió d'aquesta memòria s'han utilitzat les màquines del laboratori Kleinrock per comprovar el correcte funcionament de la infraestructura i la interacció de tots els components que la formen. Del conjunt de màquines del laboratori s'ha instal·lat el client de la BOINC en 9 d'aquestes. S'han mantingut executant unitats de treball durant les nits i els caps de setmana, per no molestar als altres projectistes que utilitzen aquestes màquines.

La Taula 21 mostra l'evolució, en nombre d'unitats de treball processades, durant el període que transcorre entre els dies 3 de maig de 2007 i 29 de maig de 2007.

Data	Nombre d'unitats de treball processades
2007-05-03	9
2007-05-04	67
2007-05-05	146
2007-05-06	212
2007-05-07	256
2007-05-08	272
2007-05-09	352
2007-05-10	435
2007-05-11	507
2007-05-12	617
2007-05-13	711
2007-05-14	736
2007-05-15	825
2007-05-16	914
2007-05-17	998
2007-05-18	1091
2007-05-19	1257
2007-05-20	1416
2007-05-21	1432
2007-05-22	1509
2007-05-23	1586
2007-05-24	1635
2007-05-25	1704
2007-05-26	1752
2007-05-27	1779
2007-05-28	1789
2007-05-29	1791

Taula 21: Nombre d'unitats de treball processades pel període del 3 al 29 de maig de 2007

El primer paquet de treball consistia en 256 claus, per fer les primeres proves de funcionament i es va acabar de processar per les màquines client el dia 7 de maig de 2007. El següent paquet de treball, un cop comprovat que tot funciona degudament, consisteix en 4096 unitats de treball.

F. Contingut del CD-ROM

Juntament amb aquesta memòria s'entrega un CD-ROM amb tot el necessari per poder posar en funcionament aquest projecte en qualsevol màquina amb un sistema operatiu GNU/Linux. Cal tenir en compte, però, que les dependències software necessàries han de ser instal·lades de forma independent.

A continuació es detalla el contingut del CD-ROM, en directoris, indicant, sobre l'estructura d'un projecte BOINC, on s'ubica cada part del contingut.

- Proves inicials: codi font i executables per les proves de la primera fase (no calen pel projecte BOINC).
- Aplicacions client: codi font i executables de les aplicacions client de la BOINC (s'ubiquen en subdirectoris del *apps*).
- Aplicacions servidor: codi font i executables de les aplicacions servidor específiques del projecte (s'ubiquen al directori *bin*).
- Modificacions de la base de dades: *scripts* per la creació de les taules específiques del projecte a la base de dades.
- Arxius de configuració: els arxius *config.xml* i *project.xml* del projecte i la configuració del servidoor Apache, ubicats al directori principal.
- Web del projecte: fitxers amb la funcionalitat web afegida pel projecte, que s'ubiquen al directori *html*.
- Plantilles: fitxers *xml* amb les plantilles de les unitats de treball i resultats (no són necessaris).
- Altres: altres arxius utilitzats (no necessaris per la BOINC).

També s'hi pot trobar aquesta memòria en versió digital.

A partir de les indicacions de l'annex C i el contingut del CD-ROM es pot posar en funcionament el servidor de la BOINC del projecte. Cal indicar, però, que no s'adjunten les claus per a la signatura digital del projecte, per raons de seguretat, i que per tant, s'han de generar de nou amb la utilitat *crypt_prog* subministrada per la BOINC. Tampoc s'adjunta una còpia de la base de dades ja que aquesta es va modificant cada dia i immediatament es quedaria desactualitzada.

Signatura de l'autor:

David Cervelló Batlle

Aquest projecte té com a objectiu participar en el desafiament d'RSA Laboratories corresponent a trencar el criptosistema RC5-32-12-9 proposat. Per realitzar-ho s'ha triat realitzar un atac per força bruta, mitjançant el càlcul distribuït i, més concretament, utilitzant la Public Resource Computing. La plataforma escollida és la Berkeley Open Infrastructure for Network Computing (BOINC), coneguda per la seva utilització en grans projectes com ara SETI@home. En aquest projecte es posa en funcionament la infraestructura i es desenvolupen les aplicacions necessàries per iniciar els càlculs que haurien de permetre el trencament del criptosistema.

Este proyecto tiene como objetivo en participar en el desafío de RSA Laboratories correspondiente a romper el criptosistema RC5-32-12-9 propuesto. Para realizar-lo se ha escogido realizar un ataque por fuerza bruta, usando el càlculo distribuido i, más concretamente, la Public Resource Computing. La plataforma escogida es la Berkeley Open Infrastructure for Network Computing (BOINC), conocida por ser usada en grandes proyectos como SETI@home. El este proyecto se pone en funcionamiento la infraestructura y se desarrollan las aplicaciones necesarias para iniciar los cálculos que deberían permitir romper el criptosistema.

The main objective of this project is to participate in the challenge of RSA Laboratories corresponding to break the cryptosystem RC5-32-12-9. The strategy used to achieve this goal is to use a brute force attack using distributed computing techniques and, more precisely, Public Resource Computing. The chosen platform is the Berkeley Open Infrastructure for Network Computing (BOINC), a well known infrastructure since it has been used in large projects like SETI@home. In this project the BOINC infrastructure has been configured and it has been developed all necessary applications in order to start the computation that should allow breaking the proposed cryptosystem.