Instituto Superior Manuel Teixeira Gomes

Engenharia Informática | 3º Ano

Ano Lectivo 2015/2016

Relatório Final

Computação Distribuída

Brute Force MD5 Distribuído

Docente:

Mestre Francisco Melo Pereira

Discentes:

João Vieira N.21500414

Rui Gambóias N.21204323

Introdução

Este documento engloba o relatório final relativo ao projeto académico de Computação Distribuída.
O projeto tem como base a aplicação de conhecimentos práticos, técnicos, teóricos e capacidades adquiridas.
A sua incidência baseia-se na proposta previamente apresentada de uma aplicação distribuída, de criptografia, para a descoberta de chaves MD5, através de técnicas "BruteForce".

Sobre o MD5

O MD5 (Message-Digest algorithm 5) é uma função de dispersão criptográfica (ou função hash criptográfica) de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na RFC 1321, é utilizado por softwares com protocolo ponto-a-ponto (P2P) na verificação de integridade de arquivos, logins ou passwords.

Esta técnica é usada para deteção de modificações em ficheiros e (logo) assinaturas digitais, guardar senhas de forma segura e outras funções criptográficas.

Uma inevitabilidade do MD5 é que se ele gera um valor de dispersão de 128 bits, então só consegue dar valores de dispersão diferentes a 2_{128} combinações diferentes de dados. Inevitavelmente, alguns dados diferentes irão gerar o mesmo valor de dispersão. A falha de deteção de modificações em ficheiros nesses casos é uma limitação aceite nestes algoritmos. Eles são no entanto construídos de forma a que:

1. Não seja possível prever qual a modificação que leva a um valor de dispersão idêntico, e
2. As modificações necessárias sejam muito grandes, tornando a fraude por falsificação inviável.

Devido a estas considerações, só é possível encontrar exemplos de 2 sequências de dados que geram o mesmo valor do MD5 por força bruta, ou seja, experimentando

sequencialmente (ou aleatoriamente) sequências diferentes de dados até que uma gere um valor de dispersão idêntico. Como a quantidade de combinações possível é muito grande esta exploração irá demorar muito tempo pelo que deve ser feita em paralelo por vários processadores e computadores.

Objetivos

Criação de uma aplicação distribuída que descobre chaves MD5 duplicadas para um qualquer hash de entrada. Esta aplicação irá ser dividida em dois módulos com interface de consola:

1. Módulo de interface com o utilizador (Master) , que divide o problema da pesquisa por tantos computadores quantos estiverem disponíveis, fazendo a gestão d as explorações em cada computador, recebendo ao mesmo tempo relatórios de progresso de cada um para exibir ao utilizador.
2. Módulo(s) de exploração de dados (Slave). Estas aplicações devem estar dormentes até serem “acordadas” pela aplicação de interface. O pedido de exploração deve incluir o hash em causa e o “ponto” de onde deve começar (e até onde ir) a exploração. A ligação é mantida aberta e a aplicação deve regularmente indicar o progresso a quem fez o pedido.

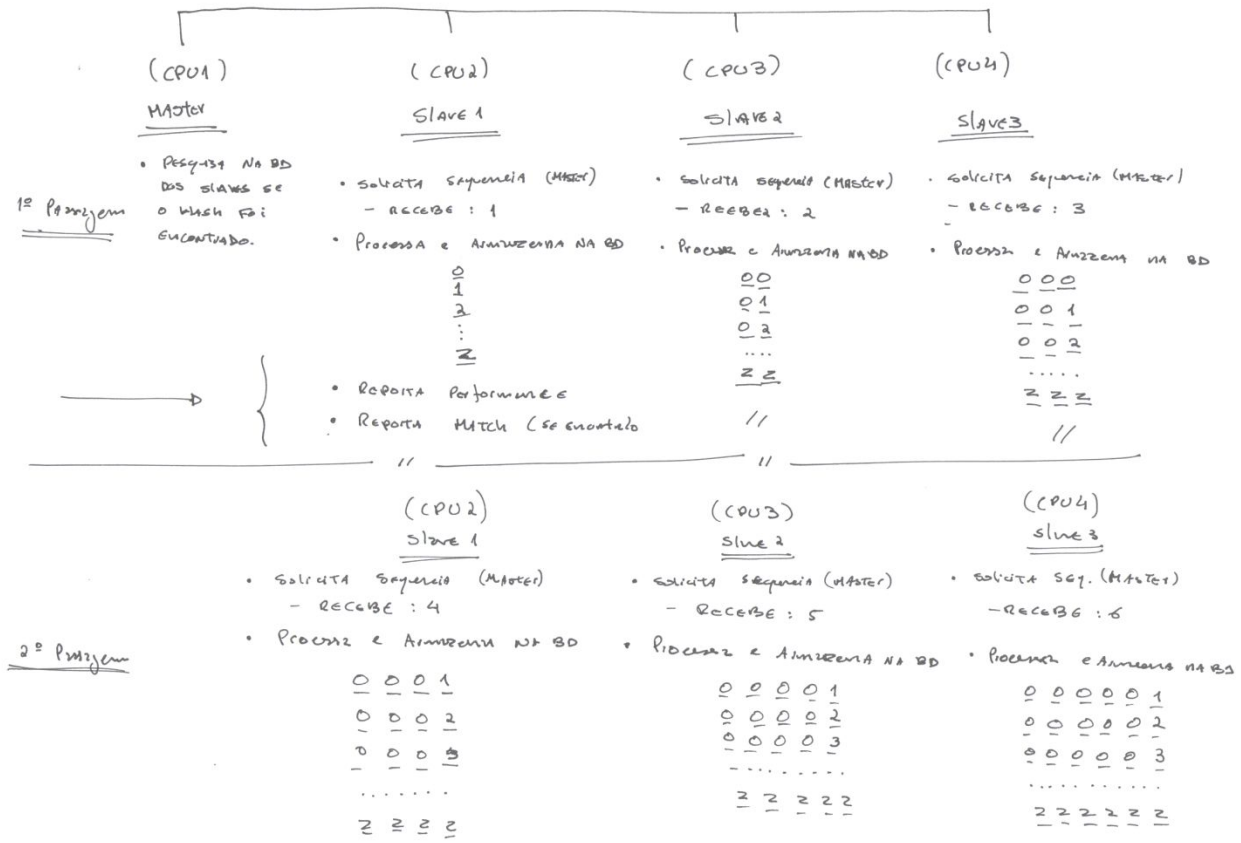
O módulo Slave guarda em base de dados os respetivos hash gerados, para que futuras passagens sejam feitas mais rapidamente.

O módulo Master deverá reportar ao utilizador o estado dos Slaves bem como do numero de iterações e hash em base dados existentes, deverá também fornecer algum tipo de estatística sobre a performance dos módulos Slave.

O módulo Master faz a deteção automática de slaves no mesmo segmento de rede.

Modelação Funcional - (Final)

Planeamento funcional dos módulos:



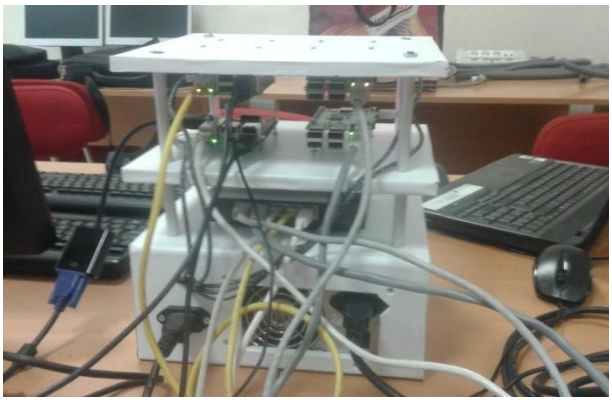
Hardware

De modo a acomodar as necessidades / características do projeto foi utilizado o seguinte hardware:

1) 1 fonte de alimentação
2) 1 raspberry pi2
3) 3 raspberry pi3
4) 1 switch
5) Cabos diversos

Foi construído em madeira uma estrutura de apoio ao projeto em forma de torre com 3 níveis distintos, alimentação, networking e processamento. Deste modo agrupa-se todo o hardware necessário numa estrutura sólida e robusta de fácil transporte.






Funcionalidades

A nível funcional, ambas as aplicações, Slave e Master, limitam-se á geração e gestão de “Hash”’s.

Modulo Slave

O modulo Slave apresenta sobretudo funcionalidades relacionadas com a gestão do processo de geração de Hash’s tais como funcionalidades que instruem o “Engine” sobre qual o MD5 a procurar, numero de sequencia a ser gerada, bem como a gestão da base de dados para guardar, procurar ou eliminar entradas de Hash. É ainda feita a gestão do processo do Engine (thread), tais como shutdown do Engine, ou simplesmente o “quit” da ligação a que o master se encontra ligado.

É ainda feita a autenticação através de strings ao ligar á socket deste módulo, que são providenciadas pelo modulo que se liga, ou seja pelo Master, o Slave apenas aceita comunicação caso o cliente esteja devidamente autenticado.

Modulo Master

O módulo Master é responsável pela deteção e gestão de módulos Slave que encontrar no mesmo segmento de rede.
Para isso recorre aos dispositivos de rede que encontra na máquina, solicitando ao utilizador o dispositivo que este pretende fazer “bind” ou seja, que pretende usar como ponto interface.

É detetado o IP da interface e são realizadas tentativas de ligação de socket na porta 3000 em todos os IP’s dentro do segmento de rede, após deteção de todos os Slaves, é feita a autenticação nos mesmos e uma vez ligados, cabe a este módulo a sua gestão individual.

Comandos

Modulo Master

O módulo Master aceita entradas de comando no software, este não possui sockets de escuta telnet, todas as configurações são feitas no próprio engine.

//Comando shutdown

Se o comando escrito na linha de comandos for “shutdown”, é executada a função shutDownAgents() que se encarrega de enviar o sinal de shutdown aos agentes, esperando pelo fim da thread de cada Slave, finalmente termina o processo Master encerrando assim o cluster.

//Comando toggletable

Caso o comando escrito na linha de comandos seja “toggletable” é executada a função toggleTable() e é apresentada a tabela de estados e estatísticas individuais para cada Slave ligado ao módulo Master.

Modulo Slave

O módulo Slave aceita entradas de comando quando ligado por telnet na porta 3000, só é possível configurações através das sockets.

//Comando quit

Se o comando for “quit”, é chamada a função disconnect() que encerra a socket entre o cliente e servidor, contudo qualquer processamento que esteja a decorrer é mantido em execução.

//Comando shutdown

Se o comando for "shutdown", a flag de “shutdown” é inicializada a “TRUE” e o daemon do Slave desliga as ligações de socket e encerra o processo.

//Comando process

O comando “process” dá início ao processamento da sequência MD5.

//Comando setmd5

O comando “setmd5” espera o input do utilizador que deverá ser uma string de MD5.

//Comando setsequence

O comando “setsequence” é utilizado definir uma nova sequência de processamento.

//Comando usesql

O comando “usesql” espera um input de “Y” ou “N” do utilizador habilitando o uso da base de dados no processamento ou não.

//Comando truncate

O comando “truncate” apaga todas as entradas de MD5 na base de dados.

//Comando reset

O comando “reset” causa a paragem do processamento caso este esteja a ocorrer, e limpa as variáveis de “MD5” e “Sequence”, ficando a aguardar novos dados.

//Comando status

O comando “status” devolve uma string na ligação de socket com o estado atual do Slave (waitting4sec, running, iddle), este comando é usado pelo master para saber o estado individual de cada Slave.

//Comando goraw

O comando “goraw” é usado para não devolver outputs de confirmação na socket, por norma é evocado pelo Master antes de iniciar o processamento nos Slaves.

//Comando getmatch

O comando “getmatch” é utilizado para obter o resultado da chave MD5 após esta ser descoberta.

Implementação

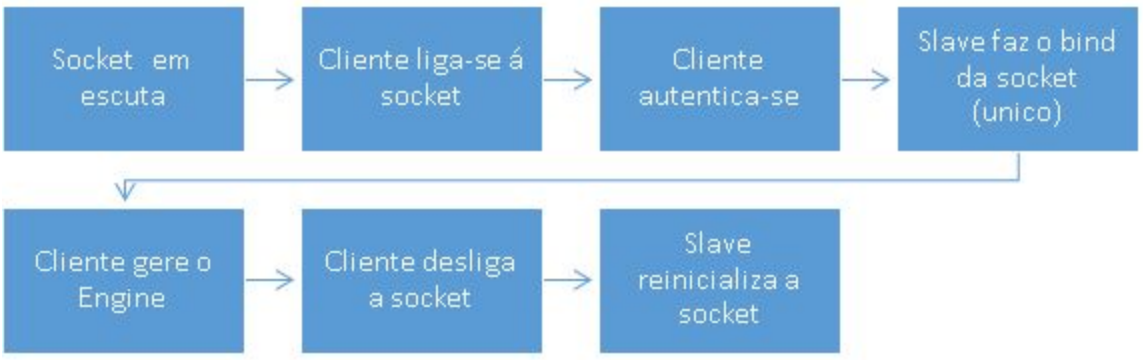
Desde a conceção da ideia, a implementação do software foi feita com recurso á programação em C++, linguagem de eleição por ser orientada a objetos e perto da máquina o que permite um maior desempenho por parte dos CPU’s.

Módulo Slave

A construção e planeamento do módulo Slave teve como base as seguintes diretrizes:

- Uso de sockets para que possa ser controlado remotamente.
- Segurança através de processo de autenticação.
- Pesquisa e arquivo de chaves MD5.
- Envio de estado em tempo real (monotorização).
- Possibilidade de configurar o Daemon como um serviço de maquina.

-Modelação funcional - Ligação ao Módulo Slave:



Em pormenor:

- È feito uma ligação TCP/IP através de socket (telnet, ex) á porta 3000 do IP do Slave que se encontra em estado listening.
- Assim que a ligação é estabelecida ao módulo Slave é realizada uma autenticação por parte do Slave que requisita ao cliente o input de um username e password. É dado ao cliente 3 possíveis tentativas, se após as mesmas este não autenticar os dados de input a socket é fechada pelo Slave e o cliente perde a ligação ao módulo.
- Caso o cliente autentique, o Slave faz o bind a socket ao IP prevenindo demais ligações enquanto esta se encontra ativa.
- O cliente pode agora enviar os comandos ao módulo para gestão do Engine do Slave.
- Assim que o mesmo perder a ligação ou evocar comandos como “shutdown” ou “quit”, este termina a socket e volta a reinicializar.

-Modelação funcional - Geração de HASH’s:

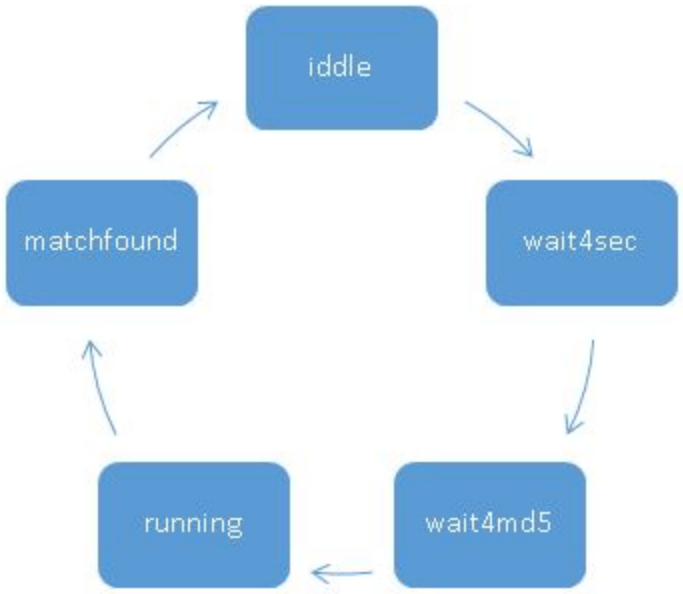
Em pormenor:

- Recebe nova sequência para processamento do cliente.
- Recebe o MD5 caso não esteja definido e faz uma query na BD com a string que recebeu, procurando assim entradas que possam corresponder, caso encontre o Engine entra em estado “matchfound”.
- Caso não encontre entradas para este MD5 o processamento é iniciado gerando a cada iteração um Hash de MD5.

- De modo a otimizar queries, são introduzidos registo de 50 Hash's em cada query, assim otimiza-se o número de queries / segundo que o servidor de MYSQL suporta.
- Quando o algoritmo de "Geração de Sequências" determinar que já esgotou todos as possíveis combinações para a sequência, o módulo entra em "wait4sec" aguardando nova sequencia para execução.

-Estados do Engine do Slave:

O Modulo master ao enviar o comando "status" para o Slave, este retorna 1 dos 5 possíveis estados:



Em pormenor:

Iddle - O Slave encontra-se parado, por norma verificável ao início do Daemon.

Wait4sec - O Slave encontra-se á espera de uma sequencia para processamento.

Wait4md5 - O Slave encontra-se á espera de um MD5 para processamento.

Running - O slave encontra-se em processamento e a gerar Hash's.

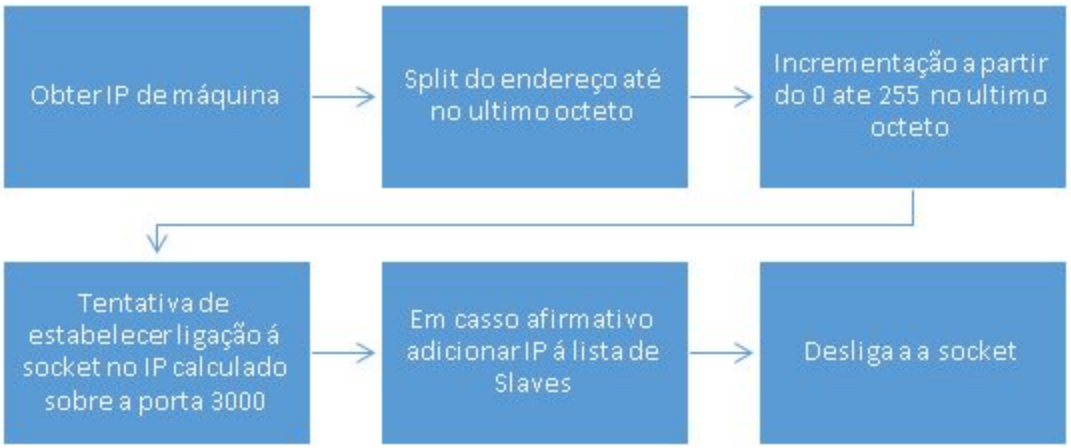
Matchfound - O slave reporta que encontrou um match para o MD5 de procura.

Modulo Master

A construção e planeamento do módulo Master teve como base as seguintes diretrizes:

- Uso de sockets para que possa controlar Slaves remotamente.
- Uso de threads para que possa gerir em simultâneo os Slaves.
- Interface de gestão para o utilizador.
- Monotorização do estado em tempo real.
- Descoberta de Nós (Slaves) automática na rede.

- Modelação funcional - Descoberta de Nós Slave:



Em pormenor:

- É solicitado ao utilizador a interface de máquina sobre o qual pretende fazer o lookup.
- O endereço dessa mesma interface é obtido, e é feito o “split” dos octetos do IP, de modo a limpar o ultimo octeto para que seja possível criar uma lista de todos os IPS na rede.
- Iterando pelas 255 possibilidades de uma netmask /24 é feita uma tentativa de ligação á socket na porta 3000, caso o Master consiga fazer a ligação, entende que a porta se encontra aberta e é adicionado o Slave á lista.

- Modelação funcional - Slave Thread:

Em pormenor:

De modo a que modulo Master suporte vários Slaves simultaneamente é necessário que este processe cada um dos mesmos numa “Thread” unica a cada Slave, isto porque seria impossível fazer a gestão de cada Slave iterativamente de uma forma eficiente.

Assim cada Slave possui a sua própria “Thread” que a cada 5 segundos faz a gestão e invoca os comandos necessários.

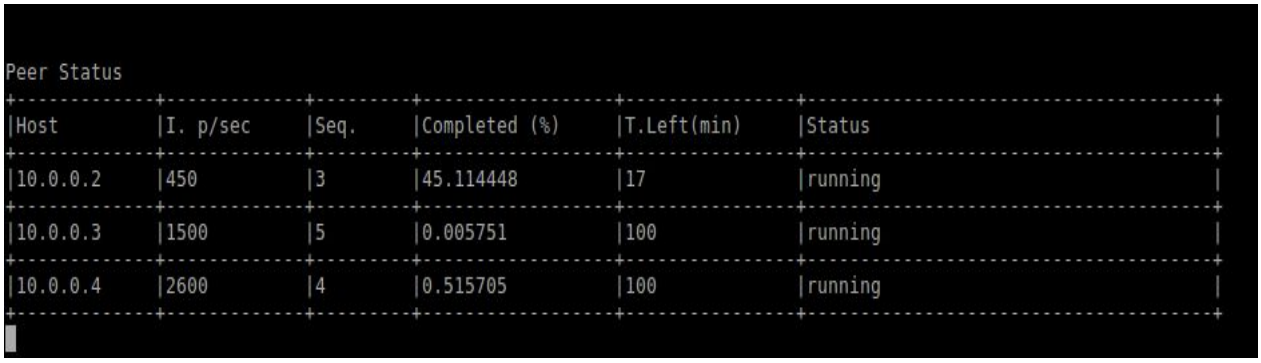
Em suma o processo da “Thread” é resumido da seguinte forma:

- Faz a autenticação do módulo Master caso este não esteja autenticado.
- Pergunta o status do mesmo, e responde consoante necessidade, por exemplo poderá enviar uma determinada sequência ou fazer “set” do MD5.
- Cada vez que é enviada uma determinada sequencia, o módulo Master cria uma entrada de base de dados com o IP e a sequencia que foi atribuída ao Slave, após este ter completado a sequência esta entrada na BD é alterada com uma Flag de sequencia completa. Este processo permite ao Master manter um registo das sequências que já foram ou estão a ser geradas pelos Slaves otimizando assim o processamento evitando que os Slaves processem

sequencias repetidas ou que continuem o processamento de determinada sequencia caso esta não esteja completa.
- Acompanha o processamento através do comando “status” e informa o utilizador.
- Em caso de “match”, pergunta ao Slave a string e informa utilizador.

- Monotorização:

A monotorização é feita em tempo real através de uma tabela visível na linha de comandos invocando o comando “toggletable”, sendo que utilizador terá acesso aos dados de cada nó, tais como numero de iterações por segundo, percentagem da sequencia concluída, tempo restante de processamento etc…



Algoritmo

-“Geração de Sequencias”

De modo a poder gerar um MD5 sobre determinada string é necessário calcular essa mesma string, sendo que o MD5 varia de acordo com o número de caracteres que a string possui ou “lenght” da mesma e os próprios caracteres é necessário calcular todas as hipóteses dentro de uma determinada “length” ou seja:

Por exemplo, uma string com “length” == 4, será composta por 4 letras:

Obtém-se assim __ . __ . __ .__ 4 “slots” possíveis a ser explorados, a isto denominamos de “Sequencia”.

Será então necessário desenvolver um algoritmo que explore todas as possíveis hipóteses para uma determinada sequencia sobre um determinado dicionário.

Entenda-se por dicionário todo um conjunto de caracteres tal como dic = { A, B, C, D, …. ,* ,/ }

O numero total de combinações possíveis sobre determinada sequencia é facilmente calculável através da formula { !sizeof(dic) *sequence }

Dependendo do tamanho da sequência e do dicionário, esta poderá gerar números de combinações relativamente grandes, como tal é necessário ter em atenção ao uso da memória ao desenvolver o algoritmo, sendo que técnicas recursivas não serão adequadas ao problema, como tal foi desenvolvido um algoritmo iterativo para o cálculo.

Código C++:

```
for(i =this->md5Sequence-1; i >= 0; i--)
{

        this->matrix[i]++;

        if(i==0 && this->matrix[i]>this->sizeOfDict-1)
        {
                this->waittingSequence = true;
                break;
        }

        if(this->matrix[i]<this->sizeOfDict)
                break;

        if(this->matrix[i]>=this->sizeOfDict)
               this->matrix[i]=0;

  }
```

Highlights (pontos altos)

Nos “highlights” do projeto destacamos:

- Possibilidade de interromper processo de geração de Hash’s e resumir a sequencia mais tarde.
- Possibilidade do utilizador desligar a base de dados para otimizar procuras.
- Facilmente escalável, basta adicionar módulos slave na rede.
- Algoritmo iterativo de processamento, permite o processamento praticamente infinito sem correr riscos de overflow de memória.

Ambientes de Desenvolvimento e OS Suportados

O software foi desenvolvido em ambiente linux usando o G++ para compilação.
Prevê-se que funcione na maioria das distros Linux, bastando que se faça a compilação na máquina.

Foi instalado o SO Ubuntu para Raspbery tanto nos módulos Master e Slave para acomodar as respetivas instalações.

Em ambos os lados foi instalado o MYSQL community server que serve as base de dados.

Limitações

Nas limitações de projeto, destacamos:

- Não consegue partir sequências, ou seja cabe ao Slave o processamento de toda a sequência até que esta chegue ao fim, poderá ser problemático em sequências grandes, pois o tempo de processamento tende a incrementar exponencialmente.

- Não deteta subnets de rede ao fazer lookup dos Slaves, esta funcionalidade não foi programada devido á sua complexidade, não se tendo achado relevante para a demonstração.

Dificuldades

Nos principais focos de dificuldade, destacamos:

Timeout nas sockets - Algum tempo de debug primeiro que se compreende-se o porque de o Master não conseguir fazer ligação á socket. Ao desenvolver primeiramente o master sobre ambiente “localhost” não nos apercebemos do timeout que deveria ser superior ou igual a pelo menos 5 segundos de modo a que a comunicação pela rede pudesse ser efetuada.

Complexidade do algoritmo - Foi sem duvida o ponto alto na implementação de todo o projeto e onde foram aplicadas mais horas em situações de (trial & error). Primeiramente pela dificuldade lógica e depois por não existir qualquer modelo iterativo facilmente consultável no mundo da WEB.

Conclusão

O relatório apresentado serve como complemento ao trabalho prático realizado, nele foram descritos os passos relevantes para a sua execução.

Todos os equipamentos e recursos foram selecionados e estudados de acordo com o projeto a ser desenvolvido, tendo como base a Unidade Curricular de Computação Distribuída.

O código implementado no trabalho surge de conhecimentos obtidos ao longo do curso.
O trabalho desenvolvido teve como objetivo final, a criação de uma aplicação distribuída, de criptografia, para a descoberta de chaves MD5, através de técnicas "BruteForce".