IEDCS

File Player With Integrated DRM

Segurança 2015/2016

Mestrado Integrado em Engenharia de Computadores e Telemática
David Silva 64152
Rui Ribeiro 68794

Índice

Fase 1

Problema

O objetivo deste trabalho é desenvolver um sistema DRM end-to-end - o Identify Enabled Distribution Control System ou IEDCS.

O IEDCS contempla a utilização de uma Device Key (relativa à maquina do cliente), uma Player Key (relativa à implementação do Player; disponível dos dois lados), uma User Key (relativa ao utilizador que está ligado ao sistema; apenas disponível do lado do servidor) e uma File Key (computada a partir das outras chaves e única para cada instância de compra de um ficheiro cifrado para uma máquina de um determinado utilizador).

Adicionalmente, espera-se que os ficheiros não estejam disponíveis no sistema de ficheiros e que haja restrições a nível do acesso ao seu conteúdo.

Abordagem

Arquitetura do serviço

A arquitectura sobre a qual nos baseámos para este trabalho está de facto muito próxima da inicialmente apresentada, sendo que a loja online é o único componente cuja implementação está prevista para apenas a segunda fase do projeto.

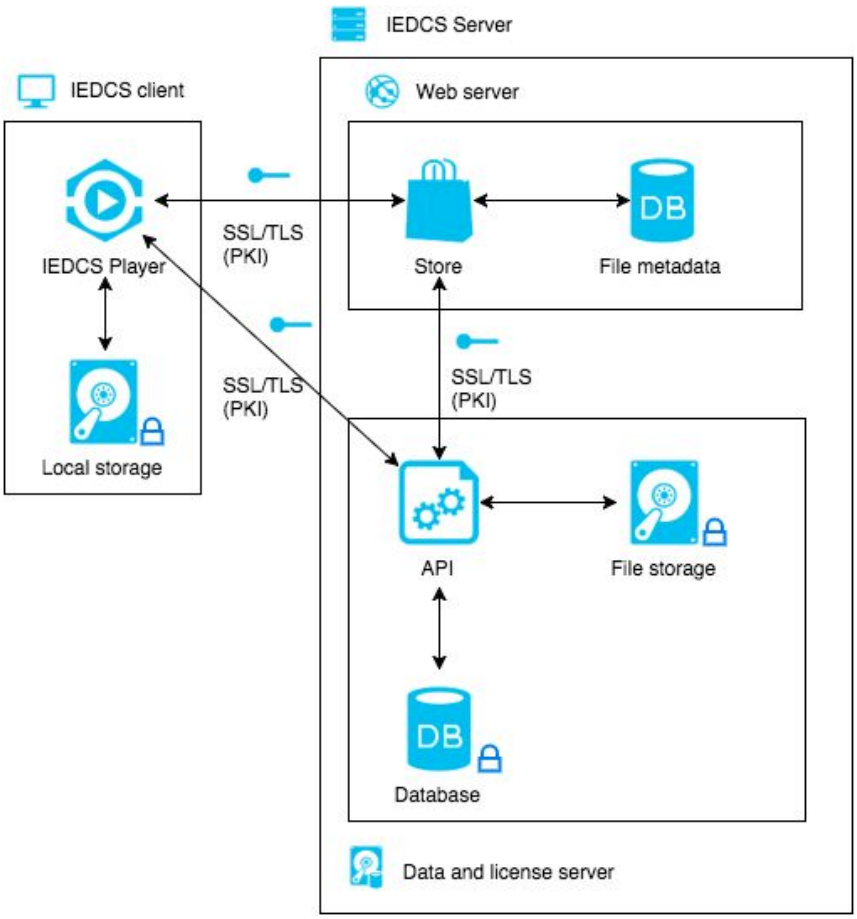


Imagem 1: Diagrama de componentes apresentado no dia 20 de Outubro.

Métodos de cifra

Optámos por utilizar chaves simétricas em todo o trabalho devido à facilidade que estas nos conferem aquando da dedução da File Key do lado do Cliente de forma segura.

O modo de cifra utilizado é o ECB na dedução da File Key (acreditamos que este modo é seguro o suficiente neste contexto, pois o secretismo da chave final prende-se com o próprio secretismo das chaves usadas) e o CBC na cifra do ficheiro (para obter uma cifra por blocos que utiliza um valor inicial, IV, no processo de cifra).

Adicionalmente, estamos a utilizar um túnel SSL para a comunicação entre o cliente e o servidor, de modo a minimizar o risco de ataques por Man-in-the-Middle (MITM) e DNS Spoofing.

Restrições de acesso a conteúdos

A nossa base de dados foi projetada com o objetivo de detetar momentos em que um mesmo utilizador inicia sessão num dispositivo diferente, gerando um ‘alias’ de identificação da instância do cliente. Esta abordagem permite-nos identificar quando um mesmo utilizador inicia sessão num computador diferente, o que nos permite limitar o número de dispositivos associados a um mesmo cliente. Para o sucesso desta implementação, pretendemos adicionar um conjunto de funções de autorização e desautorização de dispositivos, que será integrado na loja online e que apenas estará disponível na próxima fase do projeto.

Derivação das chaves iniciais

Player Key

A Player Key está hardcoded no lado do cliente na altura da instalação e configuração geral do Player. Esta chave é um segredo pré-partilhado de cada um dos lados (cliente/servidor).

Na nossa abordagem, a Player Key só é atribuída ao cliente no momento em que se verifica que o mesmo foi capaz de passar um desafio básico proposto pelo servidor aquando da primeira ligação. Este desafio consiste na combinação de uma autenticação básica HTTP, um User-Agent específico e um valor específico no campo ‘data’ do pedido HTTP enviado do cliente para o servidor.

Device Key

A Device Key é gerada no cliente através de uma função de síntese de um valor que resulta da junção do endereço MAC da máquina e de um outro valor, sendo este o número de série do processador (no caso do Linux), o número de série do computador (no caso do OS X) ou o número de série do disco rígido (no caso do Windows).

Pensámos inicialmente em usar apenas o endereço MAC, contudo este é facilmente forjável, comprometendo o conceito inteiro de uma chave única.

User Key

A User Key é gerada no servidor através de uma função de síntese de um valor aleatório gerado sempre que uma nova combinação de utilizador + dispositivo estabelece ligação com o servidor. Esta combinação é tratada como um alias e é enviada em todas as mensagens que se estabelecem entre o cliente e o servidor, conferindo um mecanismo de autenticação muito elementar, que será explorado de melhor forma no próximo trabalho.

Derivação da File Key

Um dos objetivos deste trabalho consistia em encontrar uma solução segura para computar a File Key através da combinação User Key, Device Key e Player Key, tanto no cliente como no servidor, assumindo que o cliente nunca conhece uma das chaves (User Key), que apenas está disponível do lado do servidor.

Servidor

Do lado do servidor, a File Key é obtida através de cifra encadeada, definida da seguinte forma:

1. É cifrado um valor inicial (IV) com a Device Key;
2. O resultado desta cifra é cifrado com a User Key;
3. A File Key é obtida cifrando com a Player Key o criptograma anterior.

Cliente

Do lado do cliente, a File Key não está imediatamente disponível, tendo de existir um processo de computação que requer auxilio por parte do servidor:

1. O cliente solicita ao servidor o IV mencionado anteriormente;
2. O cliente cifra o IV com a Device Key e envia o resultado para o servidor;
3. O servidor cifra esse resultado com a User Key e envia o novo criptograma de volta para o cliente;
4. Por fim, o cliente cifra este criptograma com a Player Key, obtendo a mesma File Key que foi gerada no servidor.

Armazenamento de informação

Cifra de ficheiros

Para além da cifra com a File Key, computada do lado do Player no momento da reprodução do ficheiro, conforme mencionado anteriormente, os ficheiros transferidos estão cifrados com AES CBC, onde os primeiros 16 bytes do ficheiro cifrado constituem o IV necessário para proceder à decifra do mesmo.

Ficheiros decifrados

Uma das preocupações que tivemos durante o desenvolvimento deste trabalho foi a garantia de que o conteúdo de cada ficheiro não estava exposto para leitura sem o IEDCS Player, do lado do cliente, evitando, assim, a quebra do propósito deste trabalho. Para garantir isso, o resultado de uma decifra com a File Key no lado do cliente não é em nenhuma circunstância armazenado em disco, ficando residente em memória como um buffer de bytes.

Bases de dados

Atendendo ao facto de que as chaves tinham de ser guardadas de forma segura numa base de dados, optámos por as cifrar com uma chave adicional a que chamámos de Master Key, que serve apenas como salvaguarda da segurança das chaves (podendo esta implementação ser mudada na segunda fase do trabalho em prol de uma mais segura e eficiente).

Por fim, uma das garantias que devemos ter é a de que o utilizador SQL relativo ao IEDCS Server não tem mais do que as permissões efetivamente necessárias para realizar o seu trabalho, evitando assim um comprometimento da informação presente nas bases de dados. Apesar de reconhecermos isto, nenhuma base de dados deste projeto está protegida com acesso por utilizador e palavra-passe devido ao facto de tal segurança não ser oferecida pelo SQLite. Num ambiente profissional de desenvolvimento, o SGBD deverá suportar este requisito.

Implementação

Ambiente de desenvolvimento

Todo o projeto foi desenvolvido dentro de ambientes virtuais do virtualenv, em Python. A escolha da linguagem prendeu-se com o facto de procurarmos uma linguagem de alto nível e de fácil manipulação. A escolha de ambientes virtuais prendeu-se com a intenção de permitir que as duas instâncias até agora desenvolvidas pudessem comunicar e correr sem causar um grande impacto na máquina do utilizador, algo que não seria tão eficiente com máquinas virtuais, por exemplo.

Ambientes virtuais

O cliente é composto por módulos em Python para a validação do Player (que neste caso é basicamente um teste de comunicação e a derivação da Device Key), ficheiros para a configuração do envio e recepção de mensagens com o servidor e um módulo de representação do .epub em texto simples numa interface gráfica rudimentar desenvolvida em Tkinter.

Por sua vez, o servidor é composto por uma aplicação desenvolvida em Flask (responsável pelo envio de dados provenientes ou processados pela API e o cliente) e uma base de dados SQLite (que é suficiente para a complexidade e volume da base de dados).

Prevenção de ataques

Durante o desenvolvimento desta solução, implementámos medidas de prevenção de ataques, nomeadamente ao nível da base de dados, onde as ações nunca são executadas como queries sem verificação. Quanto à prevenção de XSS, a mesma será tida em consideração aquando do desenvolvimento da loja, na próxima parte do trabalho, não havendo, por agora, hipóteses de existirem ataques deste tipo.

Quanto à hipótese de buffer overflow, a mesma não é considerável no contexto de programação em Python.

Compra e transferência de ficheiros

Para ter acesso a um ficheiro, o utilizador tem de o ter comprado previamente, caso contrário não é possível realizar a transferência do mesmo.

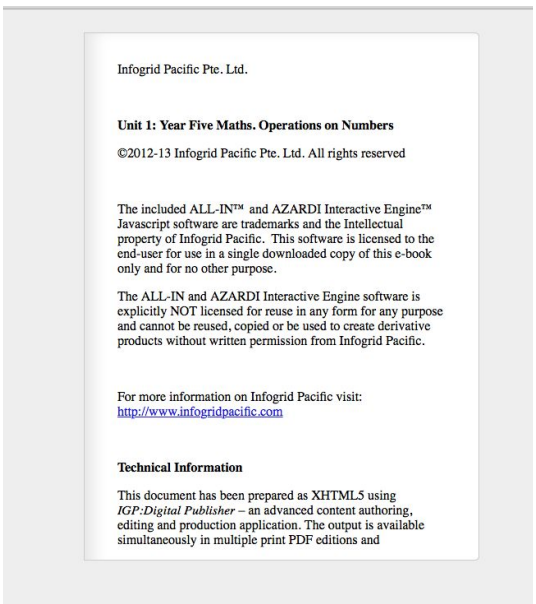
Nesta fase do trabalho, todas as compras e transferências estão a ser feitas sem qualquer restrição do número de compras por parte de um utilizador. Isso prendeu-se pelo facto de a loja online ainda não estar implementada, pelo que a invocação dos métodos é, por agora, válida para testar as File Keys.

Reprodução de ficheiros .epub

Inicialmente, apresentámos uma solução para a reprodução dos ficheiros .epub que consistia num plugin para o software Calibre, que é o software livre de referência para lidar com ebooks. Tal como previsto pelo professor na apresentação, fomos obrigados a mudar a abordagem, dado que a complexidade da implementação deste plugin iria ser demasiado trabalhosa em relação aos objetivos gerais da disciplina.

Seguidamente, procurámos uma abordagem que nos permitisse conseguir ler o ficheiro recorrendo ao Calibre, mas sem desenvolver qualquer plugin. Esta solução, que foi desaconselhada pelo professor, obrigava-nos a que o ficheiro estivesse disponível em disco para poder ser lido pelo programa, o que não era suposto.

Tentámos, também implementar um leitor de ebooks com um Web Server, do lado do cliente, recorrendo a uma biblioteca externa desenvolvida em Javascript que interpretava este tipo de ficheiros. Os resultados, apesar de graficamente satisfatórios, abriam precedência a que o ficheiro estivesse disponível para transferência através de um browser ou outra aplicação.



Infogrid Pacific Pte. Ltd.

Unit 1: Year Five Maths. Operations on Numbers

For more information on Infogrid Pacific visit:
http://www.infogridpacific.com

Technical Information

This document has been prepared as XHTML5 using *IGP:Digital Publisher* – an advanced content authoring, editing and production application. The output is available simultaneously in multiple print PDF editions and

Imagem 2: Leitura de um ficheiro .epub utilizando o leitor em Javascript

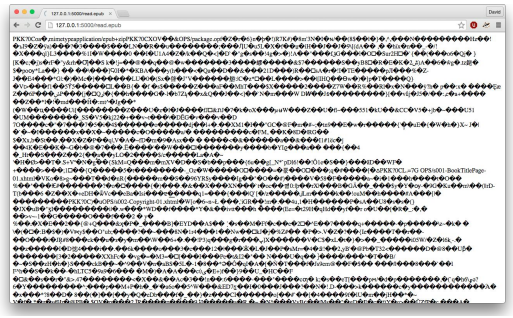


Imagem 3: Resultado de um acesso ao ficheiro decifrado utilizando um browser.

```
aluno-3346:server davidsilva$ wget http://127.0.0.1:5000/read.epub
--2015-11-16 18:23:08--  http://127.0.0.1:5000/read.epub
Connecting to 127.0.0.1:5000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 872978 (853K) [text/html]
Saving to: 'read.epub'

read.epub           100%[===================>] 852.52K  --.-KB/s   in 0.002s

2015-11-16 18:23:08 (384 MB/s) - 'read.epub' saved [872978/872978]

aluno-3346:server davidsilva$
```

Imagem 4: Resultado de uma tentativa de utilização do comando wget para transferência do ficheiro decifrado.

Finalmente, procurámos seguir a abordagem aconselhada pelo professor: reescever o método _load() da biblioteca ebooklib para que este pudesse aceitar um buffer de bytes (file-like object) ao invés de um caminho para um ficheiro. Neste sentido, utilizámos a biblioteca textract para juntamente com o novo método _load() fosse possível interpretar o .epub e apresentá-lo em texto simples. Como modo de visualização do texto, implementámos uma interface gráfica rudimentar em Tkinter.

Principais bibliotecas utilizadas

- PyCrypto - para as funções de síntese e cifradecifra
- ebooklib e textract - para parsing do texto de um ficheiro ePub
- Flask - para implementação da API e gestão dos pedidos ao servidor
- sqlite3 - para utilização de SQLite em Python
- SQLAlchemy - para abstração nas queries à base de dados

Discussão dos resultados obtidos

Conseguimos desenvolver um sistema de comunicação segura entre dois ambientes virtuais e conseguimos visualizar o conteúdo do ficheiro decifrado do lado do cliente sem comprometer o secretismo do ficheiro decifrado. O conteúdo do ficheiro é visualizado em texto simples, pois é o suficiente para a verificação de que o sistema DRM funciona corretamente.

Possíveis adições a fazer ao trabalho, caso não estivéssemos já a entregá-lo fora do prazo da entrega, seriam uma implementação de um túnel SSL com Diffie-Hellman incorporado como forma de antever a transferência do próprio Player para o cliente, protegendo a Player Key de um possível ataque por MITM, implementar uma interface gráfica de interação com o cliente e criar um mecanismo de melhor registo de compras na base de dados. Não tendo sido possível fazer isto nesta entrega, esperamos que alguns destes pontos sejam incluídos aquando da segunda fase de entrega do trabalho.

Fase 2

Contextualização

Dada a primeira fase de implementação, o sistema que tínhamos permitia a dedução da File Key tanto no servidor como no cliente sem que houvesse a transmissão desta chave. Nesta dedução estão envolvidos um número secreto aleatório único para cada instância que serve como vetor de inicialização da cifra por AES (CBC) de um ficheiro, a Device Key, a User Key e a Player Key. Após a dedução, é feito um download de um ficheiro cifrado que é reproduzido sem que ele nunca esteja em disco decifrado e de seguida o ficheiro é apagado, sendo este sistema, um sistema estável de mero *proof-of-concept.*

Melhorias na segunda fase

Para esta segunda fase, o grupo melhorou várias questões de implementação relativas à primeira fase (tanto funcionais como estéticas) e adicionou obviamente os requisitos pedidos para esta fase, nomeadamente a autenticação do utilizador no sistema, utilizando o Cartão de Cidadão, e a cifra de ficheiros em disco, de modo a evitar o comprometimento do servidor.

Nova arquitetura do serviço

Inicialmente, tínhamos planeado a implementação de uma loja online baseada num website. Essa ideia foi descartada mais tarde devido ao facto de não trazer mais segurança ao sistema. Assim, o novo diagrama da arquitetura do serviço é o seguinte:

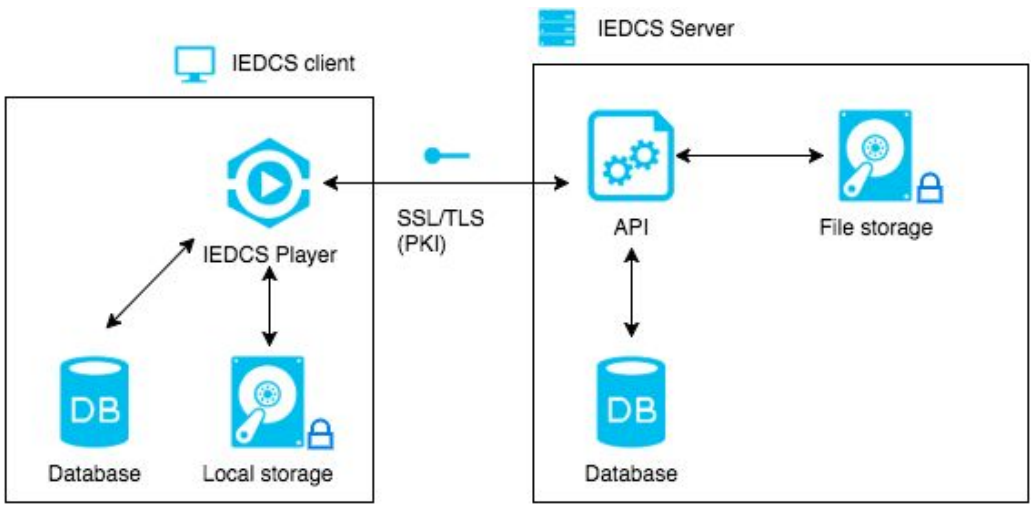


Imagem 5: Diagrama final da arquitetura do serviço

Existe agora uma base de dados do cliente que é responsável por guardar informação sobre os ficheiros que foram adquiridos naquela máquina e o utilizador que os adquiriu.

Proteção das chaves do servidor

De modo a proteger as chaves da base de dados do servidor, no caso de um comprometimento desta mesma, o grupo decidiu cifrá-las já na primeira entrega (embora não houvesse a certeza de que isso fosse um requisito da primeira fase ou da segunda; em todo o caso, o grupo entendeu que do ponto de vista de segurança este processo era de extrema relevância). Na primeira implementação, esta cifra é simétrica (AES), com uma *master_key* que está hardcoded no sistema.

Nesta segunda entrega, melhorámos este aspeto, lidando com esta questão com cifra assimétrica: utilizamos um par de chaves RSA, cuja chave privada está armazenada no servidor num ficheiro .pem protegido com palavra-passe e a chave pública está noutro ficheiro .pem. Cada chave (IV, Device Key, User Key, Player Key, File Key) ao ser criada/deduzida é cifrada com a chave pública do par e para a sua decifra é necessária a chave privada (no arranque do servidor é introduzida a palavra-passe para que este consiga aceder às chaves. Deste modo, o base de dados nunca tem informação relevante que comprometa o sistema DRM.

Restrição de dispositivos por utilizador

Por puro lapso, o grupo implementou aquando da primeira entrega, todas as funções para permitir que um utilizador com dispositivos diferentes consiga fazer o download de um ficheiro com a mesma File Key, mas não implementou efetivamente este processo.

Assim, o grupo colocou este processo em prática para a segunda entrega: existe uma restrição de 2 dispositivos por cada utilizador. Para que este utilizador tenha acesso ao mesmo ficheiro em 2 dispositivos diferentes cifrado com a mesma File Key, o processo de dedução da File Key do lado do servidor pressupõe a atribuição de um IV novo ao dispositivo novo do utilizador, decifrando a File Key original com a ajuda da Device Key nova em vez da antiga (sendo que o sistema previne que existam mais que 2 dispositivos associados a um utilizador na base de dados; é possível no entanto dissociar um dispositivo de um utilizador, apagando-o da base de dados, com o risco de todas as compras realizadas nesse dispositivo serem apagadas também se não forem transferidas para outro).

Restrição de utilizadores por dispositivos

O software desenvolvido permite a associação de mais do que um utilizador à mesma instância do programa. Esta solução não constitui um risco de segurança, pois os ficheiros de um utilizador apenas estão disponíveis para ele mesmo, não sendo possível ler livros adquiridos por outros utilizadores.

Criação de uma interface gráfica

Na primeira fase, o sistema lançava uma janela para a leitura do ebook. Agora, existe uma interface de utilização do sistema, que inclui uma biblioteca do utilizador (onde estão listados todos os livros que já foram transferidos) e uma loja, onde podemos comprar títulos e transferi-los (sendo que a segunda opção é apenas feita se o utilizador quiser). Esta interface desenvolvida em Tkinter permite que cada fase do processo (compra, download, leitura) seja escolhida pelo utilizador e não que seja feito tudo de seguida (primeira entrega) e que seja possível abrir um título à escolha e não um título hardcoded (primeira entrega; agora é utilizada uma base de dados do lado do cliente, que contém o nome do ficheiro cifrado e o seu título).

Verificação do sistema

No arranque do cliente, era feita uma verificação das configurações gerais. Esta configuração é agora realizada em cada troca de janela (de modo a prevenir situações de cartão de cidadão removido ou trocado) e inclui agora uma verificação da base de dados do cliente e da lista de ficheiros cifrados no sistema de ficheiro, confirmando se a lista de títulos na base de dados é coerente com a lista de ficheiros cifrados no sistema de ficheiros.

Autenticação com o Cartão de Cidadão

A autenticação era algo de rudimentar na primeira implementação, sendo apenas uma autenticação básica HTTP. Nesta segunda fase, a mesma autenticação foi mantida (não consideramos este fator problemático devido ao facto de estarmos numa ligação HTTPS).

Agora, a autenticação é feita com o Cartão de Cidadão, sendo feita uma assinatura com a chave privada do lado do cliente e uma verificação com a chave pública, armazenada na base de dados do do servidor. É necessário o pin de autenticação do cartão para as fases de registo de utilizador e de compra e leitura de um título.

Cifra de ficheiros em disco

De modo a proteger os ebooks decifrados no servidor, foi aplicado um dos mecanismos de cifra em disco abordados nos guiões das aulas práticas - File System Level. Assim, os ficheiros são guardados numa partição encfs que é montada no arranque do sistema e desmontada quando este encerra o funcionamento, de modo a que um utilizador que não o administrador não consiga abrir os ficheiros.

Modelo relacional da base de dados

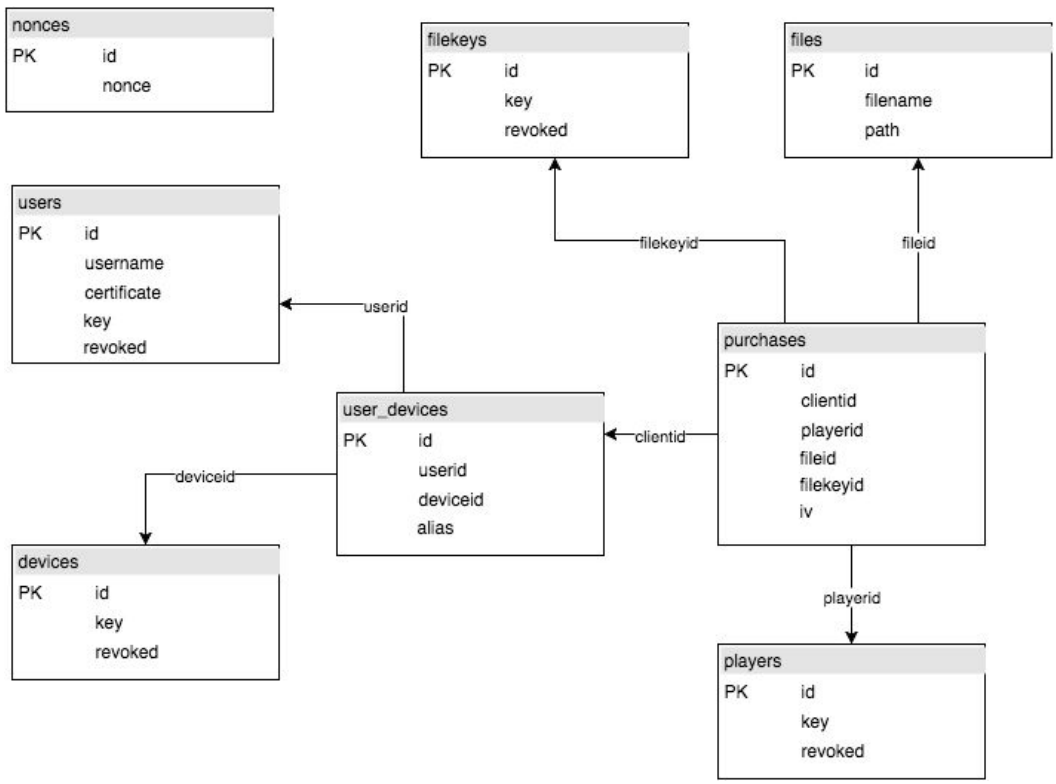


Imagem 6: Diagrama da base de dados do servidor

Não incluímos aqui o diagrama da base de dados do cliente, pois esta apenas contém informações sobre o nome do ficheiro no computador, o nome do ficheiro no servidor remoto e o utilizador que adquiriu aquele ficheiro.

Capturas de ecrã

Imagem 7: Janela inicial da aplicação

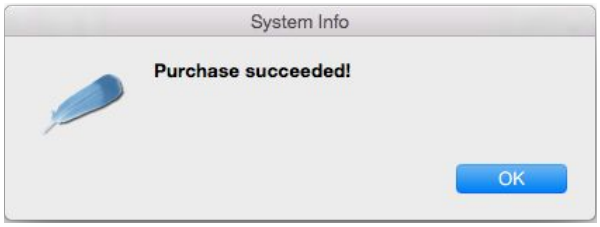
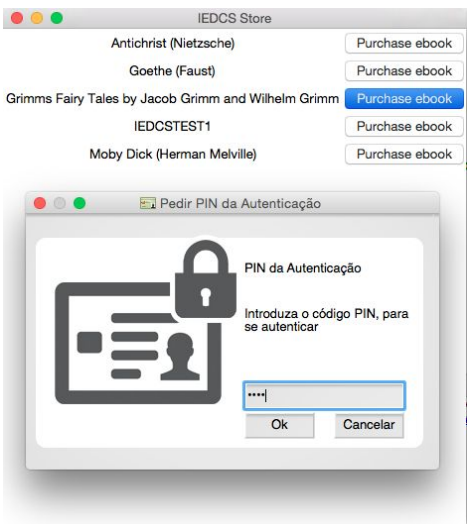
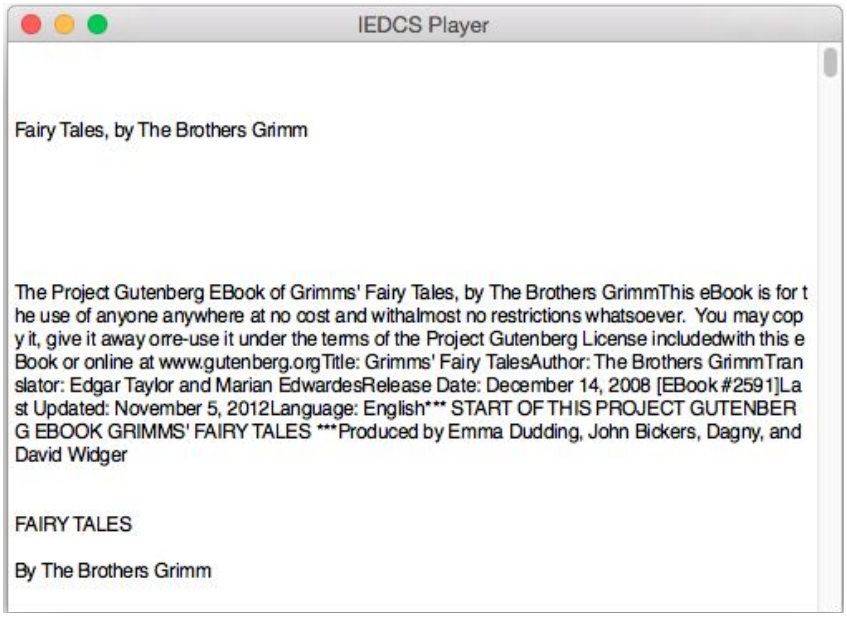


Imagem 8: Mensagem de sucesso na compra do ficheiro

Imagem 9: Mensagem de erro na comunicação com o servidor



Imagens 10 e 11: Pedido de autenticação para compra e leitura de ebook



Imagens 10 Leitor de ebooks

Bibliografia

Guia de utilização do Flask: http://flask.pocoo.org/
Biblioteca PyKCS11: http://pkcs11wrap.sourceforge.net/api/
Biblioteca cryptography: https://cryptography.io/en/latest/
Documentação oficial de bibliotecas do Python: https://docs.python.org/
Dúvidas gerais de implementação: https://stackoverflow.com
Guia de Introdução ao Tkinter: http://effbot.org/tkinterbook/
Guiões das aulas práticas
Outras referências especificadas no código