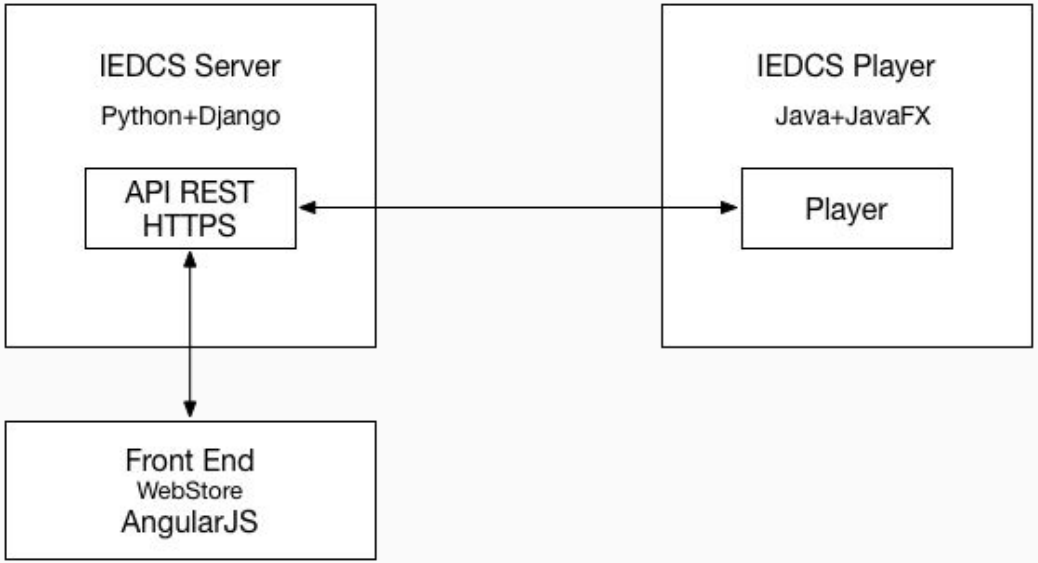
IEDCS: Identity Enabled Distribution Control System

Security 2015/2016
DETI, Universidade Aveiro

Rafael Ferreira, nmec: 67405
Rodrigo Cunha, nmec: 67800

20/10/2015

System Arquitecture



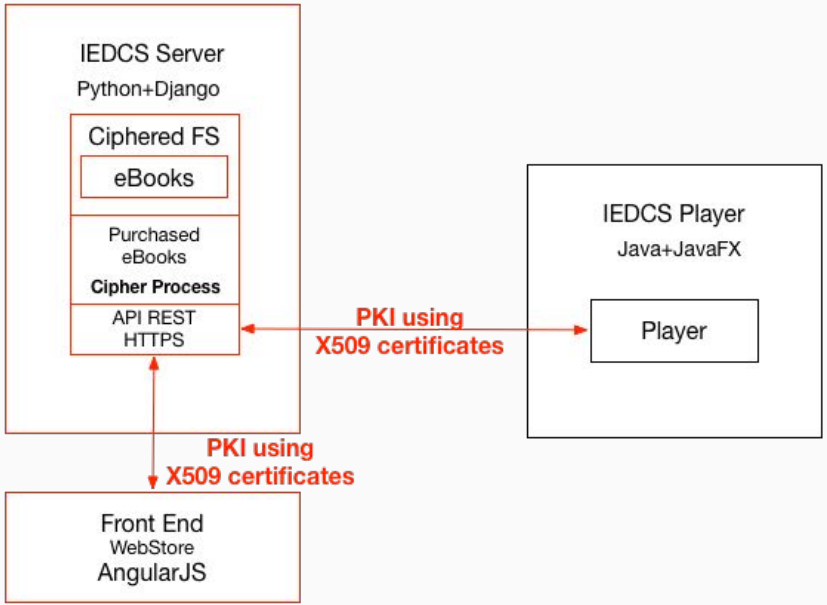
Technologies:

- **Python+Django**
- **Java+JavaFX**
- **AngularJS**

Communication channels:

- **API REST**
- **PKI with X509**
- **HTTPS**

IEDCS Server



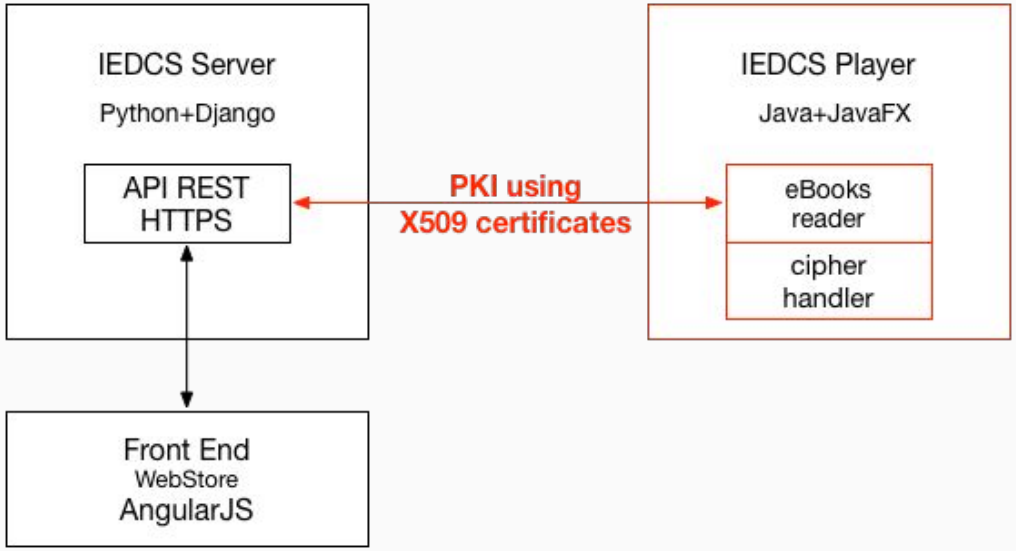
Responsabilities:

- **Store eBooks in the FS**
- **Manage all the connections with Player and FrontEnd**
- **Authenticate users**
- **Control node**

Security measures:

- **Cipher process for the eBook purchased and stock eBooks.**
- **SQL Injections, Buffer Overflows and XSS Attacks + Cross-site request forgery**
- **Authorization and authentication process**

IEDCS Player



Responsabilities:

- **Read ciphered eBooks**
- **Handle cipher process to get eBook content one page at the time** *(Random Access Stream cipher)*
- **Don’t store eBook content**

Security measures:

- **Cipher process for the eBook purchased and stock eBooks.**
- **Buffer Overflows**
- **Check player intigrity with JAR File Signature or Cyclic redundancy check (CRC)**
- **Generate Device key and send it to IEDCS Server**

Use cases

When the player is installed

IEDCS Server:

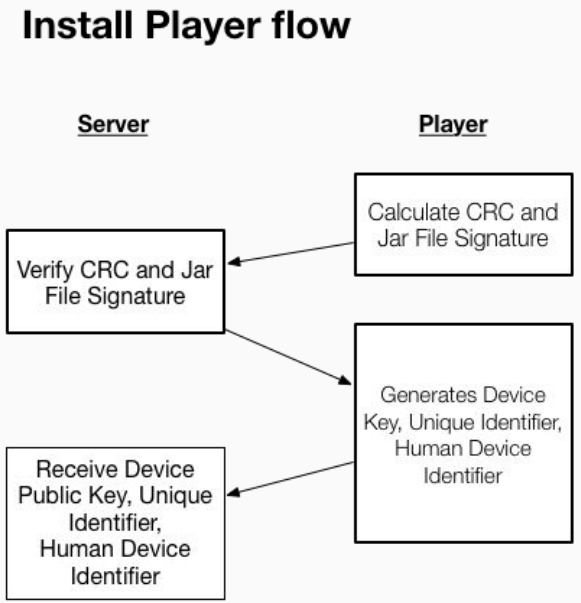
- **Receive** Device Public Key, Unique Idenfier and human device identifier

IEDCS Player:

- **Generates an assymetric device key**
- **Genarates an unique identifier**
  - Ethernet MAC addresses, devices present, CPU, serial numbers of the mother board
- **Send one human device identifier and the device key + unique identifier**

Security measures:

- **Check the integrity of the Player with one exchange by the player to the server with the CRC and Jar File Signature**



Use cases

When an user buy a book

IEDCS Server:

- **Cipher and store the book with cipher process**
- **Generates a File key**

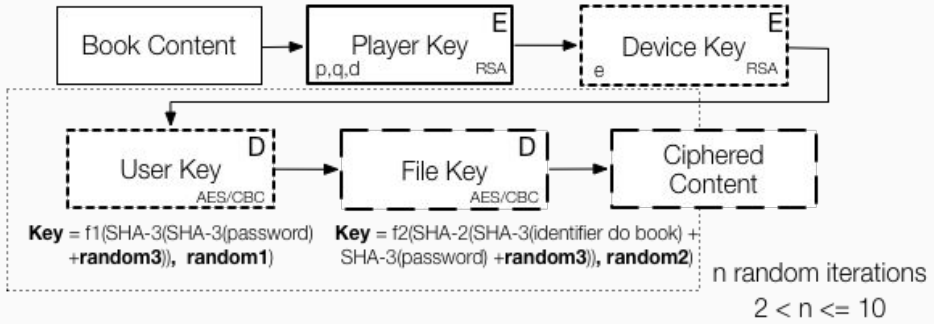
Security measures:

- **Cipher process for the eBook purchased**
- **Store some cryptography information in the database with a cipher process**
- **n = (random3 mod nº max it)**

Token generated:

- **File identifiers**
- ***Crypto tokens:***
  - **random3** *(will be sent when the player initiate the challange)*

Book content cipher process



f1(bit[] array_secret, bit[] valor_aleatorio):
-> p = array_secret + valor_aleatorio
-> p = p << 1
-> p = p mod valor_aleatorio

f2(bit[] array_secret, bit[] valor_aleatorio):
-> p = array_secret mod valor_aleatorio
-> p = p + 3
-> p = p >> 1

Note: The server and the player knows: SHA-3(password)

e = encrypt, d = decrypt

Use cases

When an user buy a book

IEDCS Server:

- **Cipher and store the book with cipher process**
- **Generates a File key**

Security measures:

- **Cipher process for the eBook purchased**
- **Store some cryptography information in the database with a cipher process**

this will be used for store some random numbers...

User crypto information

When an user buys a book..

f1(SHA-3(password), user_email):
-> p = SHA-2(SHA-3(password)<<6 + SHA-3(user_email)[first two bytes])
-> p = p << 1
-> return p

PBKDF2

f2(content, key):
-> AES/CBC(content, key)

f2(content, f1(SHA-3(password), user_email))

Use cases

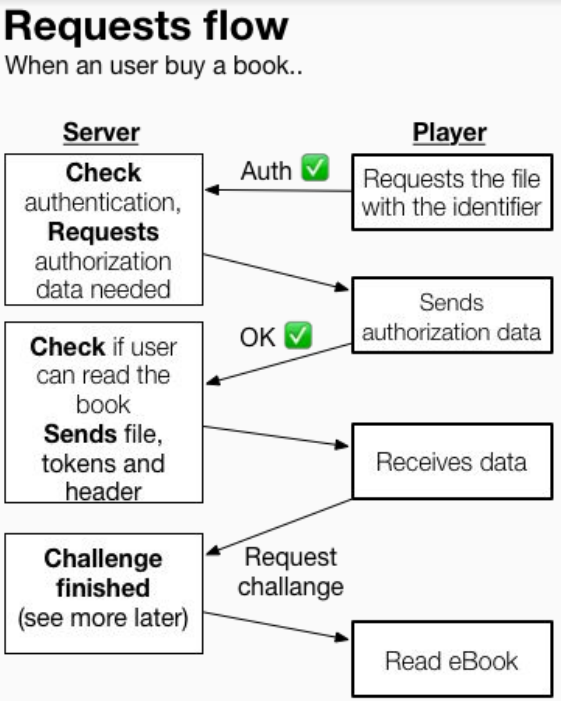
When an user view the book in the Player

IEDCS Server sends:

- **Authentication Token**
- **Authorization Token**
- **Identification Token**
- **Cryptographic headers**
- **Ciphered eBook**

IEDCS Player needs:

- **Check the player integrity**
- **Check the file integrity (CRC)**
- **Decrypt the eBook**
- **Challange the server to end the decryption**



Use cases

When an user view the book in the Player

IEDCS Server sends:

- **Authentication Token**
- **Authorization Token**
- **Identification Token**
- **Cryptographic headers**
- **Ciphered eBook**

IEDCS Player needs:

- **Check the player integrity**
- **Check the file integrity (CRC)**
- **Decrypt the eBook**
- **Challange the server to end the decryption**

Authentication token

X-Auth-Token User	Player Jar Signature

Identification header

Book identifier	File identifiers

Cryptographic header

random3

Authorization token

Restrictions

Location	Player ID
OS	System ID
Time of the Day	

Use cases

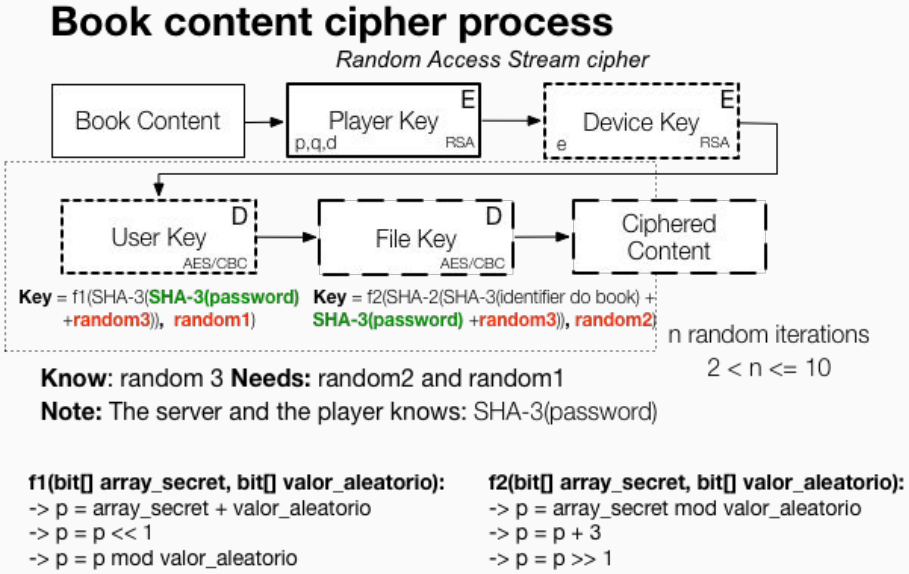
When an user view the book in the Player

IEDCS Server sends:

- **Authentication Token**
- **Authorization Token**
- **Identification Token**
- **Cryptographic headers**
- **Ciphered eBook**

IEDCS Player needs:

- **Check the player integrity**
- **Check the file integrity (CRC)**
- **Decrypt the eBook**
- **Challange the server to end the decryption**



Use cases

When an user view the book in the Player

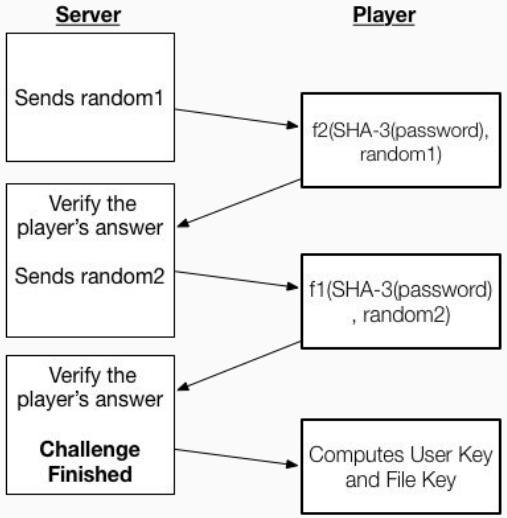
IEDCS Server sends:

- **Authentication Token**
- **Authorization Token**
- **Identification Token**
- **Cryptographic headers**
- **Ciphered eBook**

IEDCS Player needs:

- **Check the player integrity**
- **Check the file integrity (CRC)**
- **Decrypt the eBook**
- **Challange the server to end the decryption**

Challenge flow



END

"The quieter you become, the more you are able to hear."