

# Relatório do protótipo

O protótipo contém um servidor web, um serviço web, e um aplicativo Android que faz requisições ao serviço.

## 1. O servidor:

O servidor web é o Jetty, ele está configurado para funcionar com HTTPS (mais detalhes na seção 6.6.3 da monografia).

No pacote ele já está pronto para uso, inclusive com o serviço compilado e instalado (server/jetty/webapps/service.war).

Para executá-lo:

```
java -jar server/jetty/start.jar
```

O Jetty realiza a negociação TLS e faz uma verificação inicial dos certificados de cliente recebidos, negando a conexão em alguns casos.

As requisições que chegam no serviço, tem seu certificado de cliente analisado mais granularmente.

### 1.1 Keyper:

Keyper é um fork do keyczar (<https://github.com/google/keyczar>) uma ferramenta de criptografia feito pelo Google.

Ele acrescenta classes para manipular certificados digitais e é necessário para o jetty-extensions (veja a próxima seção) e para o serviço funcionarem.

Para compilá-lo e instalá-lo localmente é preciso rodar:

```
gradle -p server/keyper/java/code publishToMavenLocal
```

### 1.2 Jetty-extensions:

É um pacote Java desenvolvido para estender duas funcionalidades do Jetty.

1. Obter a keystore e a truststore do S3 (explicado na seção 6.6.2 da monografia).

2. Um mecanismo de verificação de certificados revogados alternativo (explicado na seção 6.6.4 da monografia).

Ele necessita do Keyper para ser compilado.

Para compilá-lo execute:

```
gradle -p server/jetty-extensions shadowJar
```

Em seguida, é necessário mover o JAR gerado para o diretório do Jetty:

```
cp server/jetty-extensions/build/libs/jetty-extensions-4.0.0-shadow.jar  
server/jetty/lib/ext/
```

## 2. O serviço

O serviço web foi feito na linguagem Clojure utilizando o framework Pedestal.

Ele possui três rotas:

POST /gen-certificate/:subject (POST data: 'pubkey="PUBKEY\_PEM"') - Gera certificados de cliente com subject :subject e chave pública (em formato PEM) PUBKEY\_PEM. O certificado é devolvido em formato PEM no corpo da resposta.

GET /public - Uma rota que aceita conexões tanto com certificados de cliente quanto sem

GET /private - Uma rota que aceita conexões apenas com certificados de cliente válidos

Para compilar o serviço (é necessário possuir o Leiningen - <http://leiningen.org>):

```
cd server/service
./build-war
```

Em seguida é necessário copiar o WAR gerado para o diretório do Jetty:

```
cp service.war ../jetty/webapps/
```

### 3. O diretório certificates

O diretório certificates contém todos os certificados usados pelos testes realizados pelo protótipo. Eles foram gerados com o script `certificates/certificates-generator-script`.

Para executá-lo é necessário ter o openssl e o sss instalados.

#### 3.1 Gerando os certificados

Para gerar a estrutura inicial:

```
./certificates-generator-script ca
```

Isso gera um certificado raiz "root" e dois troncos assinados pela raiz "other" e "client".

Os troncos são usados para assinar certificados finais.

A chave privada da raiz é encriptada, e a senha usada para encriptação é dividida em vários pedaços através de secret sharing (seção 3.7 da monografia).

Para gerar certificados finais:

```
./certificates-generator-script req desired-subject
./certificates-generator-script sign desired-subject signing-branch-subject
./certificates-generator-script pkcs12 --addkey desired-subject
```

Por fim é necessário gerar a `truststore.p12` usada pelo Jetty e fazer o upload de alguns desses certificados no S3 da Amazon, pois esses arquivos são obtidos pelo Jetty a partir do S3.

Para isso é necessário executar (é necessário ter o Java keytool e o aws-cli instalados):

```
./sync-script
```

### 4. Testando o servidor e o serviço

Em seguida são realizados diversos testes no servidor e no serviço através da realização de requisições HTTPS utilizando a ferramenta curl.

OBS.: O parâmetro `--insecure` é para o curl aceitar os certificados de teste usados.

#### 4.1 Requisição para gerar certificado:

```
curl --include --insecure --data-urlencode 'pubkey='$(cat
certificates/gen-certificate-pubkey.pem)"
https://localhost:4443/gen-certificate/teste
```

Resultado: Sucesso (HTTP 200), PEM do certificado gerado é recebido na resposta

Resposta:

```
HTTP/1.1 200 OK
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/plain
```

```
Transfer-Encoding: chunked
Server: Jetty(9.3.11.v20160721)
```

```
-----BEGIN CERTIFICATE-----
MIIDVzCCAj+gAwIBAgIRAP6pt/7IBkTfhF5vvmQssIcwDQYJKoZIhvcNAQELBQAwODELMAkGA1UEB
hMCQlIxGDAWBgNVBAoMD3RjYy1jbGllbnQtYXV0aDEPMA0GA1UEAwwGY2xpZW50MB4XDTE2MTIxMD
IxMjIzM1oXDTE3MDYwODIxMjIzM1owNzELMAkGA1UEBhMCQlIxGDAWBgNVBAoMD3RjYy1jbGllbnQ
tYXV0aDEOMAwGA1UEAwwFdGVzdGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDQyjjb
2P/xLKdo0x3lAoyaQnt5esB0B2kS0EtZI1K5f+6/2pkda8/UAnDIadoS6odMhbWAo3UzlcM7owcAy
adA4dXC+34ZKP0zBlCX6XBN3QrK0o0gYyoCDyJ1kEALbYcyQk/2xnGXSTNz64W0bZEf8fWrKb3g4
zc5ZtLT400Aej05SnFHPbNg/ILi5dAeAjtdMMi8ByK7W2J3Ecw4IRww8i24N+rXzGtHZf/30CB1MR
g5QDNe04tHjs6L3II89tB1exK0cKyb923HC27YZPycHy13Kww7lRiehRAzKHTxlpeo0tcU7i0y6tH
jme5iAj/YnblTMLjcek0PXpYIDSLAgMBAAGjXTBbMB0GA1UdDgQWBBQzLmIJHko4j02QWP5T09h+
RduoTAfBgNVHSMEGDAWgBSiqtHZ1xp7v7hZrTDp7Ve3mWhwcTAJBgNVHRMEAjAAMA4GA1UdDwEB/w
QEAwIDuANBggkqhkiG9w0BAQsFAAOCQAQEAxQ06Ign/UN5yc6omNgvksWtxm+NFpta5giCd1rBuf7A
aKnU9xggLy0WB49hK3I4dComTdGAz2Il9QBrWGoBy5jWAVBvS9wXsI+eXgZQB9pzyBVjF37uzNYW
lBFnt1qaVlClbULi3AMHnJedXs2X4CACub9nQ/eAcxYeCxfi3DPdLm4sKFA44dXp8LR+/uWGjykG
pq7W2f7P4wRr7ojm1off07MrJksN2h29gSxd0BPstq9WT4ww2sST1tfW0w2r+lJCgcQSahYDHA4eJ
z4wI24YxjAJA3tFTiLIVHNvIPcOK+EDu6guCpFwRpxYaWAhP/P52uV2Rof2C4XZoC0Yg==
-----END CERTIFICATE-----
```

## 4.2 Requisições para a rota pública:

### 4.2.1 Requisição sem certificado:

```
curl --include --insecure https://localhost:4443/public
```

Resultado: Sucesso (HTTP 200)

Resposta:

```
HTTP/1.1 200 OK
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/plain
Transfer-Encoding: chunked
Server: Jetty(9.3.11.v20160721)

No client certificate presented.
```

### 4.2.2 Requisição com certificado correto:

```
curl --include --insecure --cert certificates/correct_cert.pem --key
certificates/correct_key.pem https://localhost:4443/public
```

Resultado: Sucesso (HTTP 200)

Resposta:

```
HTTP/1.1 200 OK
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/plain
Transfer-Encoding: chunked
```

```
Server: Jetty(9.3.11.v20160721)
```

Certificate presented:

Subject: correct

Issuer: client

Serial number: 177842608353843309510464147802111744855

#### 4.2.3 Requisição com certificado revogado:

```
curl --include --insecure --cert certificates/revoked_cert.pem --key  
certificates/revoked_key.pem https://localhost:4443/public
```

Resultado: Bloqueado no Jetty (HTTP 401)

Resposta:

```
HTTP/1.1 401 Unauthorized  
Strict-Transport-Security: max-age=31536000; includeSubDomains  
Content-Type: application/json  
Content-Length: 33  
Server: Jetty(9.3.11.v20160721)
```

```
{ "error": "revoked_certificate" } HTTP/1.1 401 Unauthorized  
Strict-Transport-Security: max-age=31536000; includeSubDomains  
Content-Type: application/json  
Content-Length: 33  
Server: Jetty(9.3.11.v20160721)
```

```
{ "error": "revoked_certificate" }
```

#### 4.2.4 Requisição com certificado expirado:

```
curl --include --insecure --cert certificates/expired_cert.pem --key  
certificates/expired_key.pem https://localhost:4443/public
```

Resultado: Bloqueado na negociação TLS no Jetty

Resposta:

```
curl: (35) error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert  
certificate unknown
```

#### 4.2.5 Requisição com certificado válido mas não aceito pela truststore do Jetty:

```
curl --include --insecure --cert certificates/valid_but_not_accepted_cert.pem  
--key certificates/valid_but_not_accepted_key.pem https://localhost:4443/public
```

Resultado: Bloqueado na negociação TLS no Jetty

Resposta:

```
curl: (35) error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert  
certificate unknown
```

### 4.3 Requisições para a rota privada:

#### 4.3.1 Requisição sem certificado:

```
curl --include --insecure https://localhost:4443/private
```

Resultado: Bloqueado pelo serviço (HTTP 401)

Resposta:

```
HTTP/1.1 401 Unauthorized
```

```
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/plain
Transfer-Encoding: chunked
Server: Jetty(9.3.11.v20160721)
```

ERROR: No certificate

#### 4.3.2 Requisição com certificado correto:

```
curl --include --insecure --cert certificates/correct_cert.pem --key
certificates/correct_key.pem https://localhost:4443/private
```

Resultado: Sucesso (HTTP 200)

Resposta:

```
HTTP/1.1 200 OK
Strict-Transport-Security: max-age=31536000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Type: text/plain
Transfer-Encoding: chunked
Server: Jetty(9.3.11.v20160721)
```

Certificate presented:

Subject: correct

Issuer: client

Serial number: 177842608353843309510464147802111744855

#### 4.3.3 Requisição com certificado revogado:

```
curl --include --insecure --cert certificates/revoked_cert.pem --key
certificates/revoked_key.pem https://localhost:4443/private
```

Resultado: Bloqueado no Jetty (HTTP 401)

Resposta:

```
HTTP/1.1 401 Unauthorized
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: application/json
Content-Length: 33
Server: Jetty(9.3.11.v20160721)

{ "error": "revoked_certificate" }
```

#### 4.3.4 Requisição com certificado expirado:

```
curl --include --insecure --cert certificates/expired_cert.pem --key
certificates/expired_key.pem https://localhost:4443/private
```

Resultado: Bloqueado na negociação TLS no Jetty

Resposta:

```
curl: (35) error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert
certificate unknown
```

#### 4.3.5 Requisição com certificado válido mas não aceito pela truststore do Jetty:

```
curl --include --insecure --cert certificates/valid_but_not_accepted_cert.pem  
--key certificates/valid_but_not_accepted_key.pem
```

`https://localhost:4443/private`

Resultado: Bloqueado na negociação TLS no Jetty

Resposta:

```
curl: (35) error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert  
certificate unknown
```

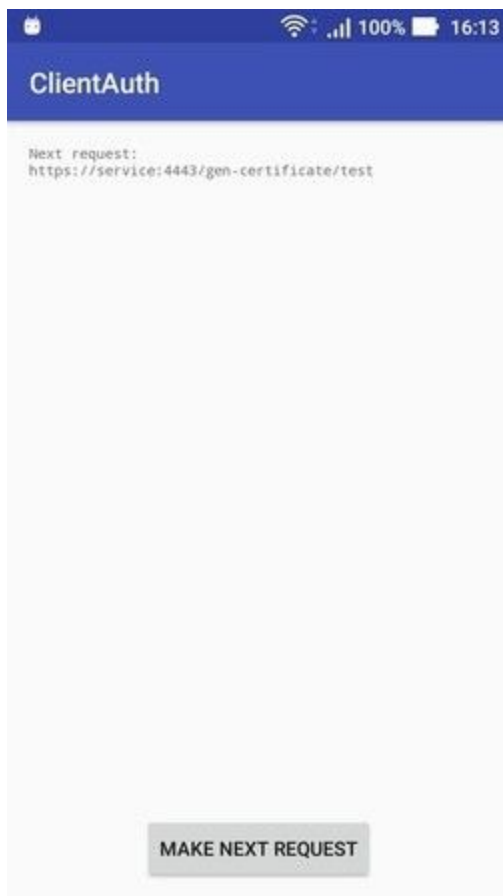
## 5. O aplicativo Android

O aplicativo Android faz requisições HTTPS ao serviço utilizando certificados.

Ele é constituído de uma tela que exibe informações e possui um botão que realiza a próxima requisição. No total são 13 requisições. Abaixo são mostradas cada uma das requisições através de screenshots do aplicativo.

O código está no diretório ClientAuth. É possível instalá-lo através do Android Studio, abrindo o projeto com ele e cliando no botão Run, isso com o celular conectado via USB com opção de Debug habilitada.

### 5.1 Tela inicial



### 5.2 Requisição para gerar certificado:

Resultado: Sucesso (HTTP 200), PEM do certificado gerado é recebido na resposta



## 5.3 Requisições para a rota pública:

### 5.3.1 Requisição sem certificado:

Resultado: Sucesso (HTTP 200)



### 5.3.2 Requisição com o certificado gerado na primeira requisição:

Resultado: Sucesso (HTTP 200)



### 5.3.3 Requisição com certificado correto:

Resultado: Sucesso (HTTP 200)



### 5.3.4 Requisição com certificado revogado:

Resultado: Bloqueado no Jetty (HTTP 401)



### 5.3.5 Requisição com certificado expirado:

Resultado: Bloqueado na negociação TLS no Jetty



### 5.3.6 Requisição com certificado válido mas não aceito pela truststore do Jetty:

Resultado: Bloqueado na negociação TLS no Jetty



## 5.4 Requisições para a rota privada:

### 5.4.1 Requisição sem certificado:

Resultado: Bloqueado pelo serviço (HTTP 401)



#### 5.4.2 Requisição com o certificado gerado na primeira requisição:

Resultado: Sucesso (HTTP 200)



#### 5.4.3 Requisição com certificado correto:

Resultado: Sucesso (HTTP 200)



#### 5.4.4 Requisição com certificado revogado:

Resultado: Bloqueado no Jetty (HTTP 401)



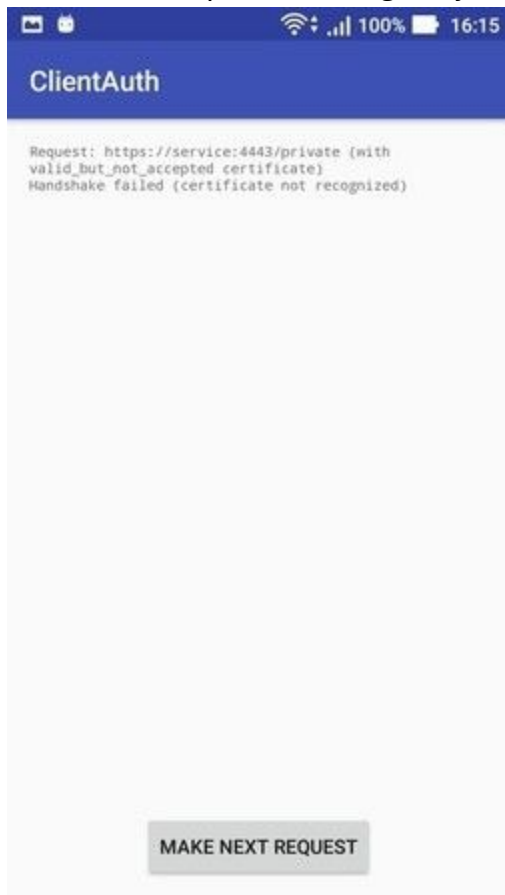
#### 5.4.5 Requisição com certificado expirado:

Resultado: Bloqueado na negociação TLS no Jetty

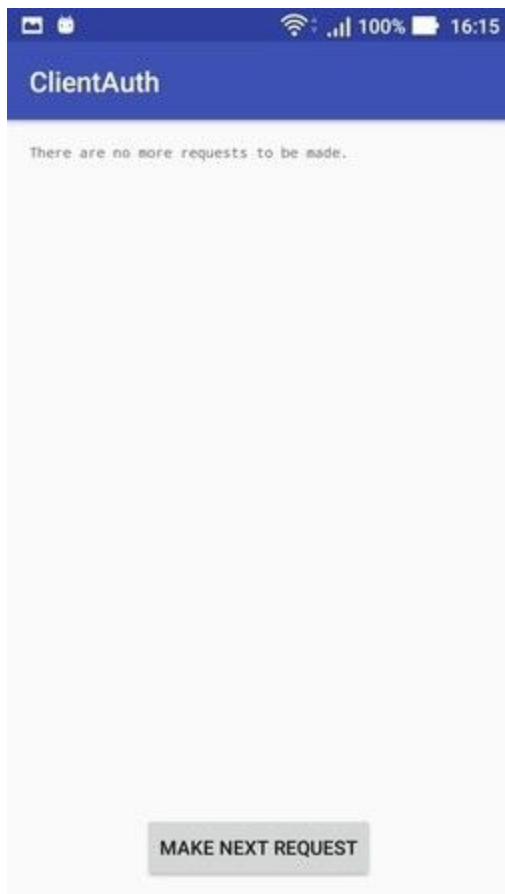


#### 5.4.6 Requisição com certificado válido mas não aceito pela truststore do Jetty:

Resultado: Bloqueado na negociação TLS no Jetty



#### 5.5 Tela final



### 6. Conclusão

Todos os resultados demonstrados pelos testes estão de acordo com o esperado. Portanto o protótipo funciona e alcança o objetivo de demonstrar como é possível efetuar autenticação e autorização através de certificados de cliente.