Gerar certificado dinâmico Itaú

Requisitos

Para a geração do certificado dinâmico, é necessário ser instalado:

1. OpenSLL - Precisamos dele para rodar alguns comandos.
2. GitBash - Terminal utilizado para escrever os comandos.
3. Java - Necessário para rodar alguns scripts de código.
4. Postman - Software para acesso de endpoints.

1- Obter par de chaves

Para gerar um par de chaves RSA, a qual é composta por uma chave privada (para decifrar) e uma chave pública (para criptografar), execute o seguinte comando no **GitBash**:

```
openssl genpkey -out private.pem -algorithm RSA -pkeyopt rsa_keygen_bits:2048
openssl rsa -in private.pem -pubout -out public.pem
```

Após executar os comandos, o par de chaves será gerado através de um arquivo denominado **public.pem** e outro **private.pem.**

Exemplo do conteúdo do arquivo **public.pem**:

```
-----BEGIN PUBLIC KEY-----MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwtCm21MHeGMHN8e4uAZi
U/pKoIrtLoyOuoLmpogmWdiV2OJz3OjysQf7JLE1o7xmkwHI8JGg41wuUv+CF6Ax
0kk3EwV6AVkLnI2lOAY2LbrgwDtsGQh6gslIUJhiadHqwwnezBanTIgV+Z4sa02O
SWZGbWufXJ9PVP91LXy/d4oyHGbKMVUQlyK350+iL8Pw0reTy6MiKKPrdSaKkj87
PhrE/Fd3vIE1ANqhJjIgHNtB4iqo4VF8bBZKohkShu8IA5Y3uV2OkMz3lD/jNtJn
8J7D7uKYCB8Xy9Sy/8O+rwvyly7i7nWLG1zpVz7KD7+3KWrnmuf2o1gefCin1FLv
```

```
pwIDAQAB-----END PUBLIC KEY-----
```

2 - Recebimento do Token temporário

Nesta etapa, deve-se enviar o arquivo public.pem para o time de atendimento do Itaú. Com esse arquivo o time do Itaú vai criptografar as informações de "**Token temporário**" e "**client id**" e envia-lo a seu e-mail cadastrado junto a eles. Esse processo pode demorar em média 1 dia útil.

Importante ressaltar que o Token temporário tem validade de **7 dias**.

Exemplo de e-mail retornado pela equipe Itaú:

```
E-mail de credenciais
Olá parceiro, seja bem vindo ao Itaú_
Você já pode gerar seu certificado dinâmico com as credenciais abaixo:
ClientId: KWMMqDhJ+p/2fejfX2uktyDnAjT8B3u2FZCyPlyA5U5t3IU4VVcS6wY5KG+SZO+a
Token Temporário:
+B8tD52L3qrexAjr3vinhj1FGrfr/iGzx/lKhE2mHJPcjOCnLbCcqP52w3YDXNL+Zi2RKmEp0ZwRFzZYJzWNBMiQh65CKTIRMPBA/2
Chave Sessão:
KT9d4C4mthX9aFWHGwUlvBS3quLagK2NrDDAXxQWiBCcpu7RDdQE+/C0dATzAjMUUiqsuKpFSnakhI7b0uNAF3CzqNgTnGc6xtu3iO
```

3 - Decifrar credenciais

Após receber o e-mail com as informações, crie um arquivo chamado **Criptografia.java** com o seguinte código dentro dele:

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;import java.security.interfaces.RSAPrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;
```

```
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
public class Criptografia {
    private static String extrairChaveRsaPem(String tipoChave, String arquivoChavesRsa)
{
        try {
            InputStream is = new FileInputStream(arquivoChavesRsa);
@SuppressWarnings("resource")            BufferedReader br = new BufferedReader(new
InputStreamReader(is));
            StringBuilder sb = new StringBuilder();            boolean inKey =
false;            for (String line = br.readLine(); line != null; line = br.readLine())
{
                if (!inKey) {                    if (line.startsWith("-----BEGIN ") &&
line.endsWith(" " + tipoChave + " KEY-----")) {                        inKey =
true;
                    }
                } else {                    if (line.startsWith("-----END ") &&
line.endsWith(" " + tipoChave + " KEY-----")) {                        inKey =
false;
                        break;
                    }
                    sb.append(line);
                }
            }
            return sb.toString();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    public static String decriptografiaAes(SecretKey key, String cipherText) {
        try {            Cipher cipher =
```

```
Cipher.getInstance("AES/CBC/PKCS5Padding");           IvParameterSpec iv = new
IvParameterSpec(new byte[16]);           cipher.init(Cipher.DECRYPT_MODE, key,
iv);           byte[] plainText =
cipher.doFinal(Base64.getDecoder().decode(cipherText));
            return new String(plainText);
        }
        catch(NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        catch(BadPaddingException e) {
            e.printStackTrace();
        }
        catch(IllegalBlockSizeException e) {
            e.printStackTrace();
        }
        catch(InvalidKeyException e) {
            e.printStackTrace();
        }
        catch(InvalidAlgorithmParameterException e) {
            e.printStackTrace();
        }
        catch(NoSuchPaddingException e) {
            e.printStackTrace();
        }
        return null;
    }
    private static byte[] decriptografiaRsa(String caminhoChavePrivada, String
dadosCifrados) {
        try {           String chavePrivada = extrairChaveRsaPem("PRIVATE",
caminhoChavePrivada);
            PKCS8EncodedKeySpec spec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(chavePrivada.toString()));
KeyFactory kf = KeyFactory.getInstance("RSA");           RSAPrivateKey privateKey =
(RSAPrivateKey) kf.generatePrivate(spec);
            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.DECRYPT_MODE, privateKey);
            return cipher.doFinal(Base64.getDecoder().decode(dadosCifrados));
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    public static void main(String[] args)
    {
        SecretKey chaveSessao = null;        Scanner scanner = new
Scanner(System.in);
        try {
System.out.println("\n========================================");
System.out.println("Informe as informações recebidas no e-mail");
System.out.println("========================================");
            System.out.println("\nClientId: ");
            System.out.flush();            String clientIdCifrado =
scanner.nextLine().trim();
            System.out.println("\nToken Temporário: ");
System.out.flush();
            String tokenCifrado = scanner.nextLine().trim();
            System.out.println("\nChave Sessão: ");
System.out.flush();            String chaveSessaoCifrada =
scanner.nextLine().trim();
            System.out.println("\nCaminho chave privada: ");
System.out.flush();            String caminhoChavePrivada =
scanner.nextLine().trim();
            System.out.flush();
            scanner.close();
            System.out.println("\n====================================");
System.out.println("    Processo de Decriptografia         ");
System.out.println("====================================");
            // Decifra a chave de sessao AES com a chave RSA privada            byte[]
chaveSessaoDecifrada = decriptografiaRsa(caminhoChavePrivada,
chaveSessaoCifrada);             chaveSessao = new SecretKeySpec(chaveSessaoDecifrada, 0,
chaveSessaoDecifrada.length, "AES");            // Decriptografa a credencial atraves da
chave de sessao AES            String clientIdDecifrada = decriptografiaAes(chaveSessao,
clientIdCifrado);            System.out.println("\nClient id decifrado com a chave de sessao
AES:\n[ " + new String(clientIdDecifrada) + " ]");            String tokenDecifrado =
```

```
decriptografiaAes(chaveSessao, tokenCifrado);           System.out.println("\nToken
decifrado com a chave de sessao AES:\n[ " + new String(tokenDecifrado) + " ]");
}
        catch(Exception e) {
            e.printStackTrace();
        }
    }}
```

Após isso, execute os seguintes comandos no terminal **GitBash** (O terminal deve ser aberto na mesma pasta que o arquivo **Criptografia.java** foi salvo):

```
javac Criptografia.javajava -cp . Criptografia
```

Ao executar esses comandos, preencha os dados solicitados no terminal **GitBash** de acordo com o que foi recebido no e-mail:

ClientId: ClientId presente no e-mail
Token Temporário: Token Temporário presente no e-mail
Chave sessão: Chave sessão presente no e-mail
Caminho da chave privada: escreva o caminho onde está o arquivo private.pem gerado anteriormente.

Após informar esses dados, o terminal deve exibir a resposta, semelhante as informações abaixo:

```
Credenciais decifradas
Processo de Decriptografia
Client id decifrada com a chave de sessao AES:
e26f2f89-0ead-4ca6-8bc3-dd44b4ab3cc7
Token decifrado com a chave de sessao AES:
eyJraWQiOiJhZTYxYWIxZi0yNTRhLTQ5ZWQtODMzNC05ZDJlN2E0MzZiNGQuaG9tLmdlbi4xNTk3NjAwMzM2OTkyLmp3dCIsImFsZy
```

4 - Geração do arquivo .csr e .key

Para gerar os arquivos **.csr** e **.key**, deve criar uma pasta para salvar os arquivos, e em seguida abrir o terminal **GitBash** nesta pasta.

Em sequência rodar o seguinte comando:

Para Windows:

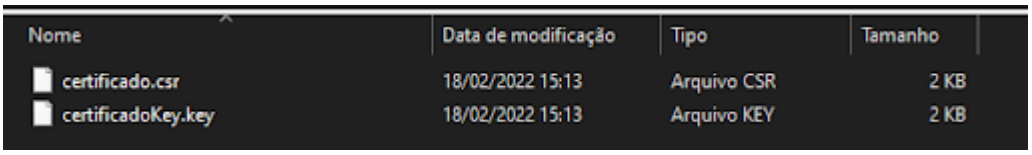
```
openssl req -new -subj "//CN={{CLIENT_ID}}\OU=SITE OU APP DO PARCEIRO\L=CIDADE\ST=ESTADO\C=BR" -out ARQ
```

Para Linux:

```
openssl req -new -subj "/CN={{CLIENT_ID}}/OU=SITE OU APP DO PARCEIRO/L=CIDADE/ST=ESTADO/C=BR" -out ARQU
```

> Lembre-se de alterar algumas das informações do script acima seguindo os dados repassados pelo banco:
> **CN**= ClientID disponibilizado no e-mail
> **OU**= Deixe preenchido com o nome do app que vai utilizar o certificado (exemplo: **MSYSGESTOR**)
> **L**= Cidade onde se localiza a Agência do Cliente (exemplo: **SAO MIGUEL DO OESTE**)
> **ESTADO**= Estado onde se localiza a Agência do Cliente (exemplo: **SANTA CATARINA**)
> **C**= País onde se localiza a Agência do Cliente (exemplo: **BR**)
> **ARQUIVO_REQUEST_CERTIFICADO**.csr= Nome do certificado, deve manter o ".csr" no final
> **ARQUIVO_CHAVE_PRIVADA**.key=Nome da chave vinculada ao certificado, deve manter o ".key" no final
>
> **OBS:** as informações **OU**, **L**, **ESTADO** e **C** devem ser todas escritas em caixa alta e sem nenhuma acentuação ou caractere especial, segundo os exemplos acima.

Ao executar o comando, os seguintes arquivos serão gerados na pasta:



5 - Autenticar arquivo **.csr** e **.key** em um arquivo **.crt**

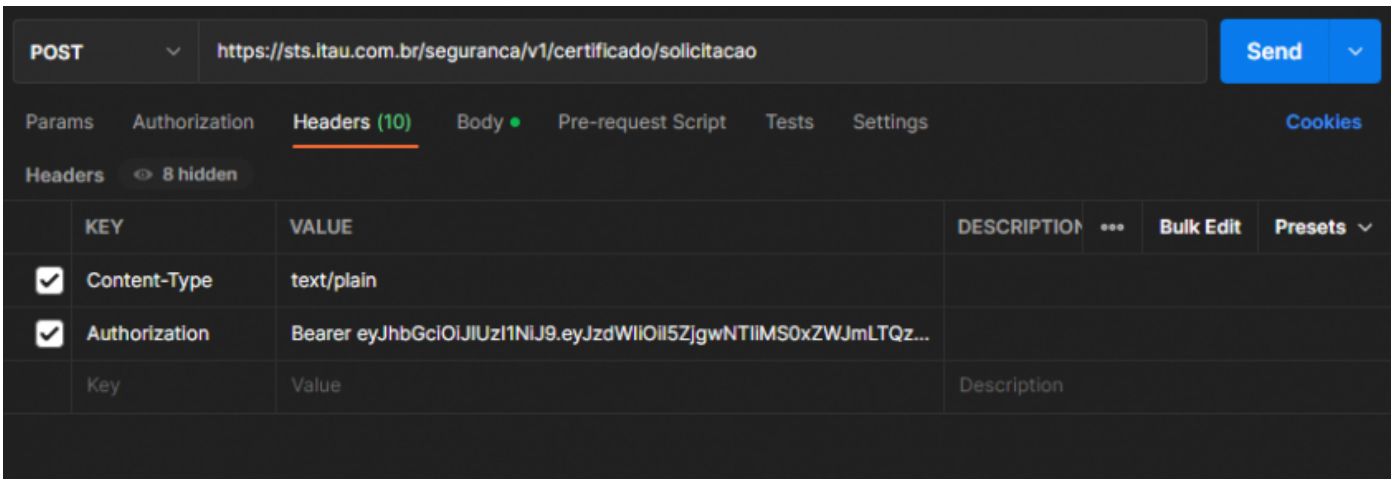
O envio do arquivo **.csr** para o banco deve ser feito via API, através de um *endpoint POST*. Aconselhamos o uso do software **Postman** para isso.

Na **URL** da requisição:

- inserir **https://sts.itau.com.br/seguranca/v1/certificado/solicitacao**

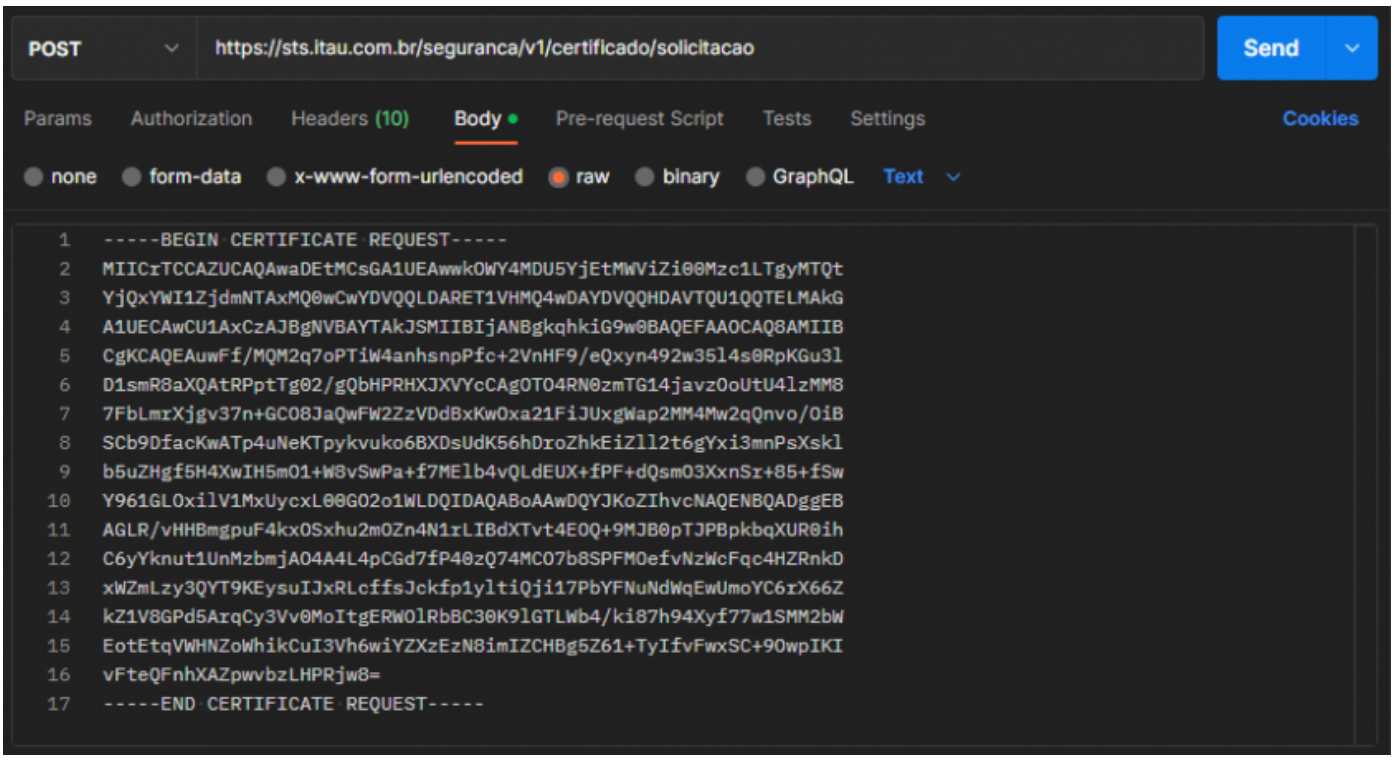
No **header** da requisição:

- Insira um Content-Type= text/plain
- Insira um Authorization: bearer {token temporário}



No **body** da requisição:

- insira o **certificado.csr** gerado anteriormente em texto. Para adquirir o texto desse arquivo, é só abri-lo com um editor de texto.



Ao enviar essa requisição POST para o Itaú, vai ser retornada duas informações, o Secret e a nova informação do certificado.

Copie o Secret e salve em um arquivo com extensão **.key** e o conteúdo do certificado em um arquivo com extensão **.crt**, preferencialmente salve ambos em uma nova pasta.

6 - Gerar arquivo .pfx

Acesse a pasta criada com os arquivos **.crt** e **.key** gerados no passo anterior, e após isso abra o **GitBash** nessa pasta.

Insira a seguinte informação no terminal:

```
openssl pkcs12 -export -out certificado_dinamico_itau.pfx -inkey certificadoKey.key -in certificadoGera
```

> Lembre-se de alterar algumas das informações do script acima seguindo os dados repassados pelo banco:
> **certificadoKey.key**= Insira o nome do arquivo gerado no passo anterior, mantenha a extensão **.key** no final.
> **certificadoGerado.crt**= Insira o nome do arquivo gerado no passo anterior, mantenha a extensão **.crt** no final.

Em sequência vai ser solicitado uma senha para o certificado e a confirmação da senha. Preencha a senha desejada as duas vezes.

Ao final vai ser gerado o seu **certificado dinâmico** para realizar o upload dele no sistema **MsysGestor**.

Referências

https://atendimento.tecnospeed.com.br/hc/pt-br/articles/7003805388567-Utilizando-o-registro-via-Web-Service-Banc%C3%A1rio-com-o-Ita%C3%BA-V2-

https://forum.casadodesenvolvedor.com.br/topic/43985-gerando-certificado-crt-para-o-banco-ita%C3%BA-pix-e-boleto-webservice/

https://forum.casadodesenvolvedor.com.br/topic/43793-como-converter-um-arquivo-de-certificado-crt-para-o-formato-pfx/

https://devportal.itau.com.br/certificado-dinamico

Revision #9

Created Tue, Dec 13, 2022 6:47 PM by Lucas Tumeleiro

Updated Wed, Dec 14, 2022 7:12 PM by Lucas Tumeleiro