UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Departamento de Informática e Estatística
CTC • UFSC

Tarefa Prática – Sistema de autenticação 3FA, derivação de chaves e criptografia simétrica para enviar mensagem para o servidor

Será implementado um sistema de autenticação com 3 fatores (3FA - *Three-factor authentication*): endereço IP (primeiro fator – estar em algum lugar), senha (segundo fator – saber algo), TOTP (terceiro fator – possuir algo). Suponha que o usuário está se autenticando em um servidor de email que só permite enviar mensagens quando o usuário tem sucesso no processo de 3FA. Depois da autenticação, o usuário irá enviar uma mensagem cifrada para o servidor.

Cadastro do usuário

O cadastro do usuário deve armazenar: nome do usuário, local (país ou cidade do usuário), senha no formato SCRYPT(senha) e o salt do usuário. Caso queira, pode armazenar também o número do celular do usuário. **O local deve ser obtido diretamente pelo IP que o usuário está usando**. A forma de fazer isso é explicada adiante.

Cadastro dos usuários no servidor

Nome do usuário	Número celular	Local (país)	Senha	Salt
nome1	999223344	Holanda	SCRYPT da senha1	ac1e37bfb15599e5f4
nome2	999887766	Brasil	SCRYPT da senha2	20814804c1767293b
nome3	999887755	USA	SCRYPT da senha2	20814804c1767293b

No lado do cliente deve existir um “cadastro de secrets” usados no TOTP. Lembrando que o cálculo do TOTP é feito assim: TOTP = HMAC(secret, time). Veja a figura 1. Existe código com exemplo de uso do TOTP no diretório “2fa”. Os exemplos em Java estão no arquivo exemplosFIPS2.zip.

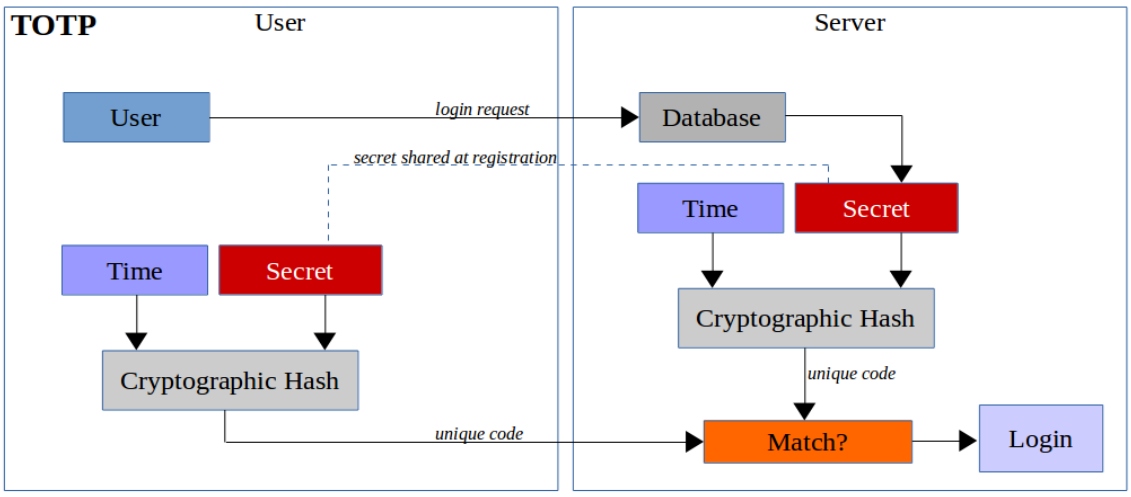

Figura 1 – TOTP – Time-based One Time Password.

O TOTP deve ser usado para derivar uma chave simétrica que será usada em ambos os lados – cliente e servidor. No lado do cliente essa chave simétrica será usada para cifrar a mensagem. No lado do servidor essa chave simétrica será usada para decifrar a mensagem e mostrar a mensagem decifrada na tela.

Verificando o IP

Para identificar o local (país ou cidade), você deve usar a biblioteca IPInfo (ou similar que funcione na linguagem que você está usando) para identificar o país/cidade a partir do IP do usuário (https://github.com/ipinfo/java). Um código exemplo em Java foi postado junto com a definição da tarefa.

Você precisa se cadastrar gratuitamente em https://ipinfo.io/signup para poder fazer consultas relativas ao IP. Você terá um "token" a ser usado dentro do código para fazer a consulta (figura 2).

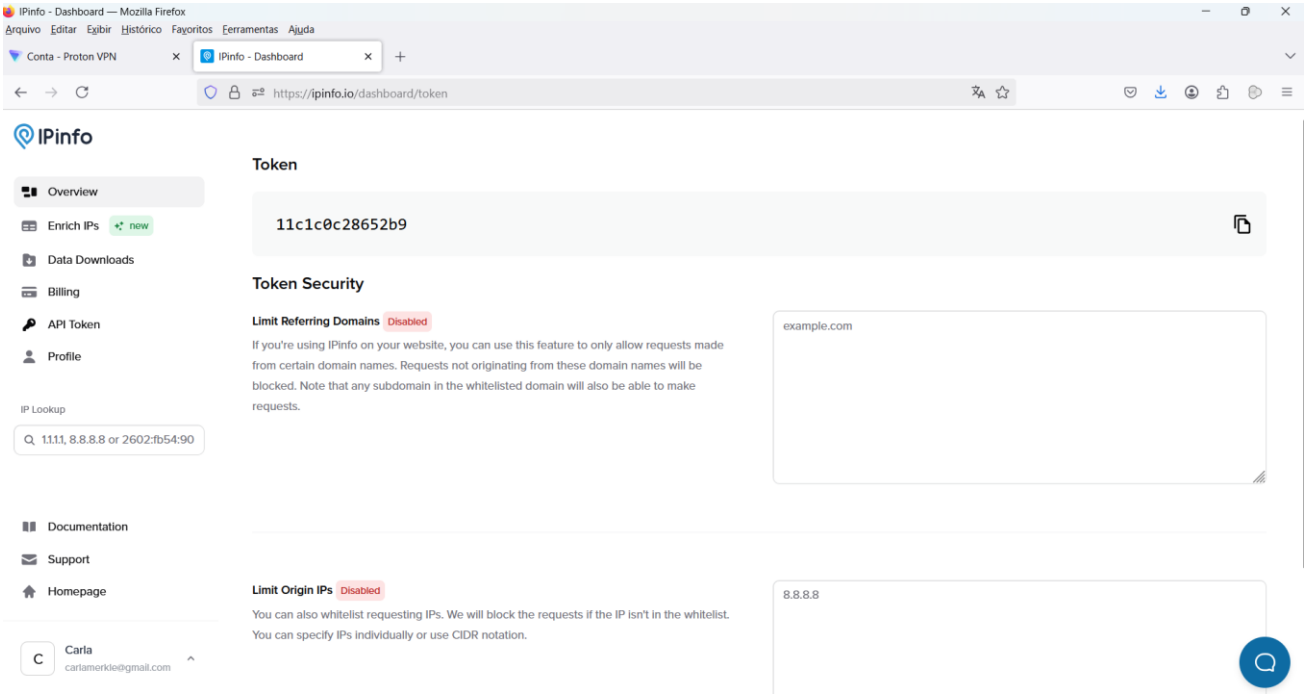

Figura 2 – Token a ser usado dentro do Código para consulta IPInfo.

Para testar IPs de países diferentes para os usuários da aplicação

Para testar o uso de IPs de países diferentes, use uma VPN (*Virtual Private Network*). Caso não use nenhum software de VPN, instale ProtonVPN:

- Crie conta na Proton VPN (free): https://account.protonvpn.com/pt-br/signup?plan=free&currency=EUR&ref=noupsell
- Instale a Proton VPN no seu sistema: https://protonvpn.com/download
- Execute o software Proton VPN, conectando seu computador com um dos servidores.

Você poderá usar as bibliotecas criptográficas fornecidas pelo provedor Bouncy Castle. As bibliotecas estão no diretório chamado “fips” do arquivo exemplosFIPS2.zip.. O provedor BCFIPS que é a versão FIPS da Bouncy Castle.

O arquivo exemplosFIPS2.zip possui os exemplos que você **DEVE EXECUTAR** e **ENTENDER** para poder desenvolver esta tarefa. Também foi anexado junto com a tarefa o exemplo getinfosip.zip.

Existem 3 bibliotecas de apoio, baixadas via Maven (ver o Exemplo 13 e referência [1]):

- ***totp*** – usa o instante atual (time) para garantir a variabilidade da OTP. TOTP é uma extensão do HOTP
- ***commons-codec*** – para conversões de hex e base32
- ***zxing*** – biblioteca para gerar QR codes

Sua aplicação, na parte de autenticação com TOTP, irá simular a autenticação de 2 fatores do Google, representada na figura 3.

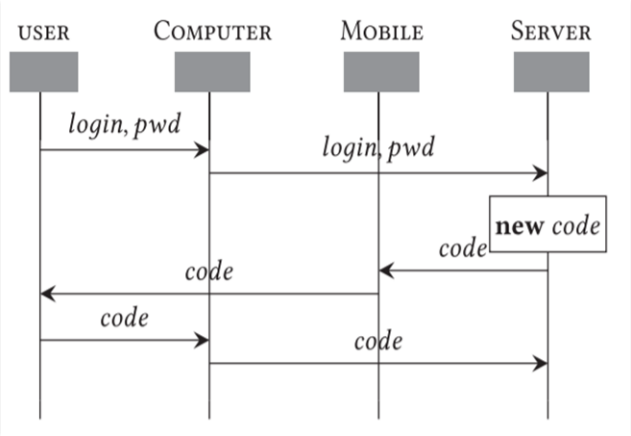

Figura 3 - Google 2-step with Verification Codes: g2V [6].

Atenção especial deve ser dispensada na criação e gerenciamento de parâmetros criptográficos da aplicação:

i. O sistema **não poderá usar** chave e IV armazenados pelo cliente em variáveis globais ou de ambiente no momento da decifragem. Você deve agir como se o cliente e o servidor estivessem localizados em máquinas diferentes.

ii. **Para gerar as chaves/IVs devem ser usados o PBKDF2 ou o Scrypt**.

iii. Deve ser usada **criptografia autenticada para cifragem e decifragem** (modo GCM ou outro).

iv. Decisões próprias sobre formatos e parâmetros devem ser feitas.

v. **NÃO É PERMITIDO ter chaves e IV fixos e escritos no próprio código.**

vi. **Se forem guardados, os parâmetros devem ser guardados em arquivo cifrado. Apenas o salt pode ficar guardado sem cifragem.**

Para entregar/apresentar:

A. O código fonte deve ser postado no moodle, juntamente com um tutorial de execução da aplicação. Deve ser possível executar a aplicação com os arquivos anexados dentro do código.

B. Apresentação desta questão será feita de maneira presencial ou online via Google Meet. Todos os membros da equipe devem apresentar para receber nota. Será usado o próprio código durante a apresentação: a aplicação é executada e depois o código é explicado.

A apresentação deve ser agendada na semana seguinte à entrega do trabalho (períodos da tarde e noite).

Avisos:
**** Se tiver cópias de código, todos os envolvidos receberão nota zero nesta tarefa.**

Referências

[1]. Ihor Sokolyk. Two-Factor Authentication with Java and Google Authenticator, 2019. Disponível em: https://medium.com/@ihorsokolyk/two-factor-authentication-with-java-and-google-authenticator-9d7ea15ffee6

[2]. Jeremy Chan. How Google Authenticator, HMAC-Based One-time Password, and Time-based One-time Password Work, 2021. Disponível em: https://levelup.gitconnected.com/how-google-authenticator-hmac-based-one-time-password-and-time-based-one-time-password-work-17c6bdef0deb

[3]. HOTP: An HMAC-Based One-Time Password Algorithm. RFC, 2005. Disponível em: http://tools.ietf.org/html/rfc4226

[4]. TOTP: Time-Based One-Time Password Algorithm. RFC, 2011. Disponível em: http://tools.ietf.org/html/rfc6238

[5]. Two-factor authentication: https://cryptography.io/en/latest/hazmat/primitives/twofactor/

[6]. Charlie Jacomme and Steve Kremer. 2021. An Extensive Formal Analysis of Multi-factor Authentication Protocols. ACM Trans. Priv. Secur. 24, 2, Article 13 (February 2021), 34 pages. DOI:https://doi.org/10.1145/3440712

[7]. https://ipinfo.io/