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

Departamento de Informática e Estatística
CTC • UFSC

Tarefa Prática – Servidor simulado de nuvem, autenticação de 2 fatores e criptografia simétrica para enviar/receber arquivos

Será implementado um servidor simulado de nuvem. O usuário poderá:
-fazer o cadastro dos seus dados no cliente e no servidor;
-fazer autenticação com 2 fatores (login/senha+TOTP);
-enviar um arquivo cifrado para o servidor;
-baixar um arquivo cifrado do servidor; e,
-mostrar o arquivo decifrado na tela.

Inicialmente acontece o processo de cadastro do novo usuário. Depois de cadastrado, o usuário realiza a autenticação de 2 fatores no servidor. Quando estiver autenticado, o usuário poderá enviar arquivos cifrados para o servidor ou baixar arquivos cifrados do servidor. Depois de baixar o arquivo, o programa deve decifrar o seu conteúdo e mostrar na tela.

Nenhuma chave de cifragem/decifragem pode ser guardada no servidor. O cliente também não poderá guardar a chave em lugar nenhum – essa chave usada para cifrar/decifrar DEVE SER DERIVADA em tempo de execução no lado do cliente.

O cadastro do usuário deve armazenar: nome do usuário, senha no formato compatível com o que é mostrado na figura 1. Caso queira, pode armazenar também o número do celular do usuário ou algum outro dado necessário.

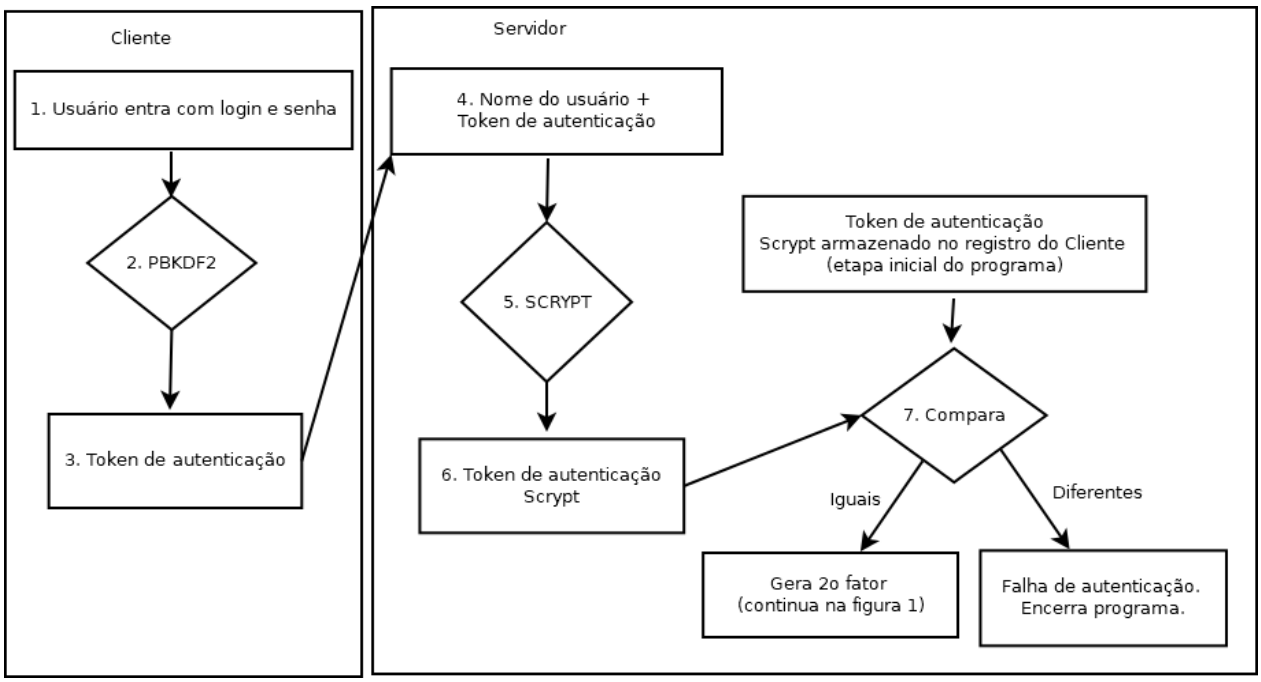

Figura 1 – Etapa da derivação de chaves e autenticação usando PBKDF2 e SCRYPT.

A figura 1 mostra o processo de autenticação do usuário com login e senha. Caso esse processo ocorra com sucesso, é gerado um 2º fator de autenticação para o usuário.

Lembre que no TOTP é necessário que cada usuário tenha um "secret" armazenado no servidor, já que o cálculo do TOTP é feito assim: TOTP = HMAC(secret, time). Veja a figura 2. Existe código com exemplo de uso do TOTP no diretório "2fa". Os exemplos em Java estão no arquivo exemplosFIPS2.zip.

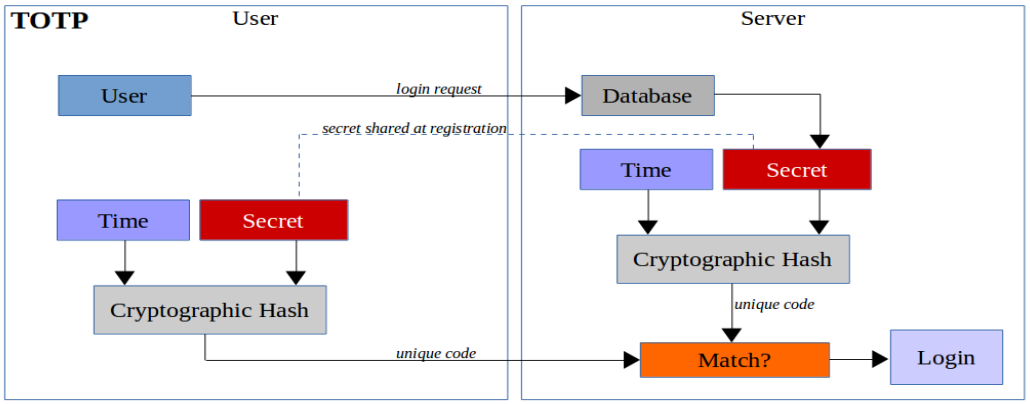

Figura 2 – TOTP – Time-based One Time Password.

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.

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.

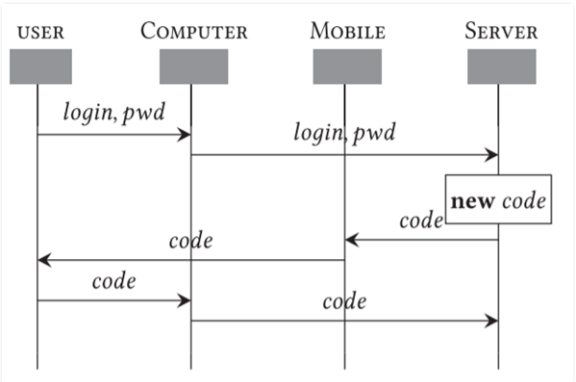

Figura 3 - Google 2-step with Verification Codes: g2V [6].

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).

Pode ser usada qualquer linguagem de programação, desde que a linguagem tenha biblioteca criptográfica disponível e possua os mecanismos necessários para a elaboração da tarefa.

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