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

Tarefa Prática – Simulação de pedido de *iFood* - Autenticação de dois fatores (2FA), derivação de chaves e criptografia simétrica

Será implementada uma aplicação para simular o pedido de um prato de comida para um restaurante, simulando um sistema de iFood:

1. O usuário escolhe o prato de comida que irá pedir;
2. O sistema pede o celular do usuário;
3. O sistema gera código TOTP e envia o QR Code para o usuário (2º fator de autenticação, simulando a necessidade do celular);
4. O usuário lê o QR Code (ou o código) e digita o código obtido na tela para enviar para o sistema;
5. O sistema valida o código obtido como 2º fator de autenticação;
6. O sistema e o usuário criam ambos, uma chave de sessão **usando o 2º fator de autenticação e o PBKDF2 ou SCRYPT para derivar essa chave**;
7. O usuário faz o pagamento e envia o comprovante cifrado com a chave de sessão para o sistema. O usuário usa o modo GCM de criptografia autenticada para cifrar;
8. O sistema decifra o comprovante e envia o pedido para o restaurante.
9. O sistema envia uma mensagem cifrada para o usuário, avisando o horário que o pedido deve chegar.
10. O usuário decifra e visualiza a mensagem recebida.

Você irá usar as bibliotecas criptográficas fornecidas pelo provedor Bouncy Castle. As bibliotecas estão no diretório chamado “fips”. Usaremos o provedor BCFIPS que é a versão FIPS da Bouncy Castle.

O arquivo disponibilizado no moodle (na área da definição da tarefa) possui os exemplos que você **DEVE EXECUTAR** e **ENTENDER** para poder desenvolver esta tarefa.

Também serão usadas 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, irá simular a autenticação de 2 fatores do Google, representada na figura 1.

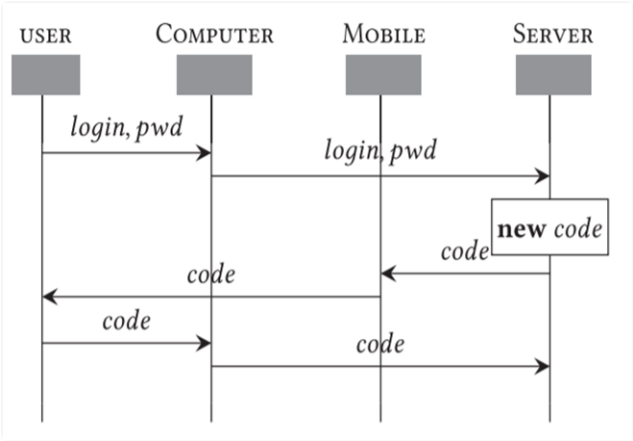

Figura 1 - Google 2-step with Verification Codes: g2V [6].

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 usuário e o sistema estivessem localizados em máquinas diferentes.

ii. Para gerar as chaves/IVs devem ser usados o PBKDF2 ou o Scrypt. Deve ser usada criptografia autenticada para cifragem e decifragem (modo GCM ou outro).

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

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

v. **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.

A apresentação deve ser agendada para 2ª feira, 3ª feira ou 4ª feira da 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://www.twilio.com/pt-br/blog/como-enviar-mensagens-pelo-whatsapp-de-aplicativos-java-usando-api-da-twilio