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

Tarefa Prática – Autenticação de dois fatores (2FA), derivação de chaves e criptografia simétrica

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.

Primeira parte – configurar e explicar a autenticação de 2 fatores (2FA – *Two-factor authentication*) do sistema Authy.

Você deve entender como a autenticação de 2 fatores funciona no sistema Authy (https://www.twilio.com/docs/glossary/totp), como é gerado o fator usando TOTP. Depois disso, você deve:

1. Setar 2FA no linkedin https://authy.com/guides/linkedin/ (ou outro site possível - https://authy.com/guides/).
2. Demonstrar essa autenticação de 2 fatores funcionando na apresentação do trabalho e explicar como acontece (parte teórica e prática).

Para ler sobre o Authy:

- https://www.techrepublic.com/article/authy-vs-google-authenticator/
- https://www.nytimes.com/wirecutter/reviews/best-two-factor-authentication-app/#how-to-set-up-and-use-authy

Segunda parte – Deve ser implementado um sistema de autenticação que irá:

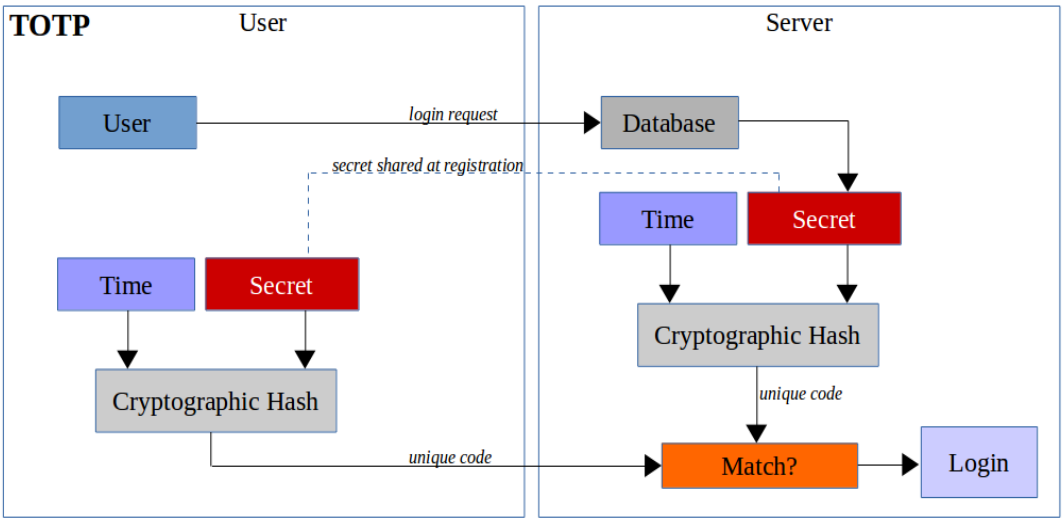
-gerenciar login e senha dos usuários de maneira segura;
-fornecer um segundo fator de autenticação (2FA). Você irá implementar uma aplicação parecida com o "Authy ou Google Authenticator" para fornecer um código de autenticação para ser o 2º fator; e,
-autenticar usuários usando login/senha e um segundo fator de autenticação (cada usuário deve ter um "secret key" diferente cadastrado para obter o segundo fator).

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 podem ser 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

As referências [2] [3] e [4] explicam os conceitos de HOTP e TOTP, já discutidos na disciplina no material sobre Hash, Mac e Criptografia Autenticada.



Vamos usar o recurso de 2FA (*Two-factor authentication*), nesse caso, o código numérico disponibilizado via aplicação (ver Exemplo 13), simulando a execução de um aplicativo como Authy ou Google Authenticator. Você irá implementar uma aplicação parecida com o "Authy ou Google Authenticator" para fornecer um código de autenticação para ser o 2º fator. Cada usuário deve ter um "secret key" diferente cadastrado para obter o segundo fator

O arquivo que guarda as senhas irá usar os seguintes mecanismos: SCRYPT para guardar o nome do usuário e GCM para cifrar a senha do usuário:

Nome do usuário	Senha	Salt
SCRYPT do nome1	GCM da senha1	ac1e37bfb15599e5f4
SCRYPT do nome2	GCM da senha2	20814804c1767293b
...		

O IV e a chave (ou chaves) não devem ser guardados em nenhum lugar. Esses parâmetros devem ser derivados usando o PBKDF2. Para criar um salt específico para cada usuário, você deve usar o sorteio e guardar o salt no arquivo. O salt pode ser guardado sem criptografia.

A aplicação deve permitir as seguintes funcionalidades, que podem ser as opções do seu menu:

1) Cadastrar dados do cliente (logins/senhas) – deve ser permitido ter vários clientes (Alice, Pedro, João, Maria).
2) Gerenciar a funcionalidade de segundo fator de autenticação;
3) Autenticar o cliente com login e senha;
4) Autenticar o cliente com o segundo fator de autenticação, implementando o TOTP, fazendo o papel de Authy ou Google Authenticator.

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

i. **Não é permitido usar** chave e IV armazenados em variáveis globais ou de ambiente no momento da decifragem.

ii. Para gerar as chaves/IVs deve ser usado o PBKDF2. Deve ser usada criptografia autenticada para cifragem e decifragem da senha.

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

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

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. 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.**

Desafio (acrescenta nota na tarefa): Implementar/usar o TOTP que usa o SHA-256 como função do HMAC.

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/docs/libraries

[8]. https://blog.trezor.io/why-you-should-never-use-google-authenticator-again-e166d09d4324

[9]. https://hideez.com/blogs/news/fido2-explained

[10]. https://www.facebook.com/help/1167409720358611?helpref=faq_content

[11]. https://surepassid.com/how-we-secure/authenticators-fido-security-keys/

[12]. https://www.socinvestigation.com/how-fido-makes-passwordless-authentication-works/