PDFWebSigner: posicionamento de estampas de assinaturas digitais em documentos PDF

Maurício El Uri[1], **Diego Kreutz**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)

mauricioeluri@gmail.com, diegokreutz@unipampa.edu.br

Resumo. *O pyHanko é uma ferramenta de linha de comando e código aberto que permite usuários realizarem assinaturas digitais em documentos eletrônicos, oferecendo suporte nativo ao formato PDF. Entretanto, além de operar via linha de comando, um dos desafios para os usuários é o posicionamento da estampa da assinatura digital, que é um processo manual e trabalhoso. Para resolver este problema, propomos o PDFWebSigner, que consiste em uma solução acompanhada de um front-end Web que permite o posicionamento visual e simples da estampa da assinatura digital no documento PDF. No back-end, utilizamos o pyHanko e um certificado PKCS#{11,12} (e.g., da ICPEdu) do usuário para posicionar a estampa visual e assinar digitalmente o PDF. Na versão atual, a solução é disponibilizada ao usuário final como uma aplicação monolítica, simples de utilizar, através de uma imagem Docker.*

1. Introdução

A ferramenta e-certsDS [Uri et al. 2021] foi concebida como uma solução automatizada capaz de gerar, publicar e validar lotes de certificados eletrônicos, como os emitidos para participantes de eventos. Os certificados eletrônicos são assinados utilizando uma assinatura digital no padrão OpenPGP (*Pretty Good Privacy*)[1] e um segundo código de verificação de autenticidade baseado na primitiva criptográfica HMAC (*Hash-Based Message Authentication Codes*) [Krawczyk et al. 1997].

A substituição das assinaturas GnuPG (GNU *Privacy Guard*)[2] por assinaturas digitais utilizando certificados ICPEdu (Infraestrutura de Chaves Públicas para Ensino e Pesquisa)[3] ou ICPBrasil[4] pode ser realizada através de ferramentas como a pyHanko[5], que permite a utilização de certificados nos padrões PKCS#{11,12} (*Public Key Cryptography Standards*) [Moriarty et al. 2014]. Através de um arquivo de configuração personalizado no formato YAML (*human-readable data serialization language*)[6], por padrão o `pyhanko.yml`, o pyHanko pode ser utilizado para assinar lotes de documentos utilizando certificados digitais padrão ICPEdu ou ICPBrasil, por exemplo. Entretanto, o posicionamento da estampa da assinatura digital no documento PDF [Hardy et al. 2017] é manual, por meio de coordenadas cartesianas ([X1,Y1] e [X2,Y2]), o que é trabalhoso e complicado na perspectiva do usuário final. O processo poderia até ser relativamente

[1]https://www.openpgp.org
[2]https://www.gnupg.org
[3]https://pessoal.icpedu.rnp.br
[4]https://www.gov.br/iti/pt-br/assuntos/icp-brasil
[5]https://github.com/MatthiasValvekens/pyHanko
[6]https://yaml.org/spec/

simples se a estampa da assinatura digital fosse posicionada sempre na mesma página e posição cartesiana do PDF, o que não é o caso para maioria dos documentos PDF digitalmente assinados.

É importante destacarmos que a solução centralizada do Gov.br, o `Assinador.iti.br`, permite ao usuário posicionar a estampa da assinatura na página e local (coordenadas) desejado. Diferentemente desta plataforma, nossa intenção é oferecer uma solução centrada no usuário, isto é, onde o certificado digital utilizado para assinar digitalmente documentos PDF permanece sempre em posse do usuário. Adicionalmente, queremos entregar uma solução baseada inteiramente em tecnologias abertas e livremente disponíveis, como é o caso do pyHanko. Vale ressaltar que existem outras soluções similares, como BatchPDFSign[7], MyPDFSigner[8], pkcs11-tools[9] e pam_pkcs11[10]. Entretanto, nenhuma dessas ferramentas (ou bibliotecas) possui a configurabilidade e a flexibilidade do pyHanko para a assinatura de documentos PDF, como a adição e o posicionamento de estampas visuais, múltiplos templates pré-configurados de estampas e assinaturas e o suporte a múltiplas assinaturas no mesmo documento.

Nosso objetivo central é desenvolver uma solução Web (i.e., *front-end* + *back-end*) que seja capaz de (a) permitir o posicionamento das estampas das assinaturas digitais no documento PDF e (b) assinar o documento utilizando o pyHanko e certificados ICPEdu, disponíveis gratuitamente para todos os usuários das instituições usuárias da RNP (Rede Nacional de Ensino e Pesquisa)[11]. Como contribuições técnicas podemos destacar: (a) projeto, implementação e disponibilização de uma arquitetura de software, denominada PDFWebSigner, que permite ao usuário selecionar graficamente o posicionamento da estampa da assinatura digital, e (b) integração da solução com o pyHanko para viabilizar a utilização de assinaturas digitais a partir de certificados digitais PKCS#{11,12}. Nas próximas seções detalhamos a arquitetura e a implementação do PDFWebSigner.

2. PDFWebSigner

Nesta seção apresentamos a arquitetura e a implementação do PDFWebSigner.

2.1. Arquitetura

A Figura 1 ilustra a arquitetura do PDFWebSigner. A solução que é dividida em três componentes básicos, o *front-end*, o *back-end* e o repositório de configurações e certificados.

O *front-end* representa a interface com o usuário, contendo duas funcionalidades essenciais. Primeiro, o usuário carrega um arquivo PDF para assinar. Este arquivo é apresentado em um componente de visualização e edição, onde o usuário pode escolher a posição da estampa da assinatura digital utilizando os controles de navegação e o mouse.

Uma vez selecionada a localização visual da estampa da assinatura digital, é coletada a informação de posicionamento (i.e., número da página e coordenadas cartesianas) e repassada para o *back-end*. O próximo passo é a assinatura digital propriamente dita, com estampa visual, que é realizada utilizado o pyHanko. Para assinar o documento, são

[7] `https://github.com/jmarxuach/BatchPDFSign`
[8] `https://www.kryptokoder.com`
[9] `https://github.com/Mastercard/pkcs11-tools`
[10] `https://github.com/OpenSC/pam_pkcs11`
[11] `https://www.rnp.br`

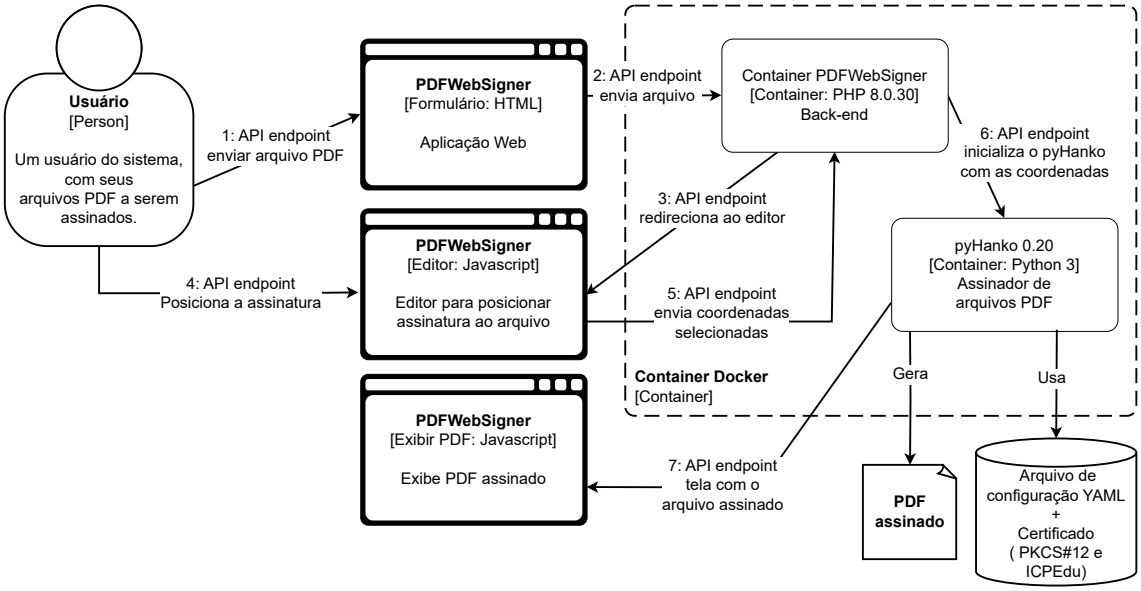

Figura 1. Diagrama Arquitetural do PDFWebSigner

necessárias quatro informações básicas, (1) o posicionamento da estampa no PDF, (2) o arquivo de configuração do pyHanko, (3) o caminho do arquivo do certificado do usuário e (4) a senha do certificado para realizar a assinatura. Essas informações são utilizadas somente no momento da assinatura e mantidas fora do *back-end* (e.g., num volume ou diretório específico do sistema hospedeiro do usuário).

É importante destacarmos o PDFWebSigner pode ser utilizado como uma aplicação monolítica na máquina hospedeira do usuário, através de uma máquina virtual ou um contêiner Docker[12], por exemplo. Entretanto, sua arquitetura Web, baseada em componentes *front-end* e *back-end* (cliente/servidor) bem definidos e uma API básica de interação, permite que a aplicação seja instanciada como um serviço online. Neste caso, os principais desafios estão relacionados aos certificados digitais dos usuários, que precisariam ficar armazenados ou ser enviados até o servidor online (*back-end*) do serviço. Um cenário de serviço online pode levar a alguns desafios adicionais de segurança e requerer a utilização de segurança assistida por hardware [Coppolino et al. 2019], através de tecnologias como Intel SGX para proteger os certificados dos usuários sendo utilizados no *back-end* [Antunes et al. 2018]. Alternativamente, seria possível pensarmos também em uma arquitetura onde o serviço de posicionamento das assinaturas esteja disponível online, mas a assinatura digital seja realizada, de fato, na máquina hospedeira do usuário. O papel do serviço online poderia ser entregar as coordenadas de posicionamento e oferecer opções de estampas de assinaturas digitais (e.g., uma galeria), por exemplo.

2.2. Implementação

Para implementar o *front-end*, utilizamos as bibliotecas: (a) `Bootstap` (versão 4.5.2) para o design da interface, (b) `Jquery` (versão 3.5.1) para uma melhor manipulação do JavaScript, (c) `Sweetalert` (versão 2.1.2) para notificações *pop-ups*, (d) `PDF` (versão 2.6.347) e `pdf.worker` (versão 2.6.347) para a visualização do PDF, e (e)

[12] `https://www.docker.com`

`pdfAnnotate` (versão 1.0.0) e `Fabric` (versão 4.3.0) para a coleta das coordenadas do mouse através da função `pdfAnnotate`. As coordenadas coletadas são enviadas ao *back-end* para o processo de assinatura do documento utilizando o pyHanko.

O *back-end* foi implementado utilizando a linguagem `PHP` (versão 8.0.30). O principal *endpoint* é um método `POST` contendo as coordenadas coletas e o arquivo PDF. Esses dados fazem parte das entradas do pyHanko para produzir a estampa visual e a assinatura digital propriamente dita.

Além do PDF e das coordenadas coletadas, o pyHanko utiliza também o arquivo de configuração YAML e o certificado .p12 do usuário para assinar digitalmente o documento. Com relação à estampa da assinatura digital, podem ser definidos o tipo de fonte, o tamanho da fonte, o formato da data, as margens e o nível de transparência.

O *front-end* e *back-end* do PDFWebSigner são disponibilizados por meio de uma imagem Docker. O repositório GitHub contém um `Dockerfile` baseado no Debian 11 (bullseye) e servidor Web `Apache` (versão 2.4.56).

2.3. Funcionamento

Primeiramente, é necessário clonar o repositório GitHub do PDFWebSigner[13] e inicializar o sistema. O usuário pode utilizar o *script* `./pdfwebsigner.sh` para informar o diretório (ou volume) compartilhado que contém o arquivo de configuração do pyHanko e o certificado digital e instanciar um contêiner Docker do PDFWebSigner. O usuário necessita configurar o arquivo do pyHanko uma única vez, informando o caminho do certificado digital e a senha (e.g., parâmetro `pfx-passphrase`) necessária para realizar a assinatura digital. Atualmente, por questões de simplicidade e praticidade, a senha fica armazenada no arquivo de configuração do pyHanko.

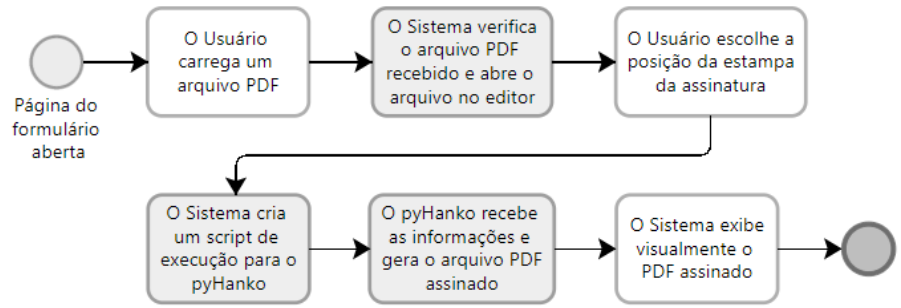

Figura 2. Fluxo de utilização do PDFWebSigner

Uma vez instanciado o PDFWebSigner, o fluxo resumido de utilização é ilustrado na Figura 2. O usuário acessa a interface Web[14] (e.g., `https://localhost`) e realiza o *upload* do arquivo PDF, como ilustrado na Figura 3. A ideia é a interface ser o mais minimalista, intuitiva e prática possível. Futuramente, pretendemos incluir também a opção *drag-and-drop* para o usuário.

[13] `https://github.com/WRSeg23/PDFWebSigner`

[14] O próprio *script* `./pdfwebsigner.sh` irá tentar detectar e abrir automaticamente um navegador apontando para a URL do contêiner Docker.

Figura 3. *Upload* do arquivo PDF

No terceiro estágio, o PDF é apresentado para o usuário em modo visualização (vide Figura 4). O usuário pode escolher a página e o local onde será inserida a estampa visual da assinatura digital. Na versão atual, disponibilizamos apenas um único estilo pré-configurado de estampa visual para o usuário. Entretanto, se o usuário quiser mudar a estampa, ele pode alterar o arquivo de configuração do pyHanko. Futuramente, pretendemos incluir a opção de galeria de estilos de estampas, permitindo ao usuário escolher o estilo de sua preferência na interface visual do sistema.

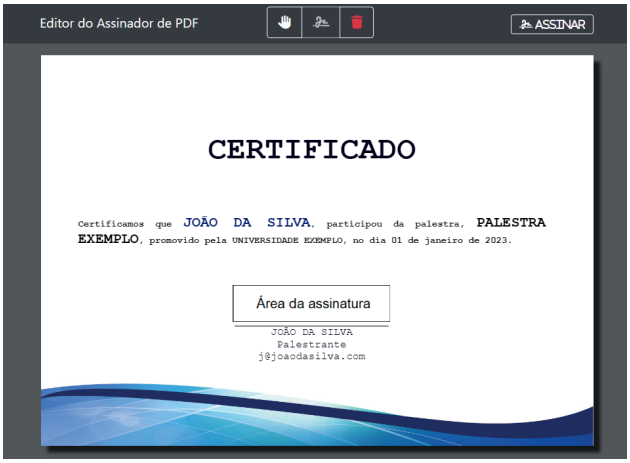

Figura 4. Posicionamento da assinatura com o editor

Na tela de visualização do PDF há o marcador "Área da assinatura", que pode ter seu posicionamento, altura e largura ajustados. Ao terminar o posicionamento, basta clicar em "ASSINAR" para a aplicação coletar as coordenadas de posicionamento e convertê-las para o formato do pyHanko.

Finalmente, as coordenadas da estampa visual são enviadas ao *back-end* e transformadas em um parâmetro de execução do pyHanko. O pyHanko adiciona a estampa visual e assina digitalmente o arquivo PDF. O documento, assinado digitalmente, é disponibilizado tanto no *front-end* quanto no diretório (ou volume) do sistema hospedeiro, compartilhado com o contêiner Docker.

Na primeira avaliação de funcionamento e usabilidade, utilizamos dois usuários especialistas, isto é, que possuem bom conhecimento em instalação, configuração e utilização de sistemas, incluindo trabalhar com contêineres Docker. Os dois usuários não tiveram dificuldades em instanciar e utilizar a solução para assinar digitalmente documentos PDF utilizando certificados ICPEdu. Futuramente, planejamos ampliar a

avaliação da solução incluindo uma gama maior e mais diversificada de usuários nos testes. Pretendemos também realizar avaliações específicas de UI, UX [Obrist et al. 2009, Maia and Furtado 2016] e aspectos de segurança (e.g., análise técnica das bibliotecas utilizadas na implementação do sistema).

3. Considerações Finais

O PDFWebSigner resolve um problema prático de utilização do pyHanko para assinatura digital de documentos PDF, que consiste no posicionamento (página e coordenadas cartesianas) das estampas visuais das assinaturas digitais. O PDFWebSigner torna possível ao usuário fácil e rapidamente utilizar o pyHanko com certificados, como os do ICPEdu, para assinar digitalmente documentos PDF quaisquer. Diferentemente do `Assinador.iti.br`, o PDFWebSigner é centrado no usuário, isto é, o certificado digital fica sempre em posse e na máquina do usuário. Em termos práticos, a solução acaba sendo também mais simples e mais prática, pois não necessita de múltiplos fatores de autenticação (e.g., código SMS (*Short Message/Messaging Service*), que pode eventualmente demorar para chegar[15]) para assinar digitalmente um documento PDF, como no `Assinador.iti.br`.

Como trabalho futuros, podemos destacar: (a) incluir opção de assinatura em lote de documentos PDF; (b) permitir a definição e utilização de diferentes estampas visuais para a assinatura digital; (c) avaliação de robustez e usabilidade utilizando um número estatisticamente mais representativo de usuários; (d) mitigar riscos de segurança associados a utilização da senha e do certificado digital propriamente dito; (e) mitigar riscos associados aos componentes de software utilizados na implementação da solução; e (f) integrar com soluções de assinatura em lote de certificados como a e-certsDS [Uri et al. 2021].

Referências

Antunes, F., Garcia, F., and Kreutz, D. (2018). SeguraAí: confidencialidade de dados sensíveis com SGX. In *ERRC / WRSeg*. SBC. `https://github.com/kreutzd/arxiv/blob/main/wrseg2018_opensgx_ws.pdf`.

Coppolino, L., D'Antonio, S., Mazzeo, G., and Romano, L. (2019). A comprehensive survey of hardware-assisted security: From the edge to the cloud. *Internet of Things*, 6:100055.

Hardy, M., Masinter, L., Markovic, D., Johnson, D., and Bailey, M. (2017). The application/pdf Media Type. RFC 8118.

Krawczyk, D. H., Bellare, M., and Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. RFC 2104.

Maia, C. L. B. and Furtado, E. S. (2016). A systematic review about user experience evaluation. In Marcus, A., editor, *Design, User Experience, and Usability: Design Thinking and Methods*. Springer International Publishing.

Moriarty, K., Nystrom, M., Parkinson, S., Rusch, A., and Scott, M. (2014). PKCS #12: Personal Information Exchange Syntax v1.1. RFC 7292.

Obrist, M., Roto, V., and Väänänen-Vainio-Mattila, K. (2009). User experience evaluation: Do you know which method to use? In *CHI*.

Uri, M., Vargas, L., and Kreutz, D. (2021). e-certsDS: Certificados eletrônicos com assinatura digital. In *Anais do XXI SBSeg*, pages 66–73.

[15]É interessante destacarmos que observamos empiricamente diversos casos de demora na entrega ou não entrega do SMS do `Assinador.iti.br`, como em momentos de indisponibilidade de telefonia.