Exploring Digital Signatures Secrecy in Web-Platform: Client-Side Cryptographic Operations.

Wellington Fernandes Silvano[1], Gabriel Cabral[1], Lucas Mayr[1], Frederico Schardong[1,2], Ricardo Custódio[1]

[1]Laboratório de Segurança em Computação (LabSEC) – Departamento de Informática e Estatística (INE) – Universidade Federal de Santa Catarina – Brazil

[2]Instituto Federal do Rio Grande do Sul (IFRS) – Brazil

{wellington.fernandes,lucas.mayr}@posgrad.ufsc.br
frederico.schardong@rolante.ifrs.edu.br, ricardo.custodio@ufsc.br

Abstract. *Online signature platforms confront critical security challenges, notably exposing sensitive documents to third-party applications. This paper presents a novel client-side cryptographic model that enhances document secrecy and key management by performing cryptographic operations within the user's browser. By employing one-time certificates, our model eliminates document uploads, reducing the risk of leakage and private key compromise. Aligned with Claude Shannon's information theory, our approach ensures robust secrecy while remaining compatible with existing digital signatures. Our implementation demonstrates practical performance, offers a significant advancement in secure digital signatures, addressing vulnerabilities in traditional web-based platforms.*

1. Introduction

The proliferation of online signature platforms in recent years has significantly enhanced user experience by enabling efficient and convenient digital document signing. However, this convenience comes at the cost of increased security risks. Notably, handling sensitive documents and managing private keys on these platforms pose significant vulnerabilities, particularly when examined through the lens of Claude Shannon's secrecy principles [Shannon 1949]. The common practice of uploading documents and granting access to private keys, or even delegating signing authority as a proxy, exposes documents to potential attacks and compromises data integrity and confidentiality.

Beyond the client's concerns about confidential document exposure, platform operators face substantial liabilities. Leaks can result in severe penalties under data protection regulations such as the EU GDPR [European Union 2018] and Brazil's LGPD [Brasil 2018]. Furthermore, platforms must comply with a complex legal framework encompassing: (i) guarantees of correspondence confidentiality; (ii) industrial property rights [Brazil 1996]; (iii) professional secrecy obligations [OAB 2015, CFM 2010]; (iv) espionage laws [United States 1917, United Kingdom 1989]; and (v) varying degrees of information classification under Freedom of Information (FOI) acts [United Kingdom 2000, Brazil 2011]. These laws impose strict access controls, often categorized as *top secret*, *secret*, or *reserved*, with specific terms varying by jurisdiction.

In his seminal 1949 work, *Communication Theory of Secrecy Systems*, Claude Shannon established the foundations for both information theory and modern cryptography

[Shannon 1949]. Shannon identified three general types of secrecy systems, which we hereafter name Shannon-Secrecy Types (SST): **SST-1** focus on hiding the existence of the message; **SST-2** where specialized equipment or special technique to retrieve the message; and **SST-3** those obscuring message content through ciphers and codes, assuming complete adversary capabilities. Based on these principles, we propose a model to enhance secrecy in digital signature web-platforms by executing critical cryptographic operations client-side within the user's browser.

Our model leverages One-Time Certificates to provide secure digital signatures while optimizing user experience through streamlined key management and a simplified signing process [Mayr et al. 2023, Mayr et al. 2024]. By executing all cryptographic operations within the client-side browser, we empower users with complete document control, substantially reducing the risk of document and private key exposure. This approach enhances security by mitigating vulnerabilities associated with sensitive document leakage and compromised key management.

We present a comprehensive exploration and practical implementation of a web-based document signing and verification system that exclusively operates on the client side. By examining existing research on document integrity, authenticity, and non-repudiation through digital signatures (Section 2), we propose a novel key management strategy compatible with legacy Public Key Infrastructures (PKIs) (Section 3). We also explore the current landscape of signature web-platforms, identifying their limitations and security challenges (Section 4). We introduce a tailored One-Time Certificate mechanism designed for heightened secrecy (Section 5), addressing implementation challenges, security implications, and performance trade-offs (Sections 6 and 7). The document concludes the study by summarizing the key contributions and outlining the challenges and future work for the proposed client-side cryptographic model using One-Time Certificates (Section 8).

2. Related Work

This section provides an overview of existing research on safeguarding the integrity of sensitive documents using digital signatures. We examine studies that explore alternative key management strategies to traditional PKIs.

[Luan et al. 2015] highlight the complexities of managing confidential documents in isolated environments where internet-based PKI systems are unfeasible. Their model uses an offline server for signing and encrypting documents, requiring physical document transfer via USB drives to ensure confidentiality. This approach highlights the unique challenges of securing information without internet access and effectively mitigates risks tied to online PKI systems.

[Choi et al. 2017] underscore the growing challenges of securing confidential documents in cloud environments. Their Doc-Trace framework addresses this issue by implementing document tracing at the hypervisor level, a software layer below the virtual machines' operating systems. Steganographic marks embedded within documents enable the detection of content modifications and the logging of access events. However, the solution's effectiveness is limited to identifying unauthorized alterations, as it cannot prevent intentional data exfiltration by malicious insiders. Doc-Trace is thus valuable for maintaining comprehensive audit trails but falls short as a standalone defense against deliberate data breaches. This highlights the ongoing need for multifaceted security approaches

to safeguard sensitive information in cloud environments.

[Shatnawi et al. 2017] address the critical challenge of maintaining integrity and non-repudiation for sensitive offline documents, including medical records, legal reports, and financial assets. Their framework embeds digital certificates directly within documents, enabling detailed tracking of modifications, including author, device, and timestamp. Integrated as a Microsoft Word plugin, this approach offers self-sufficiency in secure environments. However, the reliance on embedded certificates may hinder compatibility with traditional verification systems expecting externally accessible certificates. While adhering to X.509 and RFC 5280 certificate standards [Boeyen et al. 2008], the solution faces limitations in revocation verification and interoperability. These factors highlight the need for a balanced approach to document security, considering both robust protection and practical usability across diverse operational contexts, especially for frequently modified, highly sensitive documents.

[Aciobăniței et al. 2024] address the critical need for secure and private remote qualified electronic signatures (RQES) for confidential documents. Their proposed framework leverages a Trust Service Provider (TSP) to digitally sign document hashes, preserving content confidentiality. A browser plugin facilitates hash generation, user authentication, and secure hash transmission to the TSP. The resulting digital signature and TSP certificate are embedded within the document. To ensure long-term integrity and validity, digital signatures, certificates, and user metadata are registered on the Ethereum blockchain [Foundation 2024]. While this approach protects content privacy and aligns with regulations like eIDAS [União Europeia 2014], it introduces challenges. The document itself does not inherently identify the signer, requiring external consultation. Additionally, potential conflicts may arise between the certificate's temporal validity and the blockchain's long-term data storage. Balancing security, privacy, and practical considerations remains essential for effective online document management.

[Mayr et al. 2024] introduce a novel key management approach as an alternative to traditional PKIs. Their core innovation is the One-Time Certificate, where signature verification by a certificate is restricted to a single document through an embedded cryptographic hash of the target document inside the certificate. This model generates a key pair for each signed document, thus creating a new Certificate Signing Request (CSR) for each new signature. After the certificate is issued and the document is signed, the private key of the key pair is deleted. The document hash is incorporated into an X.509 extension before submission to the Certificate Authority (CA) for issuance. Embedding the hash within the certificate ensures its immutability and directly links the certificate to the specific document. A key advantage of this approach is the removal of ongoing certificate and key management burdens for end-users, reducing costs and technical complexities. Unlike traditional methods requiring persistent certificate and key pair maintenance, One-Time Certificates mandate the creation of a new key pair and certificate for each document, enhancing security and simplifying the signing process.

3. Background

This section provides foundational knowledge of digital signatures to support reader comprehension.

3.1. Digital Signatures

Digital signatures are cryptographic mechanisms that assert the integrity and authenticity of digital data. Functioning as electronic counterparts to handwritten signatures, they safeguard against tampering and impersonation. By employing asymmetric cryptography, digital signatures establish proof of origin, identity, and document status. The signing process involves a private key held by the signer to generate a signature, which can then be verified by anyone using its public key counterpart [Goldreich 2001].

The key generation algorithm produces an asymmetric key pair consisting of a private key sk and a public key pk, formally expressed as $\text{Gen}(1^\lambda) \rightarrow (sk, pk)$, where λ is the security parameter determining the computational complexity of the algorithms.

Typically, a message m is first run through a cryptographic hashing algorithm, producing a fixed-length digest h. Then, the private key sk is used to compute a digital signature σ using a digital signature scheme such that $\text{Sign}(sk, h) \rightarrow \sigma$.

Lastly, signature verification checks whether a given public key pk truly is the counterpart to the private key sk used in the signature process using the message m and signature σ as part of the comparison process, i.e., $\text{Verify}(pk, m, \sigma) \rightarrow \{0, 1\}$; where a result of 1 indicates a valid signature, and 0 indicates an invalid one.

3.2. Digital Certificate

Introduced by [Kohnfelder 1978], a digital certificate is essentially a binding of a key pair to an entity. The process involves an individual applying to a CA which, upon verifying identity, issues a digital certificate. As defined by [Myers et al. 1999], this certificate adheres to the X.509 standard, functioning as a structured digital file. It contains metadata about the certificate owner, including name, email, expiration date, address, and potentially other extensions. Crucially, it incorporates the owner's public key and a digital signature attesting both key ownership and the certificate's authenticity.

This signature, along with the certification path, may be traced back to a trusted root CA, ensuring the certificate's legitimacy. To obtain a digital certificate, an entity must generate a CSR and submit it to a CA for issuance. Created after generating the key pair, the CSR includes the public key and essential metadata needed for the certificate. Importantly, the CSR is authenticated using the entity's private key to verify its legitimacy. Two primary formats are commonly used to represent digital certificates: PKCS#7 [Nystrom and Kaliski 2000] and PKCS#12 [Moriarty et al. 2014].

4. Digital Signature Web-platforms

Traditional digital signature applications, such as Adobe Acrobat [Adobe Inc. 2024], operate locally on a user's device, offering PDF editing, creation, and document signing features. These applications require a valid digital certificate accessible on the device or via external media, such as a smart card. For the certificate to be recognized as valid, the certificate chain, including root and intermediate certificates, must be installed and trusted. Users are responsible for managing their certificates and ensuring the certification chain is properly configured. Additionally, they must follow procedures to mitigate risks, such as certificate or device loss, including revoking compromised certificates with the issuer.

In contrast to traditional desktop applications, numerous digital signature platforms are accessible via web browsers. These cloud-based solutions offer the convenience of

signing documents from any internet-connected device without requiring local software installation. Web platforms simplify the signing process by eliminating the need for users to manage certificates and private keys. However, this approach introduces new security challenges as the platform provider stores and manages sensitive cryptographic materials. While these platforms often employ medium-term certificates reused across multiple users, the centralized management of keys and certificates necessitates robust security measures to protect against unauthorized access and data breaches. Examples of such platforms include [Cryptomathic 2023, Bit4id 2021, Ascertia 2018, NextSense 2023, DigitalSign 2023, UFSC 2019, Brazil, Economy Ministry 2021].

In our study of signature web-platforms, we meticulously examined the often implicit flowcharts underpinning their operations. Figure 1 presents a simplified overview of the general digital signature process, assuming a registered user. The standard sequence of events is as follows: ⓐ The user logs in to the platform; ⓑ The user uploads the document intended for signing; ⓒ The user selects either platform-based or user-owned certificate signing. For platform-based signing, a platform-generated X.509 certificate and key pair, often managed by a Hardware Security Module (HSM). For user-owned certificate signing, the user can upload a new X.509 certificate or utilize an existing one stored on the platform; d_1 The document's hash is computed, following a process similar to that outlined in [Aciobăniței et al. 2024]. This hash is then signed using a backend library, and the resulting signature is embedded in the document alongside the platform's certificate; d_2 Alternatively, if the user opts for their own certificate, the private key and certificate are made accessible via the web platform; ⓔ The user's certificate and private key are transmitted to the backend signature library; ⓕ The document is retrieved; and ⓖ The document is signed using the user's private key.

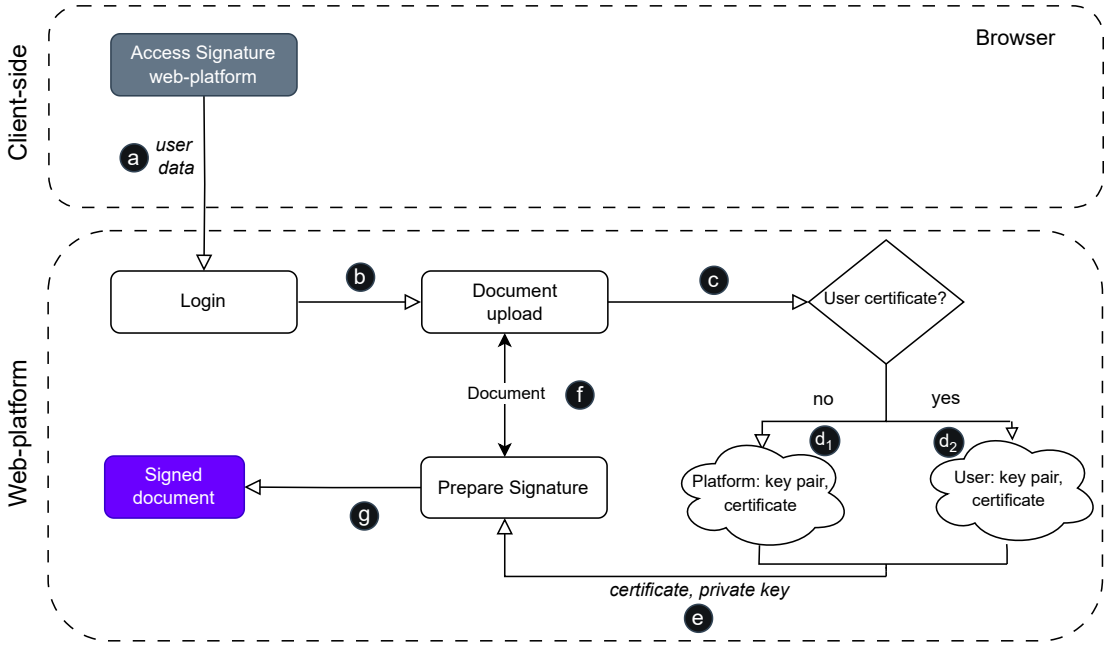

Figure 1. Simplified flowchart of the digital signature process on signature web-platforms.

Regardless of whether option d_1 (platform-based signing) or d_2 (user-owned

certificate signing) is chosen, the document is inherently exposed to the platform, even when using a secure communication channel. Furthermore, option d_2 also exposes the user's private key to the platform. Conversely, while option d_1 does not compromise the user's private key, it introduces the concept of a proxy signature, where the platform effectively signs on behalf of the user. This type of signature is only valid when verified by the platform itself, as the user's identity is neither independently authenticated nor explicitly associated with the signed document via a digital certificate.

The examined signature web-platforms lack transparency regarding document storage methods. While some platforms may employ encryption, the encryption keys are likely managed by the platform itself and stored within the database or archive infrastructure.

Consequently, the digital signature platforms raise significant concerns when analyzed through the lens of Shannon's secrecy types. Firstly, a violation of **SST-1** occurs as the mere existence of a document is revealed to entities beyond the signatories. This vulnerability exposes the risk of targeted attacks on the platform's infrastructure and its trusted operators, providing adversaries with opportunities to exploit weaknesses through phishing or social engineering. Furthermore, the absence of robust obfuscation mechanisms leads to a violation of **SST-2**, which occurs when sensitive metadata, such as file names, timestamps, signatory details, or file size, are not adequately protected, allowing adversaries to infer critical information about the document's nature or context without direct access to its content. Finally, the limited transparency regarding document storage suggests a high probability that **SST-3** is compromised, meaning adversaries could potentially access or deduce the document's content if they breach the platform's infrastructure. Although many platforms implement basic access control, they often fail to enforce robust key management protocols, prioritizing user experience over security. One possible improvement is the implementation of *Identity-Based Encryption (IBE)* [Boneh and Franklin 2001], which simplifies key management. However, this approach introduces the risk that a compromised master key could jeopardize the security of the entire system.

5. Proposed Model

This paper presents a novel model for enhancing document secrecy within online signature platforms, inspired by the One-Time Digital Certificates (OTC) approach [Mayr et al. 2024]. We propose a significant enhancement to bolster security: enabling users to generate cryptographic key pairs and CSRs directly within their web browsers. This empowers users to independently request certificates from a CA that also functions as the signature web-platforms, though this role can be delegated.

The proposed model is illustrated in Figure 2, in contrast to the traditional model shown in Figure 1. This approach significantly increases client-side processing. The web-platform provides a browser-based library encompassing all the necessary functions for document signing. Once the user is authenticated, no sensitive data is transmitted to the platform; cryptographic operations occur exclusively within the user's browser.

Initially, ⓐ the user logs into the signature web-platform. Subsequently, ⓑ the platform transmits a signature library to the client's browser. ⓒ A document selection form is displayed within the web platform's front end; upon selecting a document, the form activates the signature library directly in the browser. ⓓ. The library generates a key pair, creates a document hash, embeds it within a CSR extension, and signs the CSR using

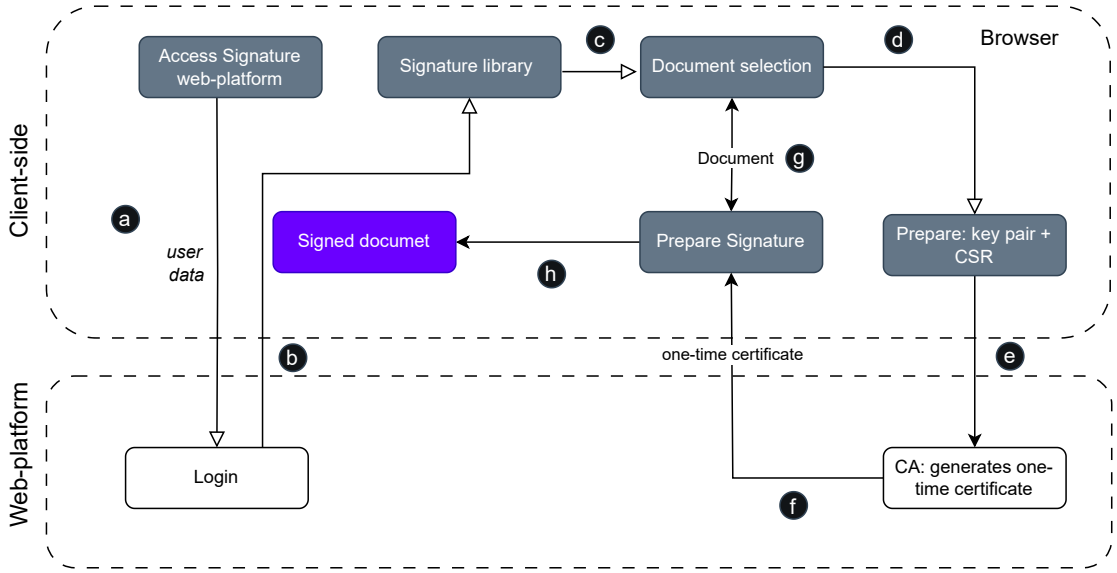

Figure 2. The proposed signature secrecy model for web-platforms.

the generated private key. ⓔ The CSR is sent to the CA (which, in this case, is the web platform). ⓕ The CA issues a one-time certificate, accompanied by the trust certification chain, and sends it to the client's browser. ⓖ The signature library retrieves the document and the one-time certificate with its trust chain, ⓗ proceeding to sign the document.

Our model requires user authentication only at step ⓐ, with optional additional verification at step ⓔ based on specific requirements. The identity provider might enforce multi-factor authentication to robustly establish user identity [Jacomme and Kremer 2021, Prabakaran and Ramachandran 2022, Perottoni et al. 2023]. Furthermore, the CA must employ robust key storage and certificate issuance mechanisms, such as HSMs, specifically designed for PKI environments [Barker and Barker 2018].

The digital document intended for signing remains within the execution environment alongside the generated key pair, with only the public key transmitted within the CSR. This ensures that the document never leaves the client environment, preserving its secrecy and ensuring **SST-1**, as neither the web platform nor any external entity possesses knowledge of the document being signed. Additionally, as the document never leaves the client environment, no sensitive metadata—such as file names, timestamps, or file size—is exposed, ensuring that adversaries cannot infer critical information about the document's nature or context, thereby preserving **SST-2**. Furthermore, since neither the document's content nor any portion of it is transmitted to the platform, cryptographic isolation is achieved. This aligns with Shannon's concept of a "true" secrecy system, ensuring compliance with **SST-3**. Even if an adversary gains access to the platform or intercepts signals, they are unable to deduce the document's content due to the use of robust cryptographic protocols confined entirely to the client environment. The independence from the platform's infrastructure removes a potential point of failure, further reinforcing that no information useful for deducing the document's content is ever exposed.

6. Model Implementation

We focus on signing PDF documents to validate the proposed model, adhering to ISO-32000 and PAdES standards [ISO 2020, ETSI 2024]. Leveraging a combination of *JavaScript* [Eich 1995] and *Next.js* [Vercel Inc. 2016] for frontend development, we execute cryptographic functions using the native *JavaScript Crypto library* [(SJCL) 2010], *pkijs* [GlobalSign and Ventures 2014], *asn1js* [Ventures 2013], and *Node-Forge* [Digital Bazaar 2010], specializing in key generation, CSR creation, and PDF signing. Addressing the challenge of incremental PDF updates required for signature validation, we develop processes to manipulate PDF structures directly within the browser.

This application allows users to choose and sign documents. Upon document selection, the JavaScript code computes the document's cryptographic hash, generates a CSR, and interacts with a CA microservice via HTTPS to obtain a one-time certificate. The signing process employs the RSASSA-PKCS1-V1.5 [Moriarty et al. 2016] and SHA-384 [Hansen and Eastlake 3rd 2011] algorithms for robust security, culminating in the signed document [Jonsson and Kaliski 2003].

6.1. Application Flow

Upon accessing the prototype, users are presented with the application's home screen. Clicking the authentication button redirects users to an external identity provider. Successful authentication returns users to the signature screen, where they can browse for documents on their computer. Clicking the *sign* button initiates the execution of the PDF signature pseudo-algorithm outlined in Algorithm 1.

Subsequently, a CSR containing the document's cryptographic hash is generated. A request, bundled with the CSR and the authentication token acquired during the authentication phase, is transmitted to the server. The server relays this request to the CA microservice. Upon validating the authentication token, the CA issues a one-time certificate. The server receives this certificate and returns it to the originating browser code. The signed document is then returned to the user as the signing process concludes.

Algorithm 1: Signing PDF documents in the client's browser using One-Time Certificate

```
1:  function SIGN(BytesPDF, userIdentity)
2:      BytesPDF ← PrepareDocumentForSignature(BytesPDF)
3:      Hash ← CalculateHashOfBytesToBeSigned(BytesPDF)
4:      KeyPair ← GenerateKeyPair()
5:      CSR ← CreateCSR(KeyPair, Hash, userIdentity)
6:      PKCS7 ← SendCSRToCA(CSR)
7:      PKCS12 ← CreatePKCS12(KeyPair.private, PKCS7)
8:      SignedBytes ← SignPDF(BytesPDF, PKCS12)
9:      return SignedBytes
10: end function
```

Each step of the pseudocode outlined in Algorithm 1, is described below:

`PrepareDocumentForSignature:` This function prepares the PDF document to accommodate the digital signature by adding the necessary placeholder without altering any pre-existing content that may already be signed.

`CalculateHashOfBytesToBeSigned:` This function computes the cryptographic hash of the PDF document.

`GenerateKeyPair`: This function generates a public-private key pair for the signing process.

`CreateCSR`: This function creates a CSR containing the public key, user identity, and document hash (in an extension). The CSR is signed with the private key.

`SendCSRToCA`: This function sends the CSR to the CA and receives a PKCS#7 containing the certificate chain.

`CreatePKCS#12`: This function converts the PKCS#7 into a PKCS#12 format using the private key.

`SignPDF`: This function signs the PDF document using the provided certificate, the BytesPDF, and the private key from the PKCS#12. Returns the signed PDF.

7. Results and Discussion

To assess the validity of the implementation, it is crucial to ensure that the signature created is interoperable, meaning it is recognized as a valid advanced signature by well-known verifiers. Additionally, since a significant portion of the code execution occurs within the client's browser, the execution times are also important for evaluating potential uses. We also discuss the advantages and disadvantages of the proposed model.

7.1. Signature Interoperability

This section assesses the compatibility of the generated digital signatures with industry-standard verification systems. The goal is to demonstrate adherence to current digital signature standards and ensure broad recognition across various platforms and applications. To demonstrate interoperability, we employed *pdfsig*, a command-line verification tool on Ubuntu [Poppler Utils 2024]. Successful validation using *pdfsig* confirmed compatibility with legacy verification systems, as illustrated in Figure 3.

```
pdfsig example_assinado.pdf
Digital Signature Info of: example_assinado.pdf
Signature #1:
- Signer Certificate Common Name: Alice Silva
- Signer full Distinguished Name: E = alice.silva@secrecy.com, CN =
   Alice Silva, C = BR
- Signing Time: Feb 09 2024 16:25:07
- Signing Hash Algorithm: SHA-384
- Signature Type: adbe.pkcs7.detached
- Signed Ranges: [0-23448], [43450-44167]
- Total document signed
- Signature Validation: Signature is Valid.
```

Figure 3. *pdfsig* tool usage example to verify PDF digital signatures.

The signature, issued to a fictitious user, Alice Silva, included metadata like the signer's email (provided by the identity provider). Verification revealed signature timestamp, hash function, and full document coverage, demonstrating backward compatibility

with traditional verification tools. The Adobe software also validated the signatures, further confirming interoperability and compliance. We successfully tested and verified multiple signatures using the same methods.

7.2. Client-Side Execution Times

This section analyzes the signature algorithm's performance within the client-side browser environment. We examine the impact of document size on signature execution time and its implications for user experience and system efficiency. To evaluate performance, we measure the total time required for five consecutive signatures on documents of varying sizes: 1.12 MB, 32.16 MB, and 60 MB. After each signature, the document was transferred to the next user (Figure 4). The evaluation was conducted on a Dell XPS system running Linux Ubuntu 22.04, equipped with 16GB of RAM and powered by an 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz CPU, using Google Chrome browser.

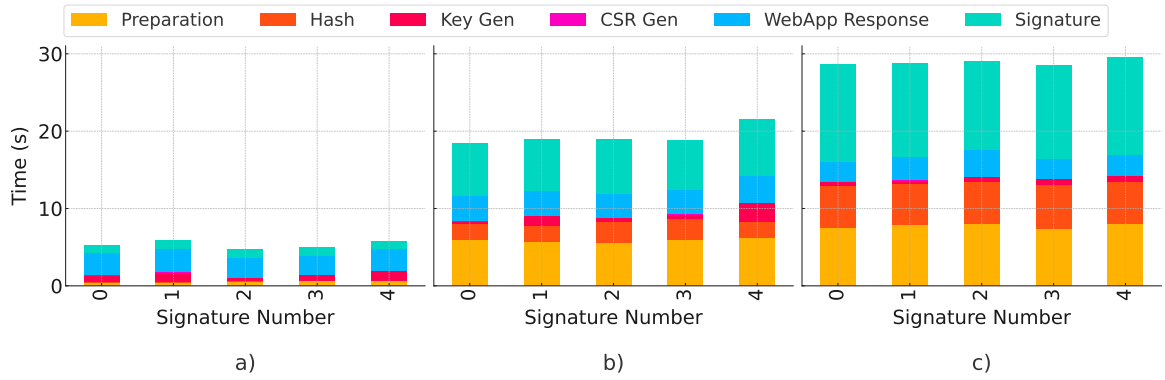

Figure 4. Total signature time for multiple signatures. Document sizes: a) 1.12 MB, b) 32.16 MB, c) 60 MB.

Analysis indicates that the total execution time is primarily influenced by file size rather than the number of subsequent signatures. As shown in Figure 4, average total execution times are approximately 5.3 seconds for 1.12 MB files, 19.3 seconds for 32.16 MB files, and 28.9 seconds for 60 MB files. The document preparation, hashing, and signing operations scale proportionally with file size. In contrast, key pair generation exhibits variability due to the inherent randomness in the process, with an average time of 836 ms (min: 139 ms, max: 2643 ms, std dev: 578 ms). The time required for CSR generation remains consistently low, averaging 23 ms, with negligible impact.The platform's certificate issuance time is stable at approximately 3 seconds.

Figure 5 illustrates the percentage contribution of each process to the total signature time. While key generation and microservice interaction become less influential with larger file sizes, the impact of signing, hashing, and preparation, increases proportionally to document size. Currently, the algorithm reads the PDF twice: once to locate the signature placement and again to apply the signature. This redundancy increases preparation and signature times. Future optimizations will streamline this process, reducing signature time to a negligible portion of the total.

7.3. General Analysis

A comparative table has been created to evaluate the key features of the proposed model in relation to traditional web-based signature platforms. Table 1 provides an analysis of

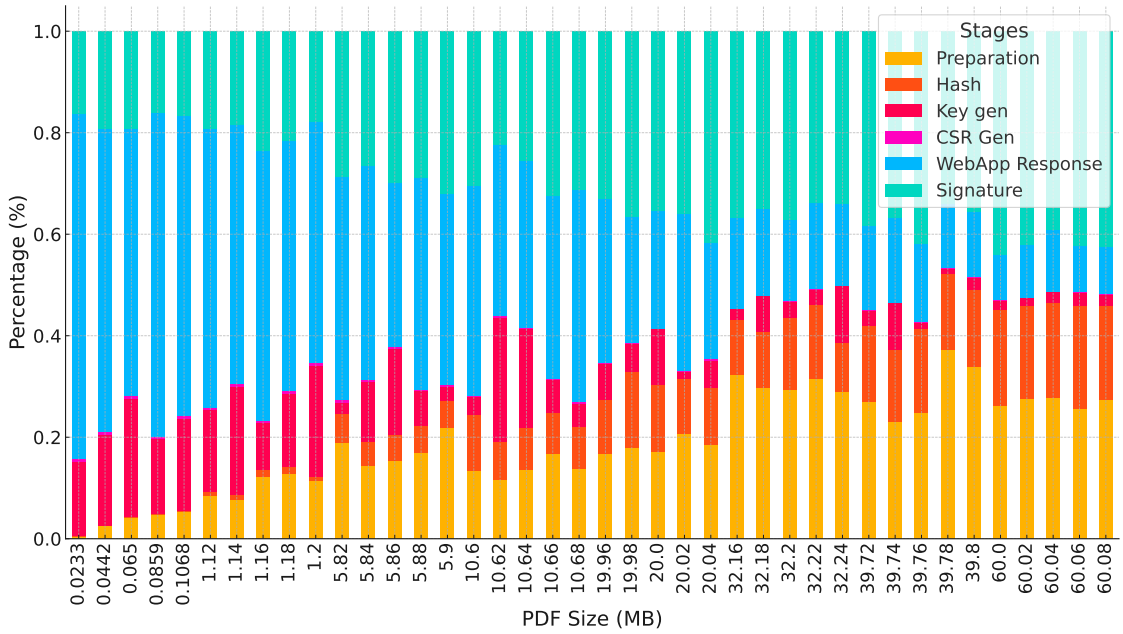

Figure 5. Percentage of Total Time by Process and Document Size.

seven critical aspects, including document secrecy, private key secrecy, key management complexity, certificate handling, support for multiple signatures, and performance across both large and small documents.

Table 1. Comparison between traditional signature web-platforms with our model

Feature	Traditional Web Platforms	Our model
Document Secrecy	Conditional	Unconditional
Private Key Secrecy	Conditional	Unconditional
Private Key Management	Complex	Simplified
Certificate Handling	Resource-Intensive	Streamlined
Multiple Signers	Link-Based	Out-of-Band
Signature Performance	Server Power Dependency	Client Machine Dependency
Large documents performance	Slower	Faster
Small documents performance	Faster	Slower

7.3.1. Document Secrecy

Traditional signature web-platforms provide conditional document secrecy, as they depend on access control mechanisms that are susceptible to attacks, especially those targeting the platform's infrastructure or trusted operators. The need to upload documents to these platforms exposes sensitive data to all violations of Shannon's secrecy principles (see Section 4).In contrast, our platform ensures unconditional document secrecy by eliminating the need to upload documents to a server. This removes the risk of secrecy violations, eliminating the platform as a potential point of failure 5. Although user-side vulnerabilities, such as browser exploits, remain a possible threat, the overall risk is reduced by restricting document handling to the client environment.

7.3.2. Private Key Secrecy

Platforms that utilize x.509 user certificates (as described in Section 4, step d_2) encounter challenges in maintaining the confidentiality of private keys, which are under the user's control rather than the platform's. This presents inherent risks, as the platform may inadvertently expose or mishandle private keys, leading to potential breaches of secrecy. In contrast, our proposal ensures unconditional private key secrecy by employing one-time signatures and generating CSR entirely on the client side. The only data transmitted to the signature web-platform is the cryptographic hash of the signed document. The certificates are used only once, there is no risk associated with long-term key storage (see Section 5).

7.3.3. Private Key Management

Ensuring that private keys are kept safe and secure is of the highest concern when dealing with digital signatures, especially considering that end users might not be as versed in security protocols as desired.Therefore, to lessen the burden of key management on their user base and as an attempt to increase overall security, some web signature platforms allow users to store their private keys in their platforms. These private key management delegation systems clearly violate Shannon's secrecy principles by exposing sensitive cryptographic material. In contrast, our model eliminates the need for continuous key management by leveraging one-time certificates with client-generated CSRs. In addition, the certificate is restricted to a single document and does not require further management as it will not be reused. Consequently, our approach eliminates the complexities traditionally associated with managing private keys while ensuring high security through robust authentication.

7.3.4. Multiple Signers

Traditional web signature platforms offer a convenient user experience, especially for documents requiring multiple signatures. Users upload the document, and signers receive a link or email to authenticate and sign, with the platform automating document distribution and signature collection. In contrast, the proposed model, while more secure, adds complexity. Each signer must transmit the document securely (e.g., via encrypted email or USB drive) and pass it to the next. A document management system can help streamline this process, but it introduces additional steps and potential delays compared to web signature platforms.

We remark that this constraint is present only in formats that do not allow parallel signatures, such as PDF, where each subsequent signature must sign its predecessor. Formats that do allow these types of signatures, such as XML, allow the original message to be sent to every signer simultaneously. After the signatures have been collected, they can be appended to the signed document with no issues in regard to signature ordering [ISO 2020].

7.3.5. Signature Performance and Document Size

Traditional platforms typically leverage powerful servers to expedite signing processes, often outperforming our approach for standard document sizes. However, network upload

times can become a bottleneck for exceptionally large files. In such cases, our method presents a potential advantage. For example, a preliminary analysis of Wikileaks data indicates that most classified documents are under 5MB, which our system can sign in approximately 7 seconds. Generally, signing times are comparable between the two.

8. Conclusion

This study introduces a novel client-side cryptographic model for digital signature platforms, leveraging one-time certificates to establish a secure environment for signing sensitive documents. By confining all cryptographic operations to the user's browser, our model safeguards document secrecy and eliminates the need for key pair management on the client side. Our findings demonstrate this approach's feasibility and substantial security advantages compared to traditional models. The key contributions include:

Enhanced Document Secrecy: protecting sensitive information by preventing document upload to external platforms;

Improved Private Key Management: eliminating the risk of private key exposure by managing key generation and usage locally;

User Experience: maintaining usability through seamless browser integration and simplified certificate management.

Adopting our model can significantly benefit digital signature platforms by mitigating liability and ensuring compliance with secrecy regulations. By offering heightened security, platforms can gain a competitive edge and foster user trust. Our approach aligns with legal secrecy and data protection mandates, safeguarding against potential penalties. Thus, our client-side cryptographic model offers a substantial improvement in digital signature technology, delivering enhanced secrecy while maintaining user convenience, particularly in key management and sign processes. Future research will aim to strengthen the model further, improving its robustness and adaptability to evolving security challenges.

Challenges and Future Work

While our model offers substantial security improvements, challenges arise in scenarios involving multiple signers and documents. Future research will prioritize optimizing usability, particularly for documents requiring sequential signatures or batch signing. This includes investigating efficient methods for handling multiple documents within a single signing session, exploring secure out-of-band communication methods, and enhancing browser-based signing capabilities for improved performance. Additionally, a comprehensive analysis of web browser security is crucial to ensure the overall resilience of our client-side cryptographic approach.

Acknowledgment

The author would like to thank the Rede Nacional de Ensino e Pesquisa (RNP), the Operador Nacional do Registro Civil de Pessoas Naturais (ON-RCPN), and Conselho Nacional de Desenvolvimento Científico (CNPq). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

[Aciobăniței et al. 2024] Aciobăniței, I., Arseni, S.-C., Bureacă, E., and Togan, M. (2024). A comprehensive and privacy-aware approach for remote qualified electronic signatures. *Electronics*, 13(4).

[Adobe Inc. 2024] Adobe Inc. (2024). Adobe acrobat. `https://acrobat.adobe.com/us/en/`. Accessed: 2024-08-19.

[Ascertia 2018] Ascertia (2018). Signinghub: Architecture and Deployment Guide. Accessed: 2024-06-08.

[Barker and Barker 2018] Barker, E. and Barker, W. (2018). Recommendation for key management. Part 2: Best Practices for Key Management Organization. Technical report, National Institute of Standards and Technology.

[Bit4id 2021] Bit4id (2021). Signcloud. Remote digital signature and key management. Accessed: 2024-06-08.

[Boeyen et al. 2008] Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., and Cooper, D. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

[Boneh and Franklin 2001] Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer.

[Brasil 2018] Brasil (2018). Lei Geral de Proteção de Dados Pessoais (General Data Protection Law. Lei n° 13.709, de 14 de agosto de 2018. *Diário Oficial da União*, 157(1):59–64.

[Brazil 1996] Brazil (1996). Lei de Propriedade Industrial (Industrial Property Law). Lei n° 9.279, de 14 de maio de 1996.

[Brazil 2011] Brazil (2011). Lei de Acesso à Informação (Freedom of Information Law). Lei n° 12,527, de 18 de novembro de 2011.

[Brazil, Economy Ministry 2021] Brazil, Economy Ministry (2021). Portaria SEDGG/ME n° 2.154, de 23 de fevereiro de 2021. Institui normas de gestão de integridade, riscos e controles internos no âmbito da Administração Pública Federal direta, autárquica e fundacional.

[CFM 2010] CFM (2010). Código de Ética Médica. Resolução CFM n° 1.931/2009.

[Choi et al. 2017] Choi, S.-H., Yun, J., and Park, K.-W. (2017). Doc-trace: Tracing secret documents in cloud computing via steganographic marking. *IEICE TRANSACTIONS on Information and Systems*, 100(10):2373–2376.

[Cryptomathic 2023] Cryptomathic (2023). Signer. Freedom to digitally sign documents remotely. Accessed: 2024-06-11.

[Digital Bazaar 2010] Digital Bazaar, I. (2010). Node-forge: A native implementation of TLS in JavaScript and Tools to Write Crypto-Based and Network-Heavy web apps. `https://github.com/digitalbazaar/forge`. JavaScript library for cryptographic and network tools.

[DigitalSign 2023] DigitalSign (2023). Signingdesk solution. Accessed: 2024-06-08.

[Eich 1995] Eich, B. (1995). Javascript. `https://developer.mozilla.org/en-US/docs/Web/JavaScript`. Programming language for web development.

[ETSI 2024] ETSI (2024). Electronic Signatures and Infrastructures (ESI): PAdES digital signatures; part 1: Building blocks and PAdES baseline signatures. Accessed: 2024-08-16.

[European Union 2018] European Union (2018). General data protection regulation, regulation (eu) 2016/679.

[Foundation 2024] Foundation, E. (2024). Ethereum. `https://ethereum.org/en/`. Accessed: 2024-08-16.

[GlobalSign and Ventures 2014] GlobalSign and Ventures, P. (2014). Pkijs: A public key infrastructure library for javascript. `https://pkijs.org/`. JavaScript library for working with X.509 certificates and cryptographic standards.

[Goldreich 2001] Goldreich, O. (2001). *Foundations of cryptography: volume 2, basic applications*, volume 2. Cambridge university press.

[Hansen and Eastlake 3rd 2011] Hansen, T. and Eastlake 3rd, D. E. (2011). US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234.

[ISO 2020] ISO (2020). ISO 32000-2: Portable document format (PDF) — part 2. International Standardization Organization.

[Jacomme and Kremer 2021] Jacomme, C. and Kremer, S. (2021). An extensive formal analysis of multi-factor authentication protocols. *ACM Transactions on Privacy and Security (TOPS)*, 24(2):1–34.

[Jonsson and Kaliski 2003] Jonsson, J. and Kaliski, B. (2003). Public-key cryptography standards (PKCS) #1: RSA cryptography specifications version 2.1. RFC 3447, Internet Engineering Task Force (IETF).

[Kohnfelder 1978] Kohnfelder, L. M. (1978). *Towards a practical public-key cryptosystem.* PhD thesis, Massachusetts Institute of Technology.

[Luan et al. 2015] Luan, H., Wang, C., Zhou, Z., and Yang, Z. (2015). Cross-access method for team confidential document based on offline key management. *International Journal of Security and Its Applications*, 9(1):97–108.

[Mayr et al. 2023] Mayr, L., Palma, L., Zambonin, G., Silvano, W., and Custódio, R. (2023). Monitoring key pair usage through distributed ledgers and one-time signatures. *Information*, 14(10):523.

[Mayr et al. 2024] Mayr, L., Zambonin, G., Schardong, F., and Custódio, R. (2024). One-time certificates for reliable and secure document signing. *arXiv preprint*.

[Moriarty et al. 2016] Moriarty, K., Kaliski, B., Jonsson, J., and Rusch, A. (2016). PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017.

[Moriarty et al. 2014] Moriarty, K., Nystrom, M., Parkinson, S., Rusch, A., and Scott, M. (2014). PKCS#12: Personal information exchange syntax v1.1. PKCS Standard 12, RSA Laboratories.

[Myers et al. 1999] Myers, M., Adams, C., Solo, D., and Kemp, D. (1999). Internet x.509 certificate request message format. RFC 2511, Internet Engineering Task Force (IETF).

[NextSense 2023] NextSense (2023). Signing suite. Accessed: 2024-06-08.

[Nystrom and Kaliski 2000] Nystrom, M. and Kaliski, B. (2000). PKCS#10: Certification request syntax specification version 1.7. PKCS Standard 10, RSA Laboratories.

[OAB 2015] OAB (2015). Código de Ética e disciplina da OAB, provimento no. 117/2000.

[Perottoni et al. 2023] Perottoni, E. D., Costa, B. P., Müller, F. L., dos Santos Camargo, V., Schardong, F., Silvano, W., Mayr, L., Custódio, R. F., Rocha, L., Lyra, C., et al. (2023). Menos certificação digital e mais identidade eletrônica: Icpedu e cafe em um assinador digital inclusivo. In *Anais Estendidos do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 93–96. SBC.

[Poppler Utils 2024] Poppler Utils (2024). pdfsig: Verify digital signatures in PDF documents. `https://manpages.ubuntu.com/manpages/jammy/man1/pdfsig.1.html`. Accessed: 2024-08-19.

[Prabakaran and Ramachandran 2022] Prabakaran, D. and Ramachandran, S. (2022). Multi-factor authentication for secured financial transactions in cloud environment. *CMC-Computers, Materials & Continua*, 70(1):1781–1798.

[Shannon 1949] Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715.

[Shatnawi et al. 2017] Shatnawi, A., Munson, E. V., and Thao, C. (2017). Maintaining integrity and non-repudiation in secure offline documents. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, pages 59–62.

[(SJCL) 2010] (SJCL), S. J. C. L. (2010). Sjcl: Stanford javascript crypto library. `https://bitwiseshiftleft.github.io/sjcl/`. A JavaScript library for cryptography developed at Stanford University.

[UFSC 2019] UFSC (2019). Portaria normativa nº 276/2019/gr, de 18 de setembro de 2019. `https://arquivos.ufsc.br/f/e28396694cc642a88d2e/?dl=1`. Institui e disciplina o uso de Certificação Digital na Universidade Federal de Santa Catarina.

[United Kingdom 1989] United Kingdom (1989). Official Secrets Act 1989.

[United Kingdom 2000] United Kingdom (2000). Freedom of Information Act 2000.

[United States 1917] United States (1917). Espionage Act of 1917.

[União Europeia 2014] União Europeia (2014). Regulamento (UE) n° 910/2014 do Parlamento Europeu e do Conselho. `https://eur-lex.europa.eu/eli/reg/2014/910/oj`.

[Ventures 2013] Ventures, P. (2013). Asn1js: A pure javascript library for parsing and serializing asn.1 data. `https://github.com/PeculiarVentures/ASN1.js/`. JavaScript library for working with Abstract Syntax Notation One (ASN.1) data.

[Vercel Inc. 2016] Vercel Inc. (2016). Next.js: The react framework for production. `https://nextjs.org/`. A React framework for building web applications.