Signatures to Go: A Framework for Qualified PDF Signing on Mobile Devices

Emina Ahmetovic[1], Thomas Lenz[1] and Christian Kollmann[2]
[1]*E-Government Innovation Center, Graz, Austria*
[2]*A-SIT Plus GmbH, Vienna, Austria*

Keywords: PDF Signature App, Qualified Electronic Signatures, Mobile Devices, Authentication, Qualified PDF Signing.

Abstract: Electronic documents are an important part of a business workflow. To assure the integrity, authenticity, and non-repudiation of those documents, both public and private sectors use qualified electronic signatures to sign PDF files. Benefits of the resulting qualified PDF signing are widely recognized, and there are many desktop and web applications used to sign PDFs. Those applications usually require additional hardware, such as smartphones, or smart cards, to assure a multi-factor authentication in the signing process. However, the prevalence of mobile devices in everyday life posed a need for public services, which can be executed on a single mobile device. In this paper, we develop a user-friendly and privacy-preserving framework for qualified PDF signing on mobile devices. We show the feasibility of our framework by implementing all necessary components: the PDF processing application, the Trust Service Provider server-side, and client-side application. The main focus of these components is to preserve the privacy of users and to meet user expectations regarding the functionalities of PDF signing applications. Furthermore, we demonstrate the practical applicability of our solution by integrating it into the productive Austrian e-Government system. Lastly, we conclude the paper with extensive performance evaluation.

1 INTRODUCTION

Signing PDF documents is a widely used service within the e-Government and business sector. To deliver trust in the way of conducting business, a signing process needs to involve security mechanisms that can protect those documents. For this purpose, qualified electronic signatures (QES) (eIDAS Regulation, 2014) are used as a secure and reliable means for signing electronic documents. QES are defined as advanced electronic signatures based on qualified certificates and created using a qualified signature creation device (QSCD) (eIDAS Regulation, 2014). They also represent a legal equivalent to handwritten signatures. When QES are incorporated into PDF file in a way that satisfies the PDF Advanced Electronic Signature (PAdES) specifications (ETSI, 2009), qualified PDF signatures are created.

The qualified PDF signing provides benefits for business and e-Government - saves time, money, delivers flexibility, makes services agile, and much more (Systems, 2009). The Adobe Document Cloud reports eight billion electronic and digital signature transactions in 2017[1] used mostly for signing contracts, invoices as well as in legal proceedings (Mladenov et al., 2019). Given this widespread use of PDF signatures, there is also a vast number of PDF signing applications. Those are web and desktop applications tailored for devices such as personal computers and laptops. While PDF signing applications have reached a satisfying level of maturity and acceptance among users (Theuermann et al., 2019b), they still expect that users own a smart card or a token for authenticating themselves and accessing the service.

As we entered the smartphone era, users are less likely to meet those expectations (Consortium, 2018). The widespread adoption of mobile handsets resulted in a transition from e-Government to mobile e-Government (m-Government) and allowed citizens to access public services via their smartphones (Mengistu et al., 2009),(Theuermann et al., 2019b). These developments also raised a need for *mobile-only* PDF signing. The previous research (Zefferer and Teufl, 2011) finds smartphones architecturally suitable for the task; however, there is currently no

[1]Adobe Fast Facts. https://www.adobe.com/about-adobe/fast-facts.html

Ahmetovic, E., Lenz, T. and Kollmann, C.
Signatures to Go: A Framework for Qualified PDF Signing on Mobile Devices.
DOI: 10.5220/0009821303300339
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 330-339
ISBN: 978-989-758-446-6

mobile app for qualified PDF signing, which at the same time satisfies aspects of security, privacy, or usability. A worrisome finding that revealed security and privacy risk cases when online web applications are exploited as mobile services urged the need to find a solution for single-device users.

Challenges. There are a couple of reasons why qualified PDF signatures remained insufficiently explored inside the m-Government context. In general, users anticipate that mobile services will offer the same functionalities at the same security level as web and desktop devices (Alliance, 2017). However, providing an elegant way to securely authenticate users emerged as one of the biggest challenges. Mobile devices have different security features and hardware capabilities compared to personal computers. Moreover, constant changes in mobile technology made it hard to come up with a long-term and stable authentication mechanism. Different approaches with smart cards, tokens, or additional smartphones, have been proposed and implemented in past years. While those solutions provide secure authentication, they suffer from usability issues when used in the mobile domain (Theuermann et al., 2019a), (Lenz and Alber, 2017). Fortunately, recent advances in mobile technology, such as secure storage and support for biometrics, can be leveraged to tackle the above-mentioned challenges.

Contribution. In this paper, we propose a user-friendly and privacy-preserving solution for qualified PDF signing on mobile devices.

First, we introduce a framework for generating qualified PDF signatures on a single mobile device. a) Users can create legally binding PDF files in such a way that authenticity, integrity, and non-repudiation are guaranteed. b) The solution addresses the privacy-related issues from prior work. The extensive client-side implementation improves the privacy of users by reducing dependencies on third-party services. c) We propose a mobile app that processes PDF files. The app provides the same usability features as desktop and web applications for PDF signing. d) Our solution employs easy means of authentication without additional hardware requirements. Device-level biometrics supports strong authentication on a single device.

Second, we implement all the necessary components to show the architectural feasibility of our solution. The heart of our implementation is the PDF Signature App, an app that processes PDF files. For demonstration purposes, we further implemented both mobile application Trust Server Provider App and web service Trust Service Provider. We show the practical applicability of our solution by integrating it into the Austrian identity management infrastructure. The components such as the Trust Service Provider and the Trust Service Provider App are instantiated with the qualified trust service operator in Austria.

Third, we evaluate the capabilities of our solution by benchmarking PDF processing app with different signature features and also performing functionality verification.

Outline. This paper is structured as follows: In Section 2, we introduce the existing PDF signature solutions and the research gap. In Section 3, we explain the architectural design of our framework. We define requirements, components, and interfaces between them. In Section 4, we describe the implementation of components, which we evaluate in Section 5. Lastly, we conclude the paper in Section 6.

2 BACKGROUND

As the benefits of paperless signing have been widely recognized, many countries inside the EU and beyond have invested in e-signature technologies that can be used by a large number of people. One such an instance is the Austrian *Citizen Card Concept* (Posch et al., 2011),(Leitold et al., 2009). This concept defines a generic protocol for creating QES based on smart-cards and server-side signature solutions. Smart cards serve as a qualified signature creation device; nevertheless, they require smart-card readers inserted into computers or laptops alongside the middleware software applications. Alternatively, a server-side signature solution represents a user-friendly approach in terms of hardware requirements. This solution is based on the server-side authentication at the Hardware Security Module (HSM), where the user engages two-factor authentication to prove the alleged identity. Two-factor authentication usually requires a combination of *knowledge* factor (password) and a *possession* factor, demonstrated by receiving One Time Password (OTP) in the form of a TAN via SMS using a mobile device. The important security feature is the fact that these two factors are verified via two separate communication channels.

Numerous desktop and web applications for PDF signing are based on both smart-card and server-based signature solutions. One popular implementation is the PDF-AS, a framework used for digital signing and verifying PDF files. The framework creates PAdES (ETSI, 2009) signatures and consists of the three main parts: Library - a core component, Web - web interface for signing PDFs, and Client - a command-line interface (Fitzek et al., 2015), (EGIZ, 2014a). Another recommended solution is the PDF-Over, a Java desktop application for signing PDF files. It provides a graphical interface that allows users to navigate through the file and choose the location of the signature. This application is based on the PDF-AS Library, and it is available for Windows, Linux, and macOS (EGIZ, 2014b). In contrast to applications where PCs and laptops are end-user devices, providing a mobile-only service for qualified PDF signing faced challenges in terms of usable authentication. Existing smart-cards solutions in countries such as Austria or Estonia either have usability and interoperability issues or cannot be used with mobile devices at all (Lenz and Alber, 2017), (Consortium, 2018). The current implementation of server-based solutions in the Austrian solution requires possession of two separate devices (e.g., two mobile devices) to be used in the mobile domain.

A concern to find a solution evolved with the appearance of privacy and security-critical use cases. Smartphones are used for a variety of personal and business purposes, where activities that require higher security protection are commonplace. The absence of appropriate technology leads to use-cases where the QES is obtained by using a single mobile device, more concretely by accessing the existing web application for PDF signing from the mobile browser. In this case, both authentication factors (e.g., password and TAN) are entered from the same mobile device and delivered to the Trust Service Provider via one communication channel. The reduction of communication channels significantly compromises the security of the process. A simple malware installed on a mobile device could gain access to user credentials, and it could also intercept the TAN (Fitzek, 2016). As mobile devices should only be used as a second authentication factor in the server-based solutions, this way of signing PDF files is not recommended by the Trust Service Provider. In addition, outsourcing PDF processing also affects the privacy of users, as the sensitive data is disclosed to third party providers.

In this paper, we aim to provide an alternative to these security-critical use cases. We leverage the server-based authentication concept (Theuermann et al., 2019a) for generating QES on mobile devices, which conforms with electronic IDentification, Authentication and trust Services (eIDAS) (eIDAS Regulation, 2014) regulations, and we make a contribution by defining a new use case that can be used to create qualified PDF signatures. The secure hardware element is used to store cryptographic material, which is further protected with local authentication methods using biometrics. The combination of the factor knowledge (password) together with the factor possession (cryptographic material) enables multi-factor authentication on mobile devices.

3 FRAMEWORK FOR SIGNING PDF FILES

In this section, we will provide a detailed description of our proposed framework for qualified PDF signing on mobile devices. First, we start by outlining the requirements for a solution that enables qualified PDF signing on mobile devices. Furthermore, we introduce the framework's architecture involving main components, and we explain the communication flow between them.

3.1 Requirements

Our goal is to develop a user-friendly and privacy-preserving solution for generating qualified PDF signatures on mobile devices. To achieve this, we derived requirements for future architectural and implementation decisions.

- **Privacy.** In environments like e-Government, where sensitive user data are handled, and misuse of private data can violate citizens and business rights(Posch et al., 2011), privacy is among the main concerns. In our framework, we reduce trust requirements in order to preserve the privacy of user data. Thus, instead of relying on the existing web services which could be used for processing of PDFs, we implement the PDF processing locally inside the mobile app.
- **Usability.** Usability, as a merit for user adoption, is another requirement of our framework. In general, e-Government heavily relies on citizens' experience with public services. The citizen's satisfaction results in a higher level of acceptance and usage of public services hence increase the flexibility of e-Government (Zefferer and Krnjic, 2012). Moreover, users are already familiar with the concept of signing PDF files on desktop applications, which draws some expectation. While certain user-friendly parameters, such as minimal interactions or execution time, are predefined by the authentication concept, others can be adopted by the framework. To address the usability as a requirement, we develop an intuitive application, where a user can dynamically place a signature on the file, and also customize signing features.
- **Security.** Security features have a strong impact on the overall citizen's communication and engagement with e-Government. Transaction-based e-services have to be perceived as secured from a

citizen's perspective (Tassabehji et al., 2007). As one of the requirements of our framework, security often comes in a trade-off with usability. Reliable authentication methods require a high level of security, but on the other hand, these methods should not be too complex for users to handle (Theuermann et al., 2019b). In our solution, we try to balance between these two requirements by integrating a user-friendly and secure multi-factor authentication mechanism. Furthermore, compliance with PAdES standard makes it harder to perform successful attacks (Mladenov et al., 2019) based on signature manipulations by adding unsigned content after the signature, as PAdES states that whole content of the file (except signature itself) needs to be signed.

In addition to the associated requirements, we include interoperability as an additional one, which allows implementation on different mobile platforms, and does not depend on vendors of mobile devices.

3.2 Framework's Architecture

To define the components of the proposed framework, we split them into two environments: client and server-side. Figure 1 illustrates the components, as well as the interactions between them.

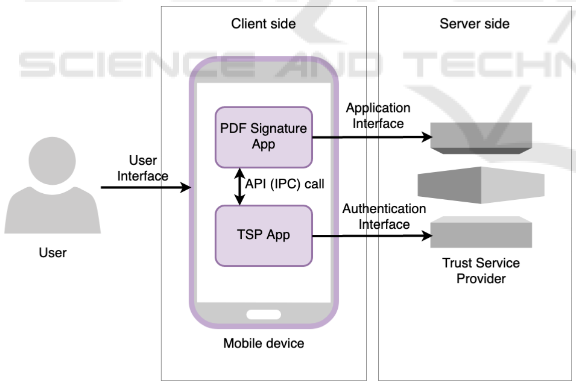

Figure 1: The architecture of the framework. High-level overview.

The client environment consists of three components.

- **User.** A natural person who initiates the signing process with the purpose of signing a PDF file.
- **PDF Signature App.** The PDF Signature App is the heart of our solution and has the role of a Service Provider that provides qualified PDF signing. The main features implemented in the app are the library for PDF processing and an API for user authentication.
- **Trust Service Provider App.** The Trust Service Provider App is the mobile application provided by the Trust Service Provider. Its main role is to provide an interface for the user authentication at the HSM at the server-side.

The server environment consists of only one component, which is a Trust Service Provider (TSP).

- **Trust Service Provider.** An entity that meets the eIDAS requirements to provide a qualified trust service.

3.3 Communication Flow

In this section, we will describe the communication flow between the components. First, we start by outlining four interfaces, and then we describe the interaction flow between components.

User Interface. This interface describes the communication between a user and the mobile device. A user establishes an interaction with PDF Signature App by requiring service and interacts with TSP App for authentication purposes.

Application Interface. The application interface defines the communication between PDF Signature App and Trust Service Provider. The PDF Signature App initiates the communication by sending a request to the TSP, outlining the purpose of the request.

API (IPC) Call. The communication between the TSP App and the PDF Signature App is established through Inter-Process communication (IPC). Optionally, this communication can be established via API call if the two apps, or their libraries, are merged inside one framework.

Authentication Interface. This interface encapsulates the authentication of users against the qualified TSP via the TSP App. One of the reasons why the authentication is done by the TSP App, and not by PDF Signature App, is due to the fact that the actual transmission of credentials is secured by the custom protocol between TSP and related components. This empowers the TSP to modify the authentication procedure conveniently.

The following enumeration describes the communication flow in more detail.

1. A user wants to sign a PDF file on a mobile device. For this purpose, a user shares a file with the PDF Signature App.
2. The PDF Signature App receives the file and starts PDF processing. This means that signing parameters are gathered from a user, and the signature

is prepared to be incorporated into the file. To finalize a signature, a user needs to authenticate herself against the HSM on the server-side successfully. Thus, the app initiates a user authentication by sending a request to the Trust Service Provider. The request contains parameters that indicate what service is required from the TSP.

3. Trust Service Provider analyzes the request and sends the response back to the PDF Signature App. The response contains parameters for IPC (or API) communication towards the TSP App.
4. PDF Signature App uses the parameters to establish communication with the TSP App. The authentication process is now delegated to the TSP App.
5. The TSP App provides an interface to the user, where she can authenticate herself.
6. User's credentials have been transmitted from the TSP App to the TSP according to their defined protocol.
7. After the authentication is completed successfully, the Trust Service Provider sends the response with the PDF signature to the TSP App.
8. The response is forwarded to the PDF Signature App from the TSP App. The PDF Signature App uses the PDF signature from the request to complete the signing process. The communication flow is illustrated in Figure 2.

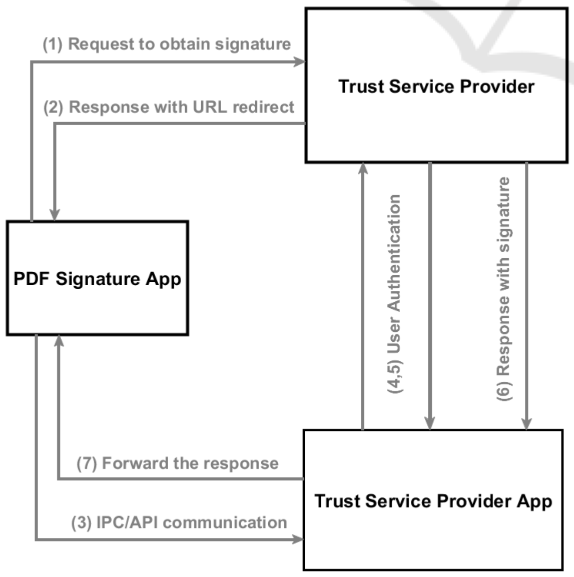
(1) Request to obtain signature
Trust Service Provider
(2) Response with URL redirect
PDF Signature App
(4,5) User Authentication
(6) Response with signature
(7) Forward the response
Trust Service Provider App
(3) IPC/API communication

Figure 2: High-Level communication overview.

4 IMPLEMENTATION

This section describes the implementation of the previously proposed framework. For simplicity, we will split the implementation into three parts; First, we provide a detailed description of the PDF Signature App. Second, we briefly describe an authentication process. To provide a proof-of-concept, we implement two additional applications: a demo Trust Service Provider (server-side) and a demo Trust Service Provider App (client-side). Third, we demonstrate our solution in practice by integrating it with the existing Austrian production system.

4.1 PDF Processing

The local implementation of PDF processing has a significant impact on addressing the privacy of the solution. To build a privacy-preserving framework, we reduce the trust requirement by moving the entire PDF processing implementation to the client-side. The existing web services for PDF processing could provide faster processing time in contrast to mobile devices with limited resources. However, we argue that local implementation is a reasonable trade-off since eliminating the need for third-party services, we have increased privacy. To additionally support this decision, the evaluation study, described in Section 5, shows that the local signature implementation does not play a significant role in the overall user experience.

Digital Signatures in PDF. One of the special features of the PDF, starting from version PDF 1.3 (Systems, 2007), is the ability to add digital signatures as a means to guarantee integrity, authenticity, and non-repudiation of electronic documents. The implementation of the qualified PDF signature into the PDF file consists of multiple iterations. The first step is the interaction with the user to select the desired PDF file to be signed. After file selection, the app creates the graphical preview of the file, allowing the user to navigate through the file opting for the desired signature position. While navigating, a user can dynamically move a signature block, which presents a placeholder for the actual signature position. Detailed descriptions of the signature block and other fundamental parameters of the signature processing will be provided in the following paragraphs.

Signature Block. The PDF Signature App provides an interface for a user to preview the file, navigate through it, and place the signature block on the desired location. The signature block represents a visualization of a signature on a file. It is implemented as a table with a configurable appearance. In the case of the invisible signature, the signature block is not visible on the file. In the case of the visible signature, a signature profile specifies the appearance of a signature.

Signature Profile. A signature profile defines all relevant information about the signature block's appearance and content. It defines a style of the table (size, font, padding, color), but more importantly, it defines identifiers of the signature block and their values. The identifiers represent the input values of the table, which provide further information about the signature. Some of them are predefined and related to one specific signature profile (e.g., the reason for signing, date), while a user can insert some other values (e.g., signatory name). The idea behind signature profiles is the ability for users to customize their signature profiles according to needs.

Signature Dictionary. Digital signatures are incorporated in the PDF file itself by using a PDF structure called the *signature dictionary*. A signature dictionary contains the signature value as well as all relevant information that defines the nature of the signature (Systems, 2006). The signature value is encoded as a binary object using Cryptographic Message Syntax (CMS) or related signature format (*PKCS*#7 (Society, 1998), CMS Advanced Electronic Signature - CAdES(ETSI, 2012)) object stored in the *Contents* entry of signature dictionary. Alongside the signature, the dictionary also contains *ByteRange*, a pair of numbers which specify for which bytes the signature digest is calculated. The field *Filter* is associated with the internal name of the signature handler, while the *Subfilter* contains valuable data for the signature validator handlers and represents the type of signature (e.g., ETSI.CAdES.detached). Other entries can be *Reason*, a string that explains why the signatory is signing the file, as well as *Location*, *Name*, *Date*. An illustration of the signature dictionary extracted from the signed file is shown in the listing below.

```
20 0 obj
<<
/Type /Sig
/Filter /Adobe.PPKLite
/SubFilter /ETSI.CAdES.detached
/M (D:20190403142400+02'00')
/Reason (Signature validation under
http://www.signaturpruefung.gv.at)
/Contents <308006092A...00000000>
/ByteRange [0 14587 22781 17992]
>>
endobj
```

Incremental Update. The important feature of the PDF is the incremental update capability (Systems, 2006). Every change applied to a signed PDF file, which alters the content of the PDF, would normally invalidate the existing signature. However, an incremental update allows the modification of the file by adding the information about file modification in a special incremental update section located at the end of the file. Thus, it enables multiple signatures without modifying previously signed data.

Additional Features. In addition to the basic signature properties, we have implemented additional features to enhance user experience with the app. A user can choose between different languages of the signature block, whether the signature will be invisible or visible. Furthermore, PDF/A (Consortium, 2019) and PDF/UA (ISO, 2014) compliance are also implemented.

4.2 Authentication Process

After processing a PDF file and creating a signature dictionary, the actual PDF signature (signature value and corresponding features) is issued by the TSP after a user successfully authenticated herself. The process of user authentication is based on a custom authentication protocol introduced in (Theuermann et al., 2019a). This authentication process requires collaboration between PDF Signature App, TSP App, and TSP. The PDF Signature App initiates user authentication; however, the actual authentication of a user is done between TSP and TSP App. The following listing describes the authentication flow in details.

1. PDF Signature App sends an *https* request to the TSP. The request contains a JSON object with defined parameters. The example of the request body is illustrated in the listing below. The parameters indicate that the purpose of a requested service is to create *CAdES* signature.

```
{
  "v":10,
  "reqID":"e7e11...b8a9c",
  "payload":{
     "name":"createCAdES",
     "params":{
        "keyId":"SecureSignatureKeypair",
        "contentMode":"detached",
       ...
     }
  }
}
```

2. The TSP analyzes the received request and returns a response that contains the *url* parameter. This *url* parameter serves to establish IPC connection towards TSP App and to delegate further user authentication towards the TSP App.

3. The PDF Signature App establishes an IPC communication with TSP App. In some scenarios, it is possible that TSP App and PDF Signature App - or its library are used as one app. In this case, we would not have an IPC call, but rather an API call.
4. The TSP App provides an authentication interface to the user.
5. The user authenticates herself by providing factor knowledge in the form of a password. The factor possession, which is cryptographic material on a secure element, is demonstrated based on the challenge-response procedure between the TSP App and TSP. The access to the cryptographic material is done by the local authentication, where a user applies fingerprint.
6. The TSP validates the received data from TSP App and sends back a response. In case a user confirmed the alleged identity, the response contains a signature.
7. The TSP App forwards the response from TSP to PDF Signature App. The PDF Signature App uses the parameters from the response to finalize the signature and complete the signing process. The example of the JSON object contained in this response is provided in the following listing.

```
{
  "v":10,
  "respID":"1 ccc8d84 ...41ccd",
  "inResponseTo":"e7e11...b8",
  "transactionID":"c01480ee -..d4285",
  "payload":{
     "name":"createCAdES",
     "result":{
        "signature":"MIAGCS...AAA"
     }
  }
}
```

4.3 PDF Signature App Integration with Official Trust Server Provider

The implementation of the mobile PDF Signature App, the TSP, and the TSP App has shown the feasibility of the proposed framework. However, we aim to demonstrate and evaluate the functionality of our solution in production. For this purpose, we have replaced prototypes of the TSP and the TSP App with the official Austrian Trust Service Provider on the production level, operated by the A-Trust[2]. A-Trust is a qualified TSP in Austria that provides the client-side application and defines the protocol of the authentication. This gives the possibility to conveniently adjust the authentication mechanism based on developments in mobile technology. We have integrated the PDF Signature App with the official TSP and TSP App. The integration shows that our solution can operate in practice.

[2]A-Trust Gesellschaft fuer Sicherheitssysteme im elektronischen Datenverkehr GmbH https://www.a-trust.at/

5 PERFORMANCE

After we have integrated our components with components that operate productively in Austrian e-Government, we evaluate the performance of our solution in a real-world scenario. First, we benchmark the implementation of PDF processing by performing tests with different file sizes on different devices and different signature features. As the entire signing process includes human interaction and results that measure the overall time of signing could be biased by users' experience, we benchmark only the part of the processing of PDF files. Second, we perform the functionality verification of our solution compared to existing web and desktop applications. Third, we briefly discuss the results of the performed tests. The purpose of the evaluation is to determine to what extent the local implementation of PDF processing on mobile devices affects the signing process, and what parameters have the most influence on the time of processing. It is known that mobile devices have less computational power compared to personal computers; thus, the client-side implementation could have drawbacks on the user experience.

5.1 PDF Processing Performance

We benchmark the processing of PDF files on mobile devices by performing tests with five different PDF file sizes, starting from 37 kB to 3980 kB. The tests are executed on a different range of smartphones, starting from low-end to high-end devices, with different OS versions. To give a sense of how fast the library process PDF files, we take into account three different implementations of signatures: invisible, visible, and PDF-A compliant signatures.

Implementations. The tests are performed for invisible, visible, and PDF-A compliant PDF signatures. Invisible PDF signatures are not visible on the document, although they exist in a byte structure of the file and comply with PAdES; thus, it is not necessary to implement visual representation such as signature blocks. Furthermore, we have the common case of visible signatures, where the signature is represented in the form of a signature block. Lastly, we introduce

PDF-A signatures that need to satisfy further requirements thus, require additional implementation. As the complexity of implementation increases with each of these signatures respectively, we measure the overall time required to process PDF file for signature integration.

Results. The achieved results are depicted in table 1. Results represent measured time in seconds for three different smartphones, five different files, and three implementations of signatures. Every test is performed 100 times, and results in the table are average values. We can conclude that different parameters can have a significant impact on the required time for processing files.

In general, the best results are obtained on the high-end device, for invisible signatures, and small file sizes. High-end devices execute the signing process fastest for all three signature implementations and all file sizes. This can be corroborated with the fact that their CPU is more powerful. The high-end device also performs, on average, approximately 1.6 times faster than the mid-end device, which is rather an expected change. However, the high-end device performs, on average, six times faster than the low-end device (figure 3).

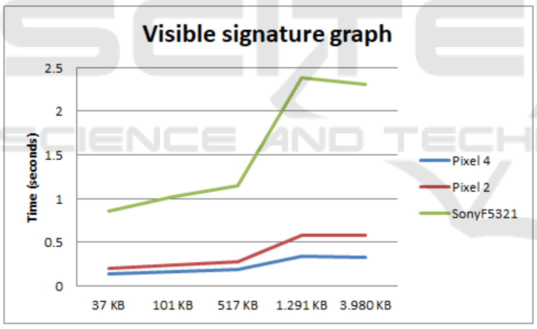

Figure 3: Graph depicts changes in measured time for visible signatures on three devices for different sizes.

Furthermore, an increased complexity of signature implementations causes an upward trend in the execution time. Invisible signatures, as the most lightweight in terms of implementation, are processed fastest on all devices, whereas the difference between visible signatures and PDF-A compliant signatures is negligible. This is due to the fact that invisible signatures take the least computations as a large part of implementation is omitted.

Lastly, we observe that the overall signing time gradually increases as the files are becoming larger. For instance, when the file size increases ten times, the overall time increases approximately three times.

Table 1: Execution times (in seconds) for PDF processing on different mobile devices (Sony F5321, Pixel 2, Pixel 4) and different file sizes from the range of 37kB to 3,98MB. Execution is calculated for three types of signatures: visible, invisible and PDF-A compliant.

#	Size	Pixel 4 Android 10	Pixel 2 Android 9	Sony F5321 Android 8
		Invisible signatures		
100x	37 kB	0.08439	0.12820	0.27785
100x	101 kB	0.08943	0.14422	0.40708
100x	517 kB	0.10460	0.18426	0.51333
100x	1291 kB	0.27362	0.48312	1.77479
100x	3980 kB	0.25540	0.47851	1.68918
		Visible signatures		
100x	37 kB	0.14104	0.20756	0.85911
100x	101 kB	0.15983	0.23777	1.02499
100x	517 kB	0.18563	0.27941	1.15277
100x	1291 kB	0.34177	0.57721	2.37649
100x	3980 kB	0.32690	0.58272	2.30262
		PDF-A compliant signatures		
100x	37 kB	0.14697	0.21355	0.87193
100x	101 kB	0.16124	0.24005	1.02774
100x	517 kB	0.17672	0.28330	1.15359
100x	1291 kB	0.34672	0.58057	2.37060
100x	3980 kB	0.33314	0.58369	2.17546

5.2 Functionality Verification

During the functionality verification, we have measured the overall time required for a user to sign a PDF file by different PDF signing applications. To get an impression for comparison, we have used three applications (web and desktop applications are already introduced in Section 2): the PDF-Over as a desktop application, the PDF-AS as a web application, and the PDF Signature App as a mobile application. All three applications use the same qualified TSP[3], whereas

- The PDF-Over application has been deployed on a MacBook Pro laptop (Intel Core i5 2GHz CPU, 8GB RAM, macOS Sierra version 10.12.6).
- The PDF-AS is used as a web application from a PC (Intel Core i74770 CPU 3.4GHz, 32GB RAM, Windows 10 OS).
- The PDF Signature App is tested on a Google Pixel 2 smartphone (Snapdragon 835 CPU, 4GB RAM, Android 9 OS).

The overall signing time has been divided into four phases:

1. OPEN - time from the first user interaction with the application until the moment a PDF file has been selected.

[3] A-Trust Gesellschaft fuer Sicherheitssysteme im elektronischen Datenverkehr GmbH https://www.a-trust.at/

2. POSITION - the time from the moment a user has a preview of the file until the moment a user places a signature block on a file.
3. SIGN - time to initiate the signing process, which also includes user authentication.
4. FINISH - time required for a user to handle the last interface.

The test is performed two times for files of different sizes to capture the changes in signing time considering file sizes. The first file has 190kB, and the second file has 2.26MB. The overall signing time is measured in seconds and performed by a single user. Tables 2 depicts the results of the test.

Table 2: The overall signing time measured in seconds for two different files executed on three: desktop, web, and mobile. *Interface not provided.

Phase	Desktop PDF-OVER	Web PDF-AS	Mobile PDF SIGNATURE APP
PDF file size 190kB			
Open	5.726	9.772	5.063
Position	5.179	0*	3.437
Sign	32.463	32.627	24.333
Finish	4.865	0*	1.951
Sum	48.233	42.399	34.784
PDF file size 2.26MB			
Open	7.893	9.174	5.874
Position	4.381	0*	4.375
Sign	38.615	39.798	27.675
Finish	5.334	0*	1.937
Sum	56.223	48.972	39.861

Results. The performed functionality verification attempts to answer the question if our framework performs the task as designed. As usability is one of the requirements, it is important to discuss whether the task can be performed in such a way that the framework meets user expectations regarding the flow of signing, time of signing, and signature features.

The results of the performed verification show that the PDF Signature App successfully performs the process of qualified PDF signing on one single device. Even though the evaluation is based on only two files, if we combine the results the ones in Table 1, it is possible to generally conclude that local PDF processing implemented in the app does not have a strong impact on the app's performance. Furthermore, we argue that PDF Signature App offers easier means of authentication in comparison to the other two applications. Both PDF-Over and PDF-AS use server-side signature solutions based on the OTP and credentials. As entering received TAN can be cumbersome, PDF Signature App, on the other hand, employs device-level biometrics together with credentials in the authentication process.

5.3 Summary

The proposed framework enables qualified PDF signing on a single mobile device. Moreover, it provides the same functionality as users got accustomed to, such as an interface for positioning signatures, and the setup for setting different signature features. Performance evaluation shows that different factors can have an influence on the overall signing process, or more precisely, PDF processing time. Mobile devices that host the PDF processing App have the biggest influence on the time required for processing PDF, meaning that both the speed and power of the device directly affect the time of signing. Furthermore, the complexity of signature implementation is an additional parameter that determines the time for the signing, as well as the size of files.

6 CONCLUSION

In this paper, we proposed a framework for obtaining qualified PDF signatures on mobile devices. Our framework addresses security- and privacy-critical use cases and tackles the general lack of mobile transaction-based technologies in the m-Government domain. To show that our framework is capable of providing qualified PDF signatures satisfying privacy, usability, and security requirements, we have implemented three components. PDF processing is implemented locally inside the PDF Signature App, whereas the authentication procedure is delegated and executed from two additional components: the TSP and the TSP App. In addition, we evaluated our solution by integrating it with the official Austrian Trust Service Provider. One of the most significant advantages of the framework is the fact that satisfying the security level of the process can be achieved without additional hardware and enhancing the privacy of users.

REFERENCES

Alliance, S. T. (2017). Mobile Identity Authentication Version 1.0. Technical report.

Consortium, C. S. (2018). Architectures and protocols for remote signature applications Published version 1.0.3.0 (2018-12). https://cloudsignatureconsortium.org/wp-content/uploads/2019/02/CSC_API_V1_1.0.3.0.pdf. Accessed: 2019-03-19.

Consortium, D. P. (2019). PDF Standards Overview. http://3dpdfconsortium.org/pdf-standards/. Accessed: 2019-04-08.

EGIZ (2014a). PDF-AS. https://joinup.ec.europa.eu/solution/pdf. Accessed: 2019-03-19.

EGIZ (2014b). PDF-Over. https://joinup.ec.europa.eu/solution/pdf-over. Accessed: 2019-03-19.

eIDAS Regulation (2014). Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. https://www.eid.as/home. Accessed: 2019-03-19.

ETSI (2009). Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES. https://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf. Accessed: 2019-04-08.

ETSI (2012). Electronic Signatures and Infrastructures (ESI); CAdES Baseline Profile. https://www.etsi.org/deliver/etsi_ts/103100_103199/103173/02.01.01_60/ts_103173v020101p.pdf. Accessed: 2019-04-08.

Fitzek, A. (2016). Multi-factor Authentication. Technical report, eGovernment Innovation Center.

Fitzek, A. G., Maierhofer, C., Tauber, A., and Zwattendorfer, B. (2015). Fortgeschrittene pdf signaturen mit pades. *eGovernment review*, (15):16–17.

ISO (2014). Use of ISO 32000-1 (PDF/UA-1). https://www.iso.org/obp/ui/#iso:std:iso:14289:-1:ed-2:v1:en. Accessed: 2019-10-27.

Leitold, H., Posch, R., and Rössler, T. (2009). Media-break resistant esignatures in egovernment: an austrian experience. In *IFIP International Information Security Conference*, pages 109–118. Springer.

Lenz, T. and Alber, L. (2017). Towards cross-domain eid by using agile mobile authentication. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 570–577. IEEE.

Mengistu, D., Zo, H., and Rho, J. J. (2009). M-government: Opportunities and challenges to deliver mobile government services in developing countries. *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pages 1445–1450.

Mladenov, V., Mainka, C., zu Selhausen, K. M., Grothe, M., and Schwenk, J. (2019). 1 trillion dollar refund: How to spoof pdf signatures. In *CCS '19*.

Posch, K. C., Posch, R., Tauber, A., Zefferer, T., and Zwattendorfer, B. (2011). Secure and privacy-preserving egovernment—best practice austria. In *Rainbow of computer science*, pages 259–269. Springer.

Society, T. I. (1998). PKCS #7: Cryptographic Message Syntax Version 1.5. https://tools.ietf.org/html/rfc2315. Accessed: 2019-10-27.

Systems, A. (2006). Digital Signatures in the PDF Language - Developer Technical Note . https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/DigitalSignaturesInPDF.pdf. Accessed: 2019-04-08.

Systems, A. (2007). Digital Signatures in Acrobat. https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/digisig_in_acrobat.pdf. Accessed: 2019-04-08.

Systems, A. (2009). The AdES family of standards: CAdES, XAdES, and PAdES. https://blogs.adobe.com/security/91014620_eusig_wp_ue.pdf. Accessed: 2019-03-19.

Tassabehji, R., Elliman, T., and Mellor, J. (2007). Generating citizen trust in e-government security: Challenging perceptions. *International Journal of Cases on Electronic Commerce (IJCEC)*, 3(3):1–17.

Theuermann, K., Tauber, A., and Lenz, T. (2019a). Mobile-only solution for server-based qualified electronic signatures. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE.

Theuermann, K., Zefferer, T., Lenz, T., and Tauber, A. (2019b). Flexible und benutzerfreundliche authentifizierungsverfahren zur umsetzung transaktionaler e-government-services auf mobilen geräten. *Handbuch E-Government: Technikinduzierte Verwaltungsentwicklung*, pages 1–30.

Zefferer, T. and Krnjic, V. (2012). Towards user-friendly e-government solutions: usability evaluation of austrian smart-card integration techniques. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 88–102. Springer.

Zefferer, T. and Teufl, P. (2011). Opportunities and forthcoming challenges of smartphone-based m-government services. *Megatrends in eGovernment*.