



Emina Ahmetovic

Enabling PDF Signing on Mobile Devices Using Qualified Electronic Signatures

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Institute for Applied Information Processing and Communications (IAIK)

Graz, August 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Abstract

Government agencies, as well as the private sector, use qualified PDF signatures to deliver flexibility, cost, and time savings in their services. For this purpose, the Portable Document Format (PDF) is used in combination with qualified electronic signatures (QES) to create legally binding e-documents and assure their integrity, authenticity, and protection against repudiation. The vast number of PDF processing applications demonstrate the general availability of PDF signatures and their acceptance among users. Nevertheless, the existing solutions are mainly tailored for stationary devices, such as personal computers, as they require additional hardware to provide multi-factor authentication. The widespread use of mobile handsets in everyday life posed a need for transaction-based solutions executable only on a single mobile device. However, this topic remained insufficiently explored inside the mobile e-Government sector.

In this thesis, we propose a mobile solution for PDF signing. We design a model that tackles gaps from previous work and provides a user-friendly and privacy-preserving service for creating qualified PDF signatures on a single mobile device. Our solution is based on a novel server-side authentication concept, where the users engage alternative authentication methods such as a fingerprint, to authenticate themselves against the remote Hardware Security Module and generate a QES. We show the feasibility of our model by implementing all the necessary components for our solution. Furthermore, we demonstrate the practical applicability of our solution by integrating it into the Austrian e-Government productive system.

Zusammenfassung

Regierungsbehörden und privatwirtschaftliche Unternehmen verwenden qualifizierte PDF-Signaturen, um ihre Flexibilität zu steigern und um Kosten und Zeit zu sparen. Zu diesem Zweck wird das Portable Document Format (PDF) in Kombination mit qualifizierten elektronischen Signaturen (QES) verwendet um rechtsverbindliche elektronische Dokumente zu erstellen, sowie deren Integrität und Authentizität zu gewährleisten.

Die große Anzahl von PDF-Anwendungen demonstrieren die allgemeine Verfügbarkeit von PDF-Signaturen und ihre Akzeptanz bei den Nutzern. Vorhandenen Lösungen sind jedoch hauptsächlich auf stationäre Geräte wie PCs zugeschnitten, nachdem diese Lösungen zusätzliche Hardware benötigen um Multifaktor-Authentifizierung zu ermöglichen. Durch die weite Verbreitung von Mobiltelefonen sind Regierungen auch daran interessiert PDF-Signaturen auf diesen Mobilgeräten anzubieten. Dieses Thema wurde jedoch im Bereich des mobilen E-Government nur unzureichend behandelt.

In dieser Arbeit präsentieren wir eine mobile Lösung für PDF-Signierung. Wir entwerfen ein benutzerfreundliches Modell zur Erstellung qualifizierter PDF-Signaturen auf Mobilgeräten mit Hilfe eines Signaturservers, welches Lücken aus früheren Arbeiten schließt und die Privatsphäre der Nutzer schützt. Unsere Lösung basiert auf einem neuartigen Konzept zur Authentifizierung, bei dem der Benutzer verschiedene Authentifizierungsmethoden wie einen Fingerabdruck anwenden kann, um sich gegenüber dem Hardware-Sicherheitsmoduls des Servers zu authentifizieren, wodurch die Erstellung der QES ermöglicht wird. Wir zeigen die Machbarkeit unseres Modells, indem wir die einzelnen Komponenten unserer Lösung implementieren. Darüber hinaus demonstrieren wir die praktische Anwendbarkeit unserer Lösung durch eine Integration in das österreichische E-Government-Produktivsystem.

Acknowledgements

First, I would like to thank Arne Tauber for his invaluable contribution during my master thesis. This work would not have been possible without his expertise and insights.

I am also most grateful to Thomas Lenz and other work colleagues, who sacrificed their time to provide me creative feedback.

Lastly, special thanks to my husband Aid, my family, and friends for their encouragement and support in every aspect of my life.

Contents

Abstract	iii
Acknowledgements	v
1. Introduction	1
1.1. Challenge	2
1.2. Contribution	3
1.3. Outline	3
2. Background	5
2.1. Electronic signatures	5
2.1.1. Advanced Electronic Signature.	6
2.1.2. Qualified Electronic Signature.	7
2.1.3. Basic principle.	8
2.2. Electronic signatures - standards	10
2.2.1. CAdES.	10
2.2.2. XAdES.	10
2.2.3. PAdES.	11
2.3. PDF basics	12
2.3.1. File structure	12
2.3.2. Signing process	17
2.3.3. Multiple signatures and Incremental update	19
2.3.4. PDF file verification	19
2.4. Summary of the chapter	20
3. Related Work	22
3.1. The Citizen Card Concept	22
3.2. PDF-AS	23
3.3. PDF-Over	24
3.4. QES on a single mobile device	26

Contents

3.5. Summary of the chapter	31
4. Model	32
4.1. Objectives	32
4.2. Participants	34
4.2.1. Signer environment	35
4.2.2. Server environment.	36
4.3. Communication flow	36
4.3.1. Interfaces	37
4.3.2. Process steps	37
4.4. Summary of the section	40
5. Implementation	41
5.1. PDF Processing	42
5.1.1. PDF Library	43
5.1.2. PDF signatures	46
5.1.3. Additional settings	51
5.2. Authentication process	54
5.2.1. Components	54
5.2.2. Communication flow	54
5.3. Summary of the chapter	62
6. Demonstrator	63
7. Discussion	69
7.1. Integration with Austrian productive solution	70
7.2. Evaluation of requirements	70
7.2.1. Security	70
7.2.2. Privacy	71
7.2.3. Usability	72
7.3. Summary of the chapter	73
8. Future work	74
9. Conclusion	76
A. A signed file	79

Contents

Bibliography

86

List of Figures

- 2.1. Digital signature: sign and verify. 9
- 2.2. Byte range of a signature. 18
- 2.3. Multiple signatures and an incremental update. 19

- 3.1. PDF-Over - desktop application for PDF signatures. 25
- 3.2. Generation of QES on a single mobile device. 28
- 3.3. Communication interface between components. 30

- 4.1. High-level model overview. 34
- 4.2. Overview of the communication process. 39

- 5.1. High-level implementation overview. 41
- 5.2. The appearance of the signature block. 44
- 5.3. Signature dictionary and the PDF signature. 47
- 5.4. PDF serial signatures. 48

- 6.1. The PDF file is open with the PDF signature application, and
the user is presented with the graphically rendered file. 64
- 6.2. Settings a user can select when signing a PDF file. 66
- 6.3. Authentication interface provided by the TSP App. 67
- 6.4. The signed file opened with a PDF viewer. 68

List of Tables

- 5.1. JSON parameters of the first request. "M" is mandatory field, while the "O" marks the optional fields in the request. 56
- 5.2. Parameters of the *payload* field. 56
- 5.3. Parameters of the *params* field. 57
- 5.4. The parameters contained in the response. 59
- 5.5. The parameters of *params* field. 59

Listings

- 2.1. Header section of the PDF file. 12
- 2.2. Body section of the PDF file. Objects structure. 13
- 2.3. Cross reference table of the PDF file. 13
- 2.4. Trailer of the PDF file. 14
- 2.5. Source code of the PDF file - shortened version. 15

- 5.1. Example of the signature profile. 44
- 5.2. Signing parameters of the file. 46
- 5.3. The creation of the signature dictionary and its entries. 50
- 5.4. The signature dictionary and its entries - shortened version. 50
- 5.5. Signature integration and incremental update. 51
- 5.6. The JSON body request from PDF Signature App to TSP. 55
- 5.7. The JSON request from TSP to PDF Signature App. 58
- 5.8. JSON request from PDF Signature App to TSP App. 60
- 5.9. JSON request from TSP App to PDF Signature App. 61

- A.1. Signed PDF file - source code shortened. 79

1. Introduction

Government sectors all over the world are employing the concept of electronic signatures to deliver agility, efficiency, cost, and time savings to their services. Electronic signatures, more specifically qualified electronic signatures (QES), provide secure and reliable authentication of users, as a prerequisite for accessing transactional public services as well as for signing electronic documents. Qualified electronic signatures are a special type of electronic signatures that must meet certain requirements and present an only signature standard that has a special status as being a legal equivalent to the handwritten signature [12]. Once the electronic signature fulfill the requirements of the qualified electronic signature, they become legal equivalent to the handwritten signature and assure data integrity, authenticity, and non-repudiation.

One of the most popular and universally recognized standards for electronic documents is the Portable Document Format (PDF). With trillions of PDFs existing in the world [3], this *de facto* format has undergone many changes since its initial version in 1993 [28]. Eight billion electronic and digital signature transactions processed through Adobe Document Cloud in 2017 corroborate the widespread use of this format in both private and public sector, mainly for signing contracts, invoices, submitting papers and in legal proceedings [25].

While the e-Government sector has provided a variety of applications for signing PDF files with qualified electronic signatures, it is noteworthy to mention that those traditional signature solutions are tailored specifically for stationary devices like personal computers and laptops. They have been in use for years and have reached a satisfying level of maturity and acceptance among users; however, they assume the possession of a smartcard, token, or smartphone in combination with web or desktop application for generating QES [9, 36, 29, 30].

1. Introduction

1.1. Challenge

Constant growth and popularity of the mobile industry changes these assumptions and set up new challenges for the e-services. The fact that a mobile handset is a common part of everyday life, in which mobile devices are more used to access the Internet than PC ¹, resulted in a transition from e-Government to mobile e-Government (m-Government) and allowed citizens to have convenient access to the information and public services from their smartphone [24].

These developments also posed a need for signing PDF files on mobile platforms. Although smartphones are generally suitable for the task [39], a simple solution providing PDF signatures remained insufficiently exploited inside the m-Government context. There is currently no solution for signing arbitrary PDF files on a single mobile device with QES, which strongly satisfies objectives such as security, privacy, and usability.

The main reason for this lack of technology is the incapability to migrate the existing concept into the new mobile environment. Smartcards and tokens, in contrast to stationary devices, cannot be used for authentication purposes on smartphones or tablets due to the hardware barrier. The server-based signature solution usually requires smartphones as a second-factor authentication. An additional obstacle is related to different security features between these two environments. While the need for a novel authentication concept was evident, for many years, it was hard to come up with the long-term stable solution due to constant changes in mobile technology [23, 37].

Finally, the recent advances such as secure element [20, 7] and biometric authentication brought new possibilities in this direction, and a new authentication method is introduced by Theuermann et al. [35]. The method is based on the user-friendly authentication against a remote Hardware Security Module (HSM) for creating QES on a single mobile device.

¹<http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet>

1. Introduction

This novel authentication concept was the missing brick for creating PDF signatures on a mobile device. In this thesis, we propose how to leverage this concept of generating QES and sign PDF documents on a single mobile device using user-friendly access to HSM.

1.2. Contribution

This thesis offers the following contributions:

1. We recognize the need for PDF signing service by evaluating the flaws and limitations of the current solution for signing PDF files on mobile devices. We conclude that the existing solution does not satisfy security objectives.
2. We propose a novel model for creating PDF signatures. Our model uses the server-side HSM for signature creation and the access to the HSM is granted by the local authentication executed on the single mobile device. We set up the objectives for our model, such as security, privacy, and usability.
3. We show the feasibility of our solution by implementing a proof-of-concept. The implementation consists of the Android application and can be divided into two parts. The first part is the implementation of the local PDF file processing, and the second part is the integration of the novel signing concept into our model.
4. We outline possible improvements in suggested concept as a part of the future work.

1.3. Outline

This thesis is organized into the following sections:

1. Introduction

- Section 2 (Background). We cover the state-of-the-art concepts as the base to our concept, such as electronic signatures, electronic signature standards, as well as the structure of PDF files.
- Section 3 (Related work) - We summarize the existing solutions which provide PDF signatures and outline the differences compared to our proposed solution.
- Section 4 (The model). We introduce the requirements and concept of our model, involved entities, a communication flow between them as well as the objectives the model aims to fulfill.
- Section 5 (Implementation). We provide a detailed description of implementation steps, as well as used technologies and standards.
- Section 6. (Demonstrator). We demonstrate the solution workflow from the user's perspective.
- Section 7 (Discussion). We discuss how the stated requirements have been met through our model and indicate the potential for further improvements.
- Section 8 (Future work). We summarize future research directions.
- Section 9 (Conclusion). We provide a conclusion of the thesis.

2. Background

In this section, we will provide the high-level knowledge of the technologies which build the basic and serve as a starting point of our proposed solution. We start with subsection 2.1, where we explain and introduce fundamentals in e-Government contexts such as electronic signatures and differences compared to advanced and qualified electronic signatures. Furthermore, in subsection 2.2, we introduce the AdES¹ family and explain what the signature formats are. In subsection 2.3, we will outline the basics of the PDF structure.

2.1. Electronic signatures

The core of e-Government architecture is to provide transaction-based services, which require secure and reliable authentication of the users. Once the user has been unambiguously proven her electronic identity, she can use those services.

The authentication is a compulsory step in vast of e-Government solutions, and due to its sensitive nature, it has always been a matter of great importance and continues improvement. Electronic signatures serve as a secure and reliable way for authentication of users and therefore play an essential role in e-Government affairs.

By its sole definition, an electronic signature means data in electronic form, which are connected to other data in electronic form and used by the signatory to sign [33]. In other words, we refer to an electronic signature as a paperless signature which provides a spectrum of benefits by supporting

¹Advanced Electronic Signature

2. Background

electronic communication and making e-Government services more agile. They assure:

- Authenticity - mapping the signed data to the signatory
- Data integrity - the content of the signed document has not been compromised
- Protection against repudiation, data manipulation, and forgery

The definition of electronic signatures is a quite general since it does not specify any requirements in terms of security or technology and intentionally represents an umbrella of different kinds of electronic signatures and uses where the proof of authenticity might not be engaged. Therefore, to obtain a bit narrowed definition of electronic signatures, eIDAS² has created standards which specify how electronic signatures can be used securely [33]. Regulation (EU) No 910/2014, or just eIDAS Regulation, has established a framework for electronic transactions. The aim of the regulations is to have convenient and secure electronic transactions across EU borders. The regulations are mandatory and wholly adopted in all EU member states, with precedence over any conflicting national laws.

2.1.1. Advanced Electronic Signature.

Advanced Electronic Signature [13, Article 26] is basically an electronic signature but with additional requirements:

- it is uniquely linked to the signatory
- it is capable of identifying the signatory
- it is created using electronic signature creation data that the signatory can, with a high level of confidence use under his sole control

²<https://www.docuSign.co.uk/learn/eidas-regulation-primer> electronic IDentification, Authentication and trust Services

2. Background

- it is linked to the data signed in such a way that any subsequent change in the data is detectable

To fulfill the stated requirements, advanced electronic signatures need to be created in a standardized way using public-key cryptography and infrastructures. Consequently, they pose a valid signature in legal terms, which are connected to the data in the way where any change of the data after the signature is detectable.

2.1.2. Qualified Electronic Signature.

For the understanding the concept of the Qualified Electronic Signature, we first need to define two terms: a qualified signature creation device or simply QSCD, and a qualified certificate. QSCD is a specific type of computer hardware or software able to generate and store cryptographic key material. This type of device is obliged to meet the requirements specified in Articles 29 and 30 in Annex II of the eIDAS Regulation[13, Annex II].

Smartcards are usually considered to be QSCD, since they store the signature creation data, for example, private keys and they never leave the device. The signature creation is executed inside the smartcards. The qualified certificate refers to the certificate for electronic signature issued by a qualified trust service provider and meets the requirements laid down in eIDAS regulations [13, Annex I]. The qualified electronic signature is an Advanced Electronic Signature created by a QSCD and based on a qualified certificate for electronic signatures. According to eIDAS Regulation Article 25 [13, Article 25], an electronic signature is a qualified if meets following three requirements :

- An electronic signature shall not be denied legal effect and admissibility as evidence in legal proceedings solely on the grounds that it is in an electronic form or that it does not meet the requirements for qualified electronic signatures.
- A qualified electronic signature shall have the equivalent legal effect of a handwritten signature.

2. Background

- A qualified electronic signature based on a qualified certificate issued in one Member State ³ shall be recognized as a qualified electronic signature in all other Member States.

From the above, we see that the creation of a Qualified Electronic Signature is not just adding a qualified digital certificate on an Advanced Electronic Signature, but the creation involves the use of a QSCD.

In the eIDAS regulation, Article 25 [13, Article 25], Qualified Electronic Signature has the same legal weight as a handwritten signature. In addition to that, Qualified Electronic Signature must be recognized in all EU Member States.

2.1.3. Basic principle.

Digital signatures ⁴ are based on public-key cryptography. The generation of two mathematically connected keys - public and private is done by a public key algorithm. The private key is used to encrypt signature-related data, and the public one is used for decryption.

Digital signature presents a one-way hash, calculated from the original data and encrypted with the signer's private key. The creation process consists of the following steps, and it is depicted in figure 2.1. First, the signer calculates the one-way hash value for the data which need to be signed. Then uses the private key to encrypt the hash.

The encrypted hash is the digital signature, which the signer along with the original data, is sending to the receiver. The signature is then to be verified. For this purpose, the receiver uses the signer's public key to decrypt the hash, then generates a new one-way hash of the signature input data. If this hash matches the original one, the data is not modified after it was signed.

³https://europa.eu/european-union/about-eu/countries/member-countries_en

⁴While the electronic signature presents a legal concept, digital signatures, on the other hand, refers to cryptographic technology on which an electronic signature solution is built. They secure the data connected with a signed document and provide the verification of the authenticity of a signed data. <https://www.esignlive.com/blog/the-difference-between-e-signatures-and-digital-signatures>

2. Background

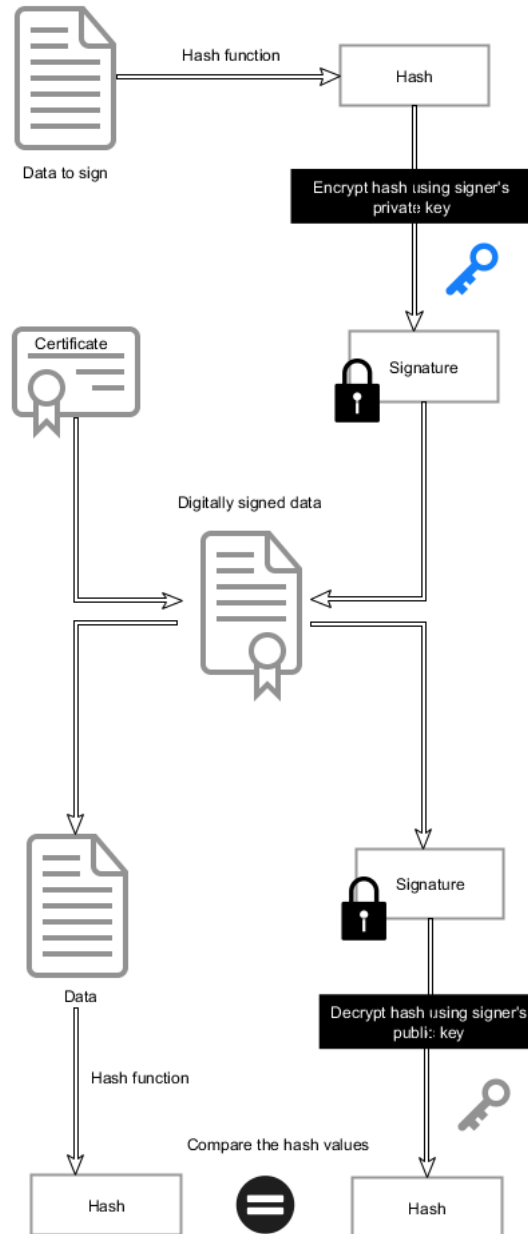


Figure 2.1.: Digital signature: sign and verify.

2. Background

2.2. Electronic signatures - standards

Electronic signatures, as a legal analog to the handwritten ones, can be validated in any of the Member States. The raised challenges in terms of the interoperability of the process served as a motivation for relying on open standards and through an eIDAS regulation couple of electronic signature baseline profiles have been defined.

One additional aspect has been tackled in the process of standardizing electronic signature profiles - long term validation. It means the ability to verify the signature a long time after it has been created (e.g., 100 years after). This approach should take into account the potential problems which could occur in the future, like loss of key information needed for validation of a weakening of cryptographic methods and therefore handle the obstacles.

The following signature formats have been defined by the eIDAS regulation:

2.2.1. CADES.

CMS Advanced Electronic Signatures (CADES) is defined as an extension to the Cryptographic Message Syntax (CMS) signed data which makes them suitable for Advanced Electronic Signature [16]. CMS is a standard for cryptographically protected messages, based on the public key cryptography. CADES has multiple profiles available, starting from the basic CADES (CADES-B), and ending with a CADES-LTA, an advanced level which supports long term validity.

2.2.2. XAdES.

XAdES stands for an XML Advanced Electronic Signatures and represents a standard for digital signatures which are XML formatted [15]. Similarly to the prior profile, we differentiate Basic Electronic Signature defined in the

2. Background

XAdES-BES. In addition, XAdES also supports long term validity at more advanced levels.

2.2.3. PAdES.

PAdES stands for a PDF Advanced Electronic Signature and presents a set of extensions as well as restrictions defined under ISO 32000-1 for PDF, making it, similarly to prior profiles, suitable for Advanced Electronic Signatures [14]. Since PDF supports the framework for digital signatures, PAdES defines the signature profile, which is compliant with eIDAS regulation.

PAdES is a supplement to the other two profiles which have been mentioned - CAdES and XAdES, but unlike them, applies only to the PDF files. It can be interpreted as a set of requirements which must be met by a software which is processing PDF files with the purpose of integrating digital signature.

One of the advantages of PAdES is the ability to be easily deployed in PDF processing applications without any additional technological requirements. The signature data is incorporated directly into PDF signed file and since PDF is a view-able format and generates a human-readable document, the signature may have its own graphical representation displayed on the document. This is one of the main features which distinguish PAdES standard from CAdES and XAdES, which are usually not used in the forms which involve visual representations of the signature.

PAdES is also supporting long term validation, since it has been acknowledged that digitally signed files can be and are in use for decades, regardless of technological changes, and therefore must have the ability for a validation check in the time they were created.

2. Background

2.3. PDF basics

Portable Document Format, or simply PDF [6] is specified by ISO32000⁵ and presents a standard for the digital representation of documents.

It is used by individuals, in businesses, governments, and many other institutions around the world as an easy and reliable means for viewing and exchanging electronic documents.

2.3.1. File structure

One of the premises of PDFs is the independence of the environment in which they were created, viewed, or printed. This also requires a strict structure of how the document is created, manipulated, and displayed to the user. The PDF file is structured into four main parts:

- **Header.** The first line which specifies the version of PDF used is called *header*. This is also the only line contained in this section, and it is a comment. From listing 2.1 below, we can see that the file refers to the PDF version 1.4.

Listing 2.1: Header section of the PDF file.

```
1 %PDF-1.4
2 %..
```

- **Body.** The body part starts below the header and above the line *xref* and presents the actual content that will be displayed. *Objects* are the main parts of the body, and they have a certain structure. They start with a *object number* - which is incremented for every new object, then *generation number* - which is 0 for all objects not extended with incremental update and end with the string "obj.". The example of the *Body* section is depicted on listing 2.2.

⁵<https://www.iso.org/standard/51502.html>

2. Background

Listing 2.2: Body section of the PDF file. Objects structure.

```
1 73 0 obj
2 << /Type /Pages
3 /Count 2
4 /Kids [71 0 R 128 0 R ] >>
5 endobj
```

In general, everything in PDF is an object, except for some parts like header, cross-reference table and some parts of the trailer. One of objects are *Dictionaries*, *Streams*, *Arrays*.

- **Cross reference table.** This table, depicted on listing 2.3, has information about all PDF objects. Since every object in the PDF has a cross-reference entry, this table provides quick access to them. It starts with a *xref*. After this line, we have a new line consisting of two characters: the first one indicates the starting object, and the second one corresponds to a number of entries - in our case 0 and 4.

Listing 2.3: Cross reference table of the PDF file.

```
1 xref
2 0 309
3 0000000000 65535 f
4 0000000015 00000 n
5 0000000588 00000 n
6 . . . .
7 0000067372 00000 n
8 0000067832 00000 n
9 0000067877 00000 n
```


2. Background

- **Trailer.** The last section of PDF file starts with keyword *trailer* and gives us the overall information about PDF file. It is used for the quick access of the cross-reference table and documents special objects like *Catalogs*. Document Catalog presents the root of all objects which are contained in the file. The entries of this component contain information about PDF. Since the PDF file is processed into memory from end to the beginning, *trailer* is the first processed part of PDF. It contains a dictionary with a minimum of two entries: root and size. The overview of the trailer is depicted on listing 2.4.

Listing 2.4: Trailer of the PDF file.

```
1 trailer
2 <<
3 /Root 306 0 R
4 /Info 1 0 R
5 /ID [E148BC519F508DF64C4F38F6086FB77E> E148BC
6 519F508DF64C4F38F6086FB77E>]
7 /Size 309
8 >>
9 startxref
10 67958
11 %%EOF
```

Listing A.1 provides the source code of the arbitrary PDF file. We aim to demonstrate the main parts of the document, such as header, body, cross-reference table, and trailer.

2. Background

Listing 2.5: Source code of the PDF file - shortened version.

```
1 %PDF-1.4
2 %..
3 1 0 obj
4 <<
5 /Title <FEFF0050007200FC006600620065007200690063006
6 80074>
7 /Author <FEFF005300690067006E0061007400750072002F00
8 5300690...60075006E0067>
9 ....
10 /Producer (Apache FOP Version 2.3)
11 /CreationDate (D:20190115111335+01'00')
12 >>
13 endobj
14 2 0 obj
15 <<
16 /N 3
17 /Length 3 0 R
18 /Filter /FlateDecode
19 >>
20 stream
21 .....
22 endobj
23 152 0 obj
24 << /S /P /P 150 0 R >>
25 endobj
26 73 0 obj
27 << /Type /Pages
28 /Count 2
29 /Kids [71 0 R 128 0 R ] >>
30 endobj
31 306 0 obj
32 <<
33 /Type /Catalog
34 /Pages 73 0 R
35 /Lang (x-unknown)
```

2. Background

```
34 /MarkInfo << /Marked true >>
35 /StructTreeRoot 148 0 R
36 /Metadata 5 0 R
37 /PageLabels 307 0 R
38 /ViewerPreferences << /DisplayDocTitle true >>
39 >>
40 endobj
41 72 0 obj
42 <<
43 /Font << /F16 134 0 R /F15 143 0 R >>
44 /ProcSet [/PDF /ImageB /ImageC /Text]
45 /ColorSpace << /DefaultRGB 4 0 R >>
46 >>
47 endobj
48 148 0 obj
49 <<
50 /Type /StructTreeRoot
51 /K [147 0 R]
52 /ParentTree << /Kids [308 0 R] >>
53 >>
54 endobj
55 307 0 obj
56 << /Nums [0 << /S /D >>] >>
57 endobj
58 308 0 obj
59 << /Nums [0 9 0 R 1 66 0 R 2 75 0 R 3 122 0 R] /
    Limits [0 3] >>
60 endobj
61 xref
62 0 309
63 0000000000 65535 f
64 0000000015 00000 n
65 0000000588 00000 n
66 ....
67 0000067372 00000 n
68 0000067832 00000 n
69 0000067877 00000 n
```

2. Background

```
70 trailer  
71 <<  
72 /Root 306 0 R  
73 /Info 1 0 R  
74 /ID [<E148BC519F508DF64C4F38F6086FB77E> <E148BC519F  
75 508DF64C4F38F6086FB77E>]  
76 /Size 309  
77 >>  
78 startxref  
79 67958  
%%EOF
```

2.3.2. Signing process

One of the special features of PDF, starting from version 1.3, is the ability to add digital signatures, as a means to guarantee integrity, authenticity, and non-repudiation of electronic documents.

PDF defines signatures to be embedded into the document itself. This allows viewing applications to modify the file without compromising it [5]. The signing process consists of the following steps:

- We turn the signing document into the stream of bytes.
- The entire document is written to the disk which has a space left for signature value or with values defined in *ByteRange* array. *ByteRange*, depicted on figure 2.2, is an array containing four numbers, precisely two pairs of bytes. The first number in a pair presents the offset in a file of the beginning of byte streams, which need to be included in the hash. The second number defines the length of the stream. The pairs specify the sequence of the bytes which need to be hashed, while the actual signature value is stored in the *Contents* section.
- When we know the location of signature, the *ByteArray* is overwritten with correct values.

2. Background

- With the hash algorithm and the bytes defined by the *ByteRange*, the hash of the entire document is calculated.
- We encrypt this hash value with the signers private key, and we generate a hex-encoded PKCS#7 signature object.
- The */Content* placeholder is overwritten with the signature object and placed on the disk.

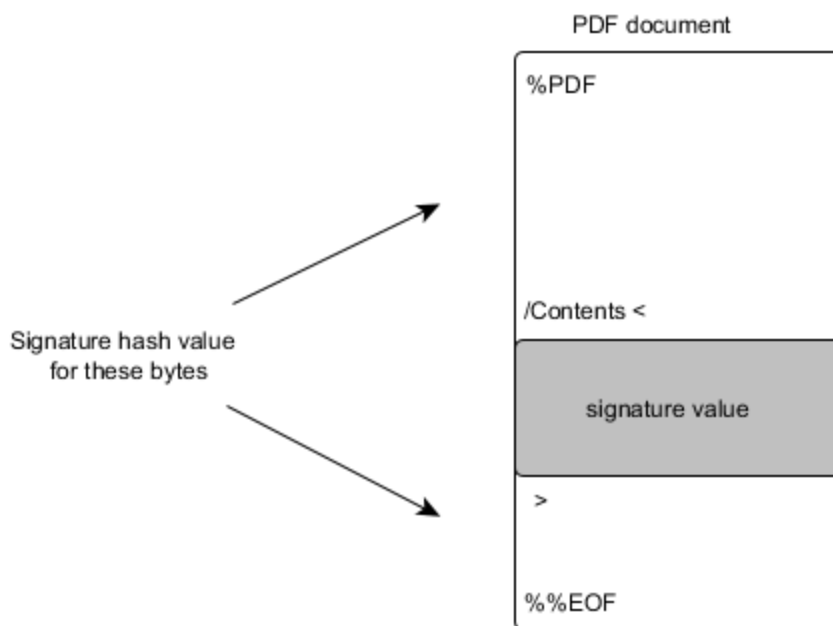


Figure 2.2.: Byte range of a signature.

2. Background

2.3.3. Multiple signatures and Incremental update

One of the features of PDF signatures is the detection of modification of the signed file. This means that any attempt to alter the signed PDF document will invalidate the existing signature. However, sometimes, it is required to sign files multiple times and to add multiple signatures. PDF feature that tackles this issue is called incremental update [10].

The incremental update capability allows modification to the PDF file by adding the information about the modification to the end of the file in a special incremental update section. This way, the byte representation of the previous signature will not be altered. This allows new signatures to be added without changing the previous version and invalidating existing signatures.

On figure 2.3, we can see how a signed file is structured. Incremental update modifies the original document by adding the new *body*, as well as the new *cross-reference table* and new *trailer*. The same procedure is then applied for each incremental update.

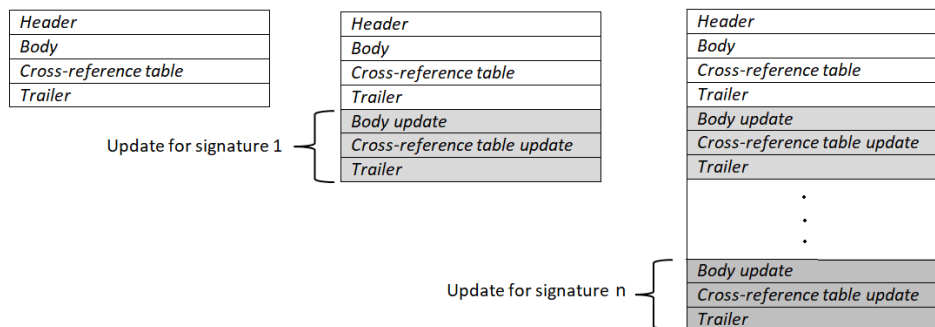


Figure 2.3.: Multiple signatures and an incremental update.

2.3.4. PDF file verification

Once the document is signed and opened with an application which supports digital signatures, the signature validation process starts [5]. This process

2. Background

involves the following steps:

- To make sure that the document is not altered after it has been signed, the corresponding application (e.g., Adobe Acrobat) is comparing the hash value from the signature with a calculated hash value of the signed document. Inequality of hash values means that the document has been corrupted, which results in notifying the user about the outcome. However, the process continues.
- The signer's certificate is checked to confirm its connection to a trusted anchor. For this purpose, a certificate chain is built, and the requirement in this step is that at least one path is found from the signer certificate to a trust anchor.
- In this step, the revocation checking is performed for an end-entity certificate to verify that the signer's certificate was valid at the time of signing. This is performed either using Online Certificate Status Protocol (OCSP) ⁶ or Certificate Revocation Lists (CRL) ⁷.
- Lastly, the timestamp is validated. If the timestamps have been embedded into signature from a trusted server, steps 1-3 are performed on the certificate for the timestamp.

2.4. Summary of the chapter

Electronic workflows all over the world involve signing PDF files, as they present electronic analog to paper and provide a portable and simple electronic form of information to users. Electronic signatures, in combination with PAdES standard, provide long term authenticity and therefore, make organizations more agile and flexible in terms of business requirements [34]. There is a vast number of use cases for signing PDF files since this form of a paperless signing can be used in the same range as the ink signature. Additional advantages are that PDF provides a visual representation of the document as well as the

⁶<https://tools.ietf.org/html/rfc6960>

⁷<https://tools.ietf.org/html/rfc5280#section-5.3.1>

2. Background

allocation of a position where the signature will be placed. Finally, users have a variety of applications to use for creating and verifying PDF signatures, and these applications will be elaborated in the following section.

3. Related Work

In this section, we will introduce the existing state-of-the-art concepts for qualified PDF signatures to outline differences between approaches. We will mostly relate to the Austria solution in terms of qualified electronic signatures. We will start with their state-of-the-art concept - The Citizen Card Concept, furthermore, we will introduce two software applications for PDF signatures, and finally, we will provide more information about novel authentication methods.

3.1. The Citizen Card Concept

The Austrian concept for transferring the identity of a user into an electronic identity is described in a concept which is called The Citizen Card Concept [32],[22]. This concept is a core component in e-Government processes, and one of its significant features is the independence of technologies and tokens. For the authentication purposes needed by accessing remote services, Citizen Card can be implemented and used in two forms. In the first approach, the Citizen Card is a smartcard - this can either be a health insurance card, bank card, student card, or similar. In the second approach, the Citizen Card is implemented as a Mobile Phone Signature. Regardless of the option, both of the implementations are used for two main functions - identification and authentication of users.

Mobile Phone Signature. The Mobile Phone Signature [26] presents one form of implementing the Citizen Card. A significant feature of this solution is the fact that it uses a Hardware Security Module (HSM), located on a server-side as a secure signature creation unit. The signature creation material, such as private keys, is managed by the certification service provider

3. Related Work

(A-Trust) and accessible only by the users. Access to private keys is protected by the multi-authentication mechanism. At the beginning of the authentication process, the user is prompted to provide the phone number and password, which presents the factor knowledge. The second factor of authentication, possession factor, is embedded in the form of token which a user obtains on their mobile phones. This can be either a QR code or a TAN which is received via SMS. Once the input data have been positively confirmed, the user has been successfully authenticated.

Smart card solution. Smart card solutions, on the other hand, require a smart card, which could be, for example, health insurance card and citizen card software. Between the citizens who evoke a process of authentication and the server-side which presents a trusted party, we have a middleware. The middleware is called citizen card environment, and it is locally installed on users personal computer. The middleware is a layer, which decouples all the technical details and security features from both parties and that way simplifies the communication process. The communication between e-Government application and citizen is defined according to the security layer interface ¹.

3.2. PDF-AS

PDF-AS [29] is a framework used for digital signing and verifying PDF files. The created signatures are compliant with the PAdES standard [17]. PDF-AS implements an authentication based on the Citizen Card Concept - Mobile Phone Signature and Smart Card, however different signature creation units can be also used such as MOA-SS², a PKCS#12³ file or a Java KeyStore⁴.

Regarding the architecture, PDF-AS consists of three components:

¹<https://www.egiz.gv.at/en/e-government/4-buergerkarte#sub-securitylayer>

²<https://joinup.ec.europa.eu/solution/moa-spss>

³<https://tools.ietf.org/html/rfc7292>

⁴<https://docs.oracle.com/javase/7/docs/api/java/security/KeyStore.html>

3. Related Work

- PDF-AS Library - Java library, which is the core component and implements the most important functionalities for PDF processing. Although they can be observed as separate, the other components of the framework strongly rely on this library.
- PDF-AS Web - Web interface for PDF-AS, usually integrated into an external application which provides PDF signature.
- PDF-AS Client - Provides a Command-line interface for PDF-AS.

PDF-AS independently verifies the signatures, but it does not include a certificate check. In addition, PDF-AS represents a modular solution, which allows other applications to be easily built upon PDF-AS or to be integrated into other applications.

3.3. PDF-Over

PDF-Over [30] is a Java desktop application used for signing PDF files. It is implemented as a graphical interface for the user who can select PDF file and then manually choose a position of the signature on the file. To create the signed PDF document, the user can choose to use the Mobile Phone Signature or a Citizen Card.

PDF-Over is based on the previously explained PDF-AS Library, but unlike PDF-AS, PDF-Over provides the visualization in the entire process of creating a signature, which makes it suitable to use as a standalone application. The objectives of this solution are interoperability and modularity. It is available for Windows, macOS, and Linux, and on figure 3.1 we provide a screenshot of the software.

3. Related Work

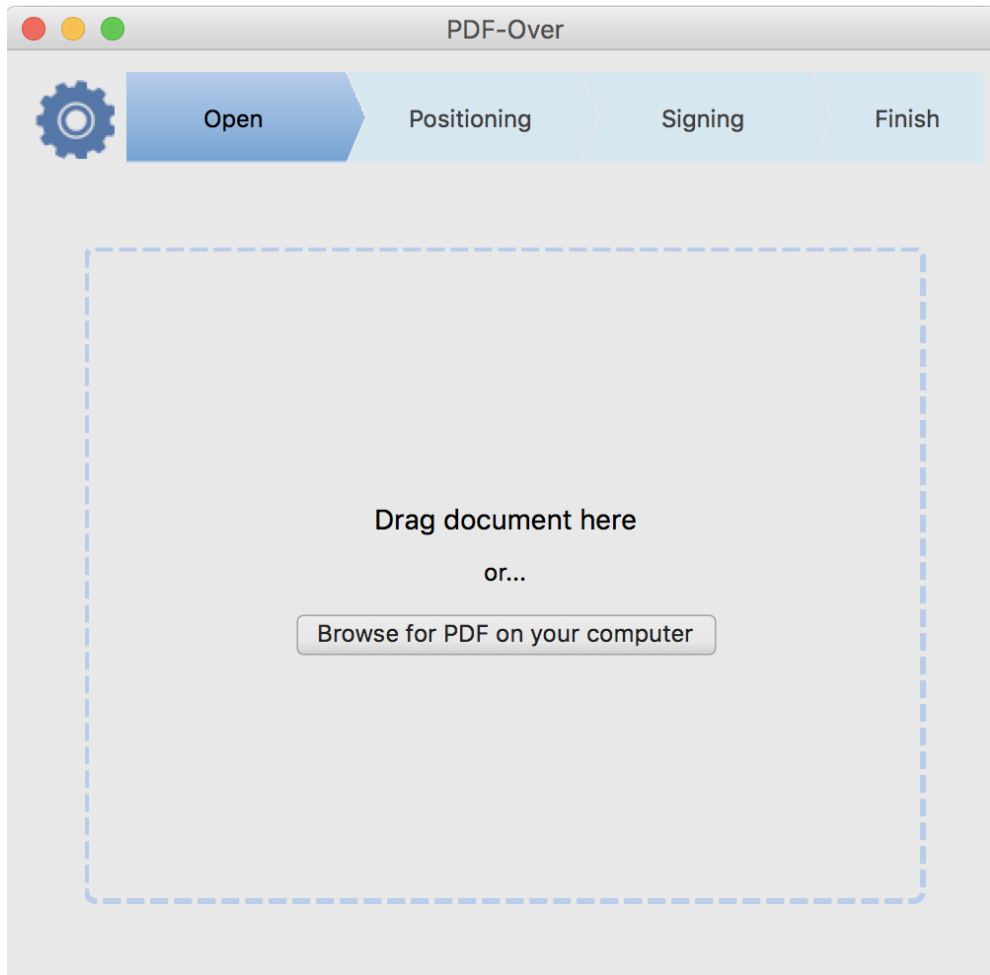


Figure 3.1.: PDF-Over - desktop application for PDF signatures.

3. Related Work

3.4. QES on a single mobile device

The traditional scenario for creating legally binding PDF documents with the Austrian Mobile Phone Signature starts on stationary devices and involves two-factor authentication employed on two different end-user devices (most commonly PC and smartphone).

However, as we move towards a mobile future, we aim to provide mobile-only transactional services, which faces quite some challenges. An attempt to use the same state-of-the-art authentication mechanism on a single mobile device so far could not be considered successful.

In this case, it is not possible to use smartcards for a qualified electronic signature generation, since smartcards require card readers, who are typically inserted into PCs; however, mobile devices do not support these technologies. Another approach would be to use server-based signature solutions. One of those solutions is Mobile Phone Signature, where the Hardware Secure Module (HSM) is used as a qualified signature creation device. The access to the HSM is provided when the user enters both the password as a knowledge factor, and one-time password (OTP) in the form of a QR code or TAN send as an SMS to the mobile phone. These two authentication factors use two different communication channels, and this separation of channels is an important security factor.

A use-case where the server-side signature solution is used for QES generation on a single mobile device is theoretically possible. A user could access the web service for signing PDF files from the mobile browser. That way, the user would have to do the authentication process entirely on the smartphone or tablet. The phone number and password would have to be entered from the same phone as the one on which she receives OTP by the same communication channel. As smartphones are designed to only participate as a second factor of authentication, in such a setting we do not have multi-factor authentication. This approach, even if it appears to utilize the second-factor authentication, compromises the security of the process [27] and it is not considered as a high-level security solution. Thus, it is forbidden to be used that way by the Trust Service Provider.

Due to the differences in security features, it has become evident that the existing concept tailored for stationary devices could not be just simply mapped

3. Related Work

into a concept for a mobile device. The new authentication solutions which provide multi-factor authentication on mobile devices are required.

Nevertheless, the possibilities in this area have been changed with recent developments of the mobile industry. One of the most significant technological contributions of new generation mobile devices is certainly their support of Trusted Execution Environments(TEE) [19], e.g., with embedded Secure Element (SE), and biometric authentication towards the TEE. TEE presents dedicated, separate tamper-resistant hardware which securely stores cryptographic key material (e.g., keys). The access to key materials is enabled by the local authentication of a mobile device. The local authentication has been considerably improved in the last decades with the support of alternative biometric authentication methods such as fingerprint, iris scanner, face recognition. This enabled the multifactor authentication on the single mobile device by combining different methods like ownership of the cryptographic key material and knowledge of a password. The combined methods of local authentication and secure element finally resulted in designing and implementing a novel authentication concept at server-side HSM for creating qualified signatures on a mobile phone [35]. The authors argue that not only the solution is feasible, but it also provides high-level security with increased user-friendliness.

Basic principle. The generation of the remote qualified electronic signature using a single mobile device presents an eIDAS compliant solution, which is rolled out in Austria on the national level. Figure 3.2 depicts the main components of the novel authentication concept. The whole process distinguishes Signer and Trusted Service Provider (TSP) environment.

3. Related Work

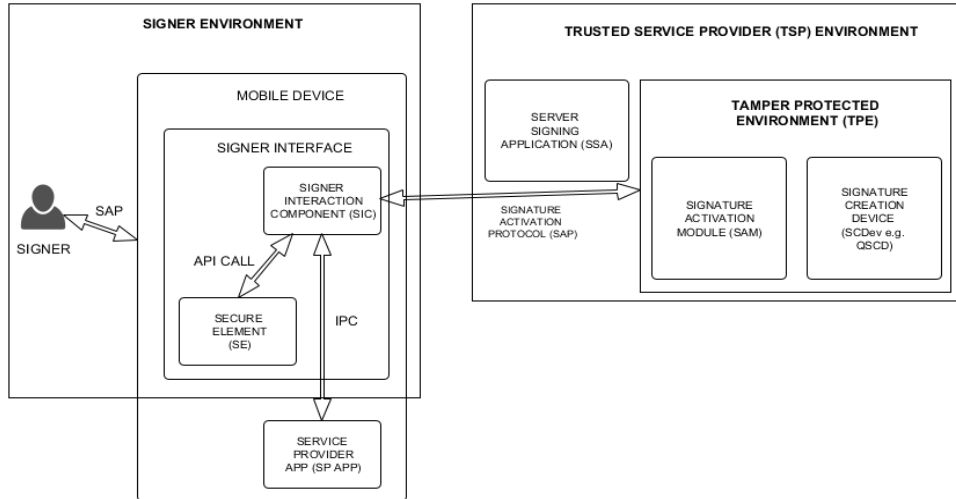


Figure 3.2.: Generation of QES on a single mobile device.

The following components are contained in the signer environment.

- **Service Provider App (SP App).** - represents the client application from the Service Provider, which require a qualified electronic signature for the authentication purpose. From figure 3.2 we see that the SP App is establishing the inter-process communication with the Signer Interaction Component (SIC) client.
- **Signer Interface.** The signer interface presents the component which is in charge of interacting with a signer. This component consists of the Signer Interaction Component, locally installed on the mobile device, and Secure Element.
- **Signer Interaction Component (SIC).** The SIC presents a mobile client application provided by the Trust Service Provider. This App establishes an IP communication with the client from the Service Provider and as well API call with the Secure Element.
- **Secure Element.** iOS Security Guide [20] defines the Secure Element, or

3. Related Work

precisely Secure Enclave, as a "coprocessor fabricated within the system on chip (SoC)". The SE provides data integrity even in the case of the compromised kernel. It has an encrypted memory and hardware random number generator. The Secure Enclave handles the cryptographic operations required for the key management. This component is also taking responsibility for processing biometric factors of authentication like fingerprint and face data from the Touch ID and Face ID sensors. The access is granted to the user when the match is determined.

From the Android documentation [7], it is stated that the SE is tamper-resistant hardware into devices used for storing cryptographic data, and thus, provides better security features. Android also provides KeyStore/KeyChain as a way to manage cryptographic key pairs and from the version 6.0 the API, which determines wheater the keys are kept in the SE, is provided.

Overall, TEE is considered to be safely used for remote authentication, since the key material is unlikely to be extracted and abused. The access to the private key stored in the TEE is granted via local authentication, such as fingerprint

TSP environment components are the following:

- **Server Signing Application (SSA)** The SSA is a communication channel between SIC and SAM, which interprets the commands from the SIC, and therefore, to perform the authentication process to obtain the QES.
- **Signature Activation Module (SAM)** This component provides the users' control of the keys. As it is known, the private key from the user is kept inside the Qualified Signature Creation Device and protected by the signature PIN.

3. Related Work

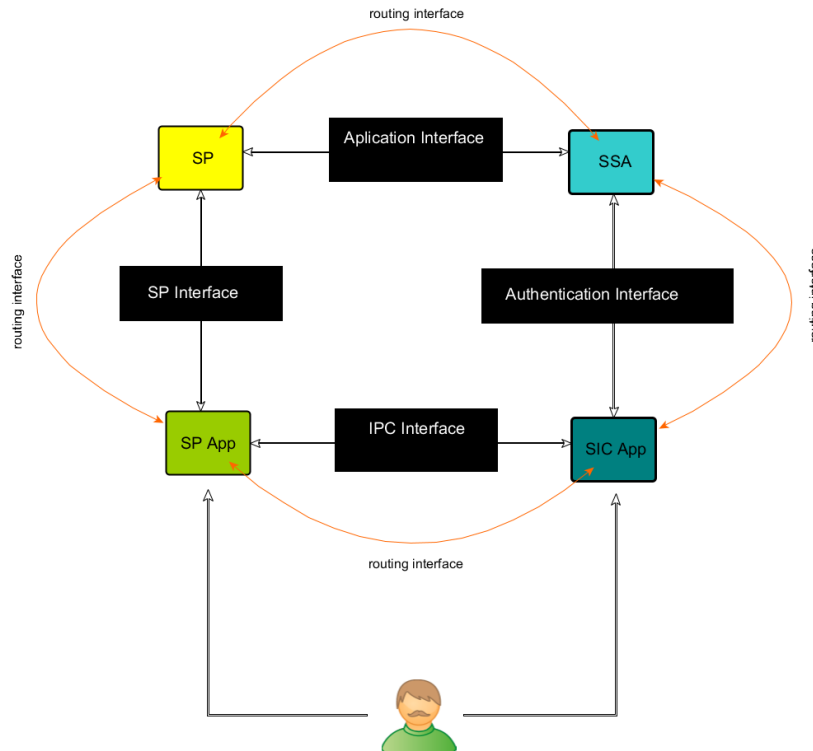


Figure 3.3.: Communication interface between components.

Now that we have introduced the main entities of the concept, we need to explain the communication flow between each of them. The interaction between entities is depicted in figure 3.3 . We define the following:

- **Routing interface** The routing interface presents a generic interface between participants of the process.
- **User interface** The interaction between the user and its device is captured in the User interface. The user interacts with the applications installed on the device in order to get the service she opts.
- **Application interface** The communication between Service Provider

3. Related Work

and Service Signing Application is defined via Application interface and is implemented as an HTTP request(response) with JSON elements through TLS connection.

- **Authentication interface** This interface defines the authentication protocol between Server Signing Application and the Signer Interaction Component. After the Service Provider has initiated the authentication process at Server Signing Application, SSA starts the authentication process by sending the authentication command to the SIC App, which after successful execution, returns the result. The resulting QES is redirected to the SP via SP App.
- **Interface to the service provider** The interaction of the application towards the service provider is defined by this interface.
- **API or IPC call** Depending on the implementation, the interaction between two Apps is done either by interprocess communication (IPC) or by API call.

3.5. Summary of the chapter

In this chapter, we have described the related applications for signing PDF files. However, we have also outlined that those applications, although very popular, have been used in a more traditional approach having mobile devices as a second authentication factor. We also introduced the novel authentication method by [35], which enables QES on a single mobile device. In the next chapter of this thesis, we will demonstrate how we used the described process for signing PDF files.

4. Model

In this thesis, we propose a usable and privacy-preserving model for signing PDF files on mobile devices. Such a model is built upon certain objectives and involves an interaction between components. In this section, we present our model by explaining the objectives, involved entities, and the communication flow between them.

The organization of this chapter is the following: In subsection 4.1, we introduce the objectives of our model. Furthermore, subsection 4.2 describes the involved entities in further detail. In subsection 4.3, we explain the interaction between involved components. Finally, in subsection 4.4, we summarize the concept shortly.

4.1. Objectives

The goal this thesis aims to fulfill is strongly based on the objectives we set up. They serve as a requirement for further architectural and implementational decisions and also provide a distinguished model which facilitate the gap from the prior work. We will outline three main objectives and explain their relevance.

Privacy. In an environment like e-Government, where e-documents with sensitive personal data are handled, and the misuse of private data can lead to violation of citizens and business rights, privacy is among the main concerns [32]. In this thesis, we set privacy as one of the main requirements of our model, since we argue that this aspect has been neglected in prior work with PDF signatures. The privacy-preserving model should emphasize the minimum data disclosure to the third-party apps. Designing the model for

4. Model

signing PDF files that satisfies this objective implied that no unnecessary third parties can be included in the signature creating process. To avoid relying on the external services, all PDF processing has to be implemented locally inside the application. This means that the document to be signed by a user is not sent to the web services, which provide PDF processing. Only the user with its mobile phone and the TSP should be involved.

Usability. Usability, as a merit for user adoption, is another requirement of our model. In general, the e-Government sector heavily relies on a citizen's experience with public services. The citizen's satisfaction results in a higher level of acceptance and usage of e-Government services, increasing the cost savings and flexibility of e-Government [38]. Intuitive use of the application, as well as the minimum number of interactions and minimum execution time, is required, as the user aims to complete the signing process as fast as possible. Easy means of authentication are also requirements of the model; thus, the minimal memorization of user credentials is a prerequisite for the overall user experience.

Security. Security is another perimeter that has a strong impact on the overall citizen's communication and engagement with e-Government. As one of the objectives of our model, security often comes with a trade-off with usability as the more complex authentication often degrades the user experience. In our model, we seek to find a balance between two opposite but fundamental objectives. The strong and reliable multifactor authentication is provided by the concept of the QES on mobile devices. Furthermore, we focus on satisfying the security aspect from the application context.

To summarize, we named several objectives which need to be taken into consideration when designing a solution for PDF signatures on mobile devices. In addition to the proposed objectives, we tend to make our model as generic as possible, so that it can be easily implemented on both Android, iOS, or any other mobile platform. The model we propose needs to be as modular in such a way that it can be easily integrated into another project and also used as a standalone solution.

4. Model

4.2. Participants

In this section, we will provide a high-level description of the entities who take part in the proposed model. We will define two environments. First one is the signer side, which consists of a user and the applications with which the user is interacting to obtain the requested service. The second part is the TSP server-side with only one component. In figure 4.1, we have provided a high-level overview of the participants in our model.

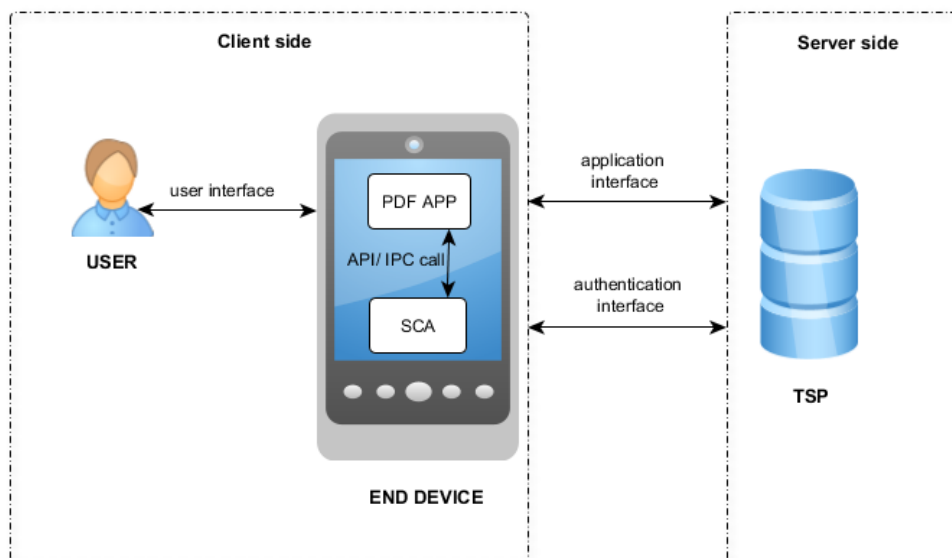


Figure 4.1.: High-level model overview.

4. Model

4.2.1. Signer environment

User and applications, with which the user is interacting form the signer environment. This environment also presents the client-side of the implementation, and from the user perspective, the entire process is executed on this side. The user, which requires the service, first interacts with the Service Provider, which is, in our case an application for signing PDF files. Furthermore, Service Provider delegates the authentication process to a client application of the Trust Service Provider. The client application from the Trust Service Provider is called a Signer Interaction Component and executes the user authentication against the Trust Service Provider.

User. Although the user does not entirely belong to the model, it is important to define the role of a user (signer), as an initiator of the process. In a general model, a user is a natural person who interacts with a Service Provider App in order to get the requested service. In our case, the user initiates the signing process from by selecting the desired PDF file to be signed and interacting with corresponding applications.

Mobile device. The mobile device presents a device such as a smartphone, tablet, or similar. This component has the role of hosting the applications and does not depend on any exact specifications. However, it should support the TSP requirements for TEE and appropriate alternative authentication methods such as fingerprint, iris scanner, face recognition, or similar.

PDF signature application. The PDF signature application has the role of a service provider and provides the signing of the PDF files. This is a mobile application which decouples the interaction with all other components of the model, such as user as well as with a Signer Component Application and Trust Service Provider. The PDF Signature Application consists of two parts. The first part is processing PDF files for signature creation, and a detailed description is provided in section 5. In the second part, this application initiates the signature process. The authentication process, which starts from this App, is delegated to the Signer Component Application to get the signature. Finally,

4. Model

the application provides the signed document, when the authentication against the TSP is terminated successfully.

Signer Component Application (SCA). The Signer Component Application presents the mobile client part (application or library) from the Trust Service Provider. The SCA implements the communication with both TSP and PDF Signature Application, which makes this component as an intermediate. The Signer Component Application receives the PDF signing request from a PDF Signature Application and performs an authentication of the signer against the Trust Service Provider.

4.2.2. Server environment.

In contrast to the client-side, the server-side of the process is hidden from the final user, and it presents a remote authentication at the HSM in order to complete the PDF signature process. This environment has only one entity, namely the TSP.

Trust Service Provider (TSP). Trust Service Provider is a participant who is generating the eIDAS compliant QES after the authentication performs successfully.

4.3. Communication flow

The complete process of generating signed PDFs consists of the authentication process against the TSP and the PDF processing part. While PDF processing is done within the PDF signature App, the authentication requires the interaction of the user with other components of the model. The high-level view of the interaction established by the components is illustrated in figure 4.2.

4. Model

4.3.1. Interfaces

- **User interface.** This interface describes the communication between a user and the end device. At the beginning of the signing process, the user is interacting with the PDF Signature APP by selecting the desired pdf file for the signature process. For authentication purposes, the user is interacting with the TSP App.
- **Application interface.** Application interface defines the communication which PDF signature APP is using for the interaction with TSP, with an aim to create a qualified signature on PDF file. This phase starts after the pdf document has been processed and prepared for signature integration.
- **API or IPC call.** The PDF signature App is interacting with the TSP App via IPC call if they are separate apps or via API call if they are run within the same application.
- **Authentication interface.** This interface provides the authentication of users by TSP via TSP App. The user is required to enter its credentials, and after the authentication is successful, the TSP can provide the desired service.

4.3.2. Process steps

1. A citizen wants to sign a PDF file on a mobile device. To initiate the process, the citizen selects the PDF and sends it to the PDF Signature App.
2. The PDF Signature App, receives and prepares the document, accordingly, for inserting the signature. After completing this part, the App initiates the required authentication process for obtaining the signature by sending a request to the Trust Service Provider. The request contains parameters, which indicate what service is required from the TSP.
3. The Trust Service Provider, after receiving the request and analyzing parameters, sends the response back to the PDF Signature App. The

4. Model

response should contain the URL for IPC (API) call, which redirects the PDF Signature App to the client app from the Trust Service Provider.

4. PDF Signature App receives the redirect and establishes an interaction with the Signer Component Application. The interaction, depending on the implementation and request specifications, can be Inter-process Communication (IPC) towards the other App. This way, the PDF Signature App delegates the authentication process to the Signer Component Application.
5. The following step includes the interaction between the Signer Component Application and the Trust Service Provider, to authenticate the user.
6. The user is prompt to provide the credentials for the authentication process.
7. After the authentication is completed successfully, Trust Service Provider sends the response with the remotely (by the HSM) created signature to the client App.
8. The response is forwarded to the PDF Signature App from the Signer Component Application. The PDF Signature App uses the signature to complete the process. The file is successfully signed.

4. Model

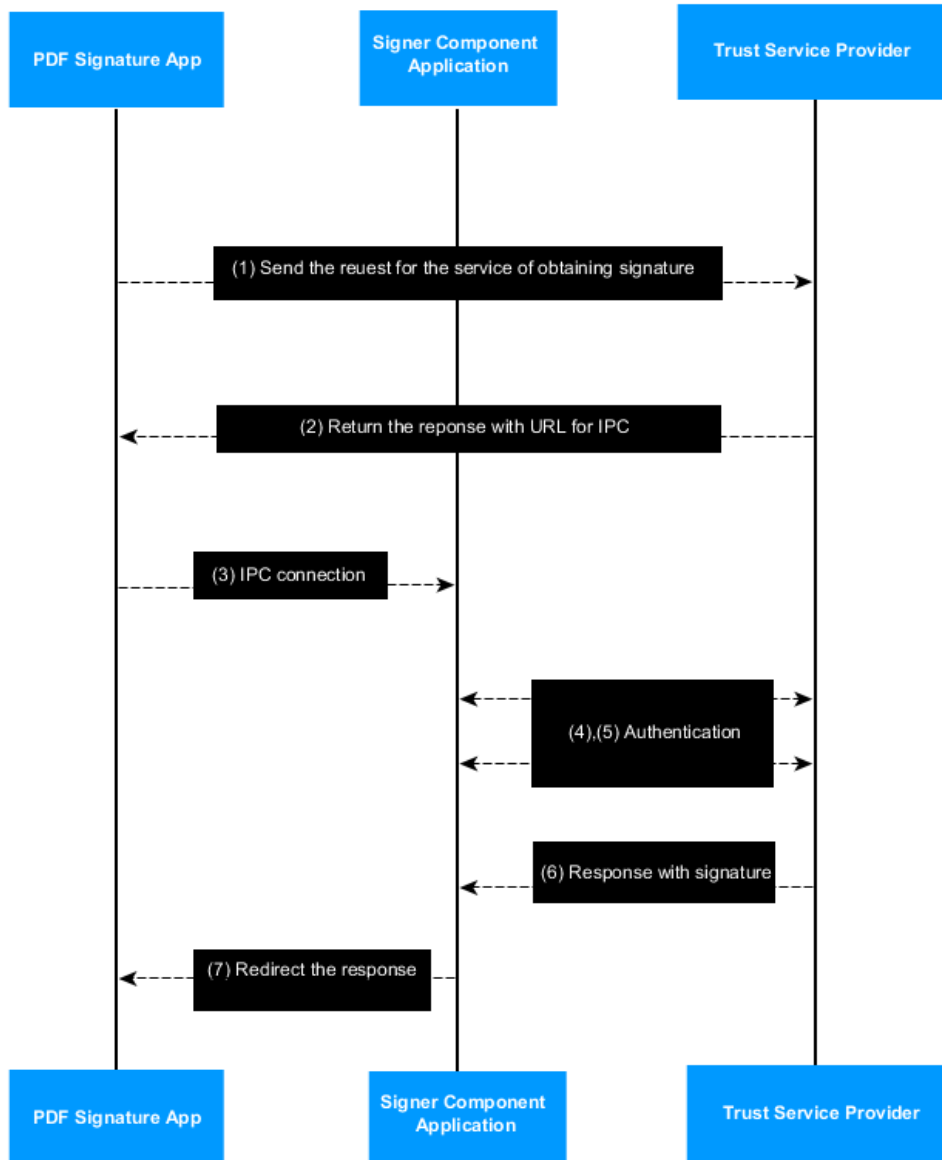


Figure 4.2.: Overview of the communication process.

4. Model

4.4. Summary of the section

In this section, a concept for providing a mobile solution of PDF signing has been proposed. Privacy, usability as well as security are the main objectives of the model; however, defining them also led to additional objectives, such as modularity and applicability. These requirements served as a basis for the design process.

As we aim to design and implement a usable model, service provider - in our case PDF Signature App, delegates the authentication process to the Signer Component Application, which performs the authentication with a Trust Service Provider with the aim to obtain the signature. The requested signature is retrieved from the Trust Service Provider after successful authentication. This way, the amount of interactions required by the user is minimized, as the user only uses one set of credentials - which consists of a password and a fingerprint.

Privacy is mainly accomplished through the local implementation of PDF processing. This implementation reduced the risk of relying on Third Party Services (e.g., a server-side hosted PDF-AS instance by the service provider) for PDF manipulation. This way we make sure that the confidentiality file is not jeopardized and as a result, the proposed system protects the user's privacy.

Security, as the third objective, is guaranteed by the used concept of the QES on mobile devices. The local authentication is done by the knowledge factor, that requires the user to enter phone number and password. The remote authentication is done with the hardware element on the phone using biometric authentication like a fingerprint. The overall process presents secure and reliable authentication assured by the multifactor authentication.

5. Implementation

This section describes the implementation of the proposed model, where multiple components collaborate in order to complete a multi-step process for obtaining a signed PDF file on a mobile device. To realize the previously described model, we have implemented a mobile Android application, namely PDF Signature App. This application represents the core of our solution and incorporates both the custom Android library responsible for PDF processing and a graphical user interface. In addition, for authentication purposes, we have modified the implementation of a prototype TSP App, that represents the client mobile Android application of the TSP, and TSP (server-side). Figure 5.1 depicts the high-level overview of the involved components and their connections.

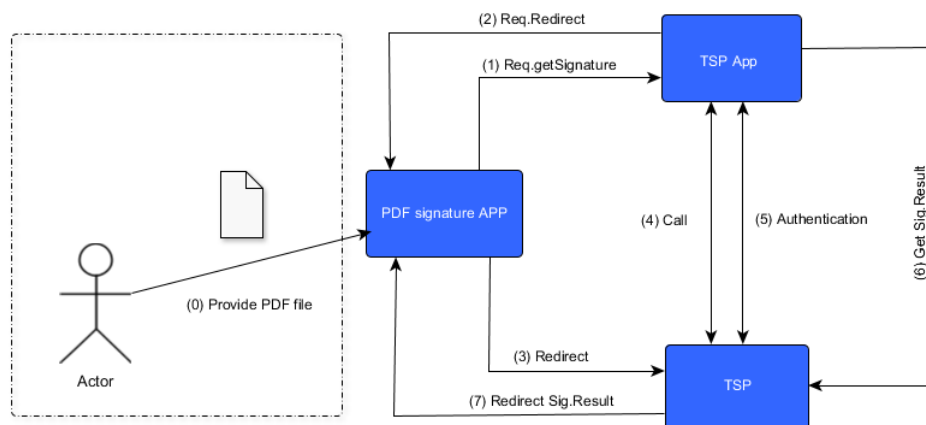


Figure 5.1.: High-level implementation overview.

5. Implementation

This section is organized as follows: Section 5.1 provides a description of the most important parts of PDF processing. The following section 5.2 explains the authentication process, which results in a remote signature needed to finalize the signing process. Lastly, this section is concluded in section 5.3.

5.1. PDF Processing

As we already mentioned, the local implementation of PDF processing, in comparison to the same remote web service, has a significant impact on fulfilling the requirements of our solution. As we aim to build a privacy-preserving solution, it required moving the entire implementation of PDF processing to the client-side. This way, we not only protect input data from sharing it with the remote web server, but the execution of the signing processes is more controllable in terms of reliability. With the client PDF processing implementation, we need to maintain only one application, and therefore, we do not depend on the availability of server-side processing. On the other hand, one could indicate that this carries an additional burden in terms of performance and speed of the process; however, we argue that this is a reasonable trade-off since this decision does not have a significant impact on the overall user signing experience.

One of the first steps in this phase is the collection and preparation of data: in other words, we create signing parameters. Signing parameters include a configuration file, a signature profile, a data source, and positioning parameters. The visible signature has its own visual representation on the file specified by these signing parameters. After the user has selected a PDF file to be signed, the App renders the file for showing its pages on the screen. This way, the PDF file has been rasterized so that a user would have a graphical presentation while navigating through the file and opting for the desired position. For the positioning purposes, a user dynamically moves a small signature block that acts as a placeholder for a signature block. The signature block is a visible part of the file whose detailed description will be provided in the further paragraphs. After the user has selected a position of the file, and all necessary information about signature appearance has been determined, we can proceed

5. Implementation

to the next phase.

In the second phase, we draw the signature block, which presents basically a table with parameters from the configuration. Furthermore, to add a digital signature on a PDF, we create a signature dictionary with corresponding fields based on the PAdES specification. The changes on the file have been incrementally updated. With this step, we terminated this phase, and we proceed to the authentication part. The incremental update operation is done when we obtain a signature value as a result of successful authentication. This will be fully explained in subsection 5.2. In the following, we will provide information about the main components of PDF processing. We start by introducing the PDF Library.

5.1.1. PDF Library

The heart of the PDF Signature application is a PDF processing library. The library implements all manipulation with PDF files, like drawing a table as a signature block and inserting a signature into a PDF file. It is an Android library written in Java programming language, which internally uses the Apache PDFBox library for signature insertion [8]. Apache PDFBox is an open-source Java framework for manipulation and processing PDF files. The framework supports adding new PDF documents, extraction, and manipulation of the existing ones. It also supports digital signatures, and it is published under the Apache License v2.0. In addition, we relied on the architectural concepts of the PDF-AS library, as a state-of-the-art solution for PDF signatures in desktop and web applications.

For better comprehension, we will outline several components of our library that describe certain behavior and incorporates certain roles.

Signature position. A signature position, on the implementational level, represents the coordinates of the file that indicate where the signature block will be allocated on the file. The signature position contains the three most important parameters. The x and y values are signature block coordinates on a specific page, where the x is the left-right coordinate, and y is top-bottom. Another value of the signature position is the w ,

5. Implementation

which tells us about the width of the signature block. This value is also configurable in the signature profiles. The last value we need is p , which indicates the page on which the signature block will be put.

Signature block. The signature block is implemented as a table. The table consists of the main and info part. Unless the signature is invisible, the table (signature block) is presented on the signed file. In the configuration file, we have defined the appearance of the table: this means that we predefined a font, paddings, margins, column width, height, borders, alignment. The outlook of the signature block is presented in figure 5.2.


	Signatory	Emina Ahmetovic
	Date/Time-UTC	2019-07-10T14:16:12+02:00
	Verification	Information about the verification of the electronic signature can be found at: https://www.signaturpruefung.gv.at
Note	This document is signed with a qualified electronic signature. According to Art. 25 para. 2 of the Regulation (EU) No 910/2014 of 23. July 2014 ("eIDAS-Regulation") it shall have the equivalent legal effect of a handwritten signature.	

Figure 5.2.: The appearance of the signature block.

Signature profile. A file called "config.properties" defines which identifiers will be visible in the signature block. The file also specifies the value of those identifiers. Some of them are predefined and related to one specific signature profile, while other values can be inserted by the user - like the signatory name. In other words, the signature profile gives us information about the content of the signature block, but it also contains information linked to the visual appearance of the block.

Since the signature block is, on an implementational level, a table with values, different signature profiles differ by a style, language, font, and data included. It is not necessary that all signature profiles have the same appearance. From the code snippet on listing 5.1, we see the signature block include data such as signatory name, date, logo, and a note.

5. Implementation

Listing 5.1: Example of the signature profile.

```
#default signature block width
sig_obj.SIGNATUREBLOCK_EN.pos=f:80;w:230

#table main settings
sig_obj.SIGNATUREBLOCK_EN.table.main.ColsWidth=1 4.5
sig_obj.SIGNATUREBLOCK_EN.table.main.Style.font=HELVETICA,5,
    ↪ BOLD
sig_obj.SIGNATUREBLOCK_EN.table.main.Style.valuefont=COURIER
    ↪ ,5,NORMAL

#table info settings
sig_obj.SIGNATUREBLOCK_EN.table.info.ColsWidth=1 2.7
sig_obj.SIGNATUREBLOCK_EN.table.info.1=SIG_SUBJECT-cv

#signature profile identifiers
sig_obj.SIGNATUREBLOCK_EN.key.SIG_SUBJECT=Signatory
sig_obj.SIGNATUREBLOCK_EN.key.SIG_DATE=Date/Time-UTC
sig_obj.SIGNATUREBLOCK_EN.key.SIG_NOTE=Note
sig_obj.SIGNATUREBLOCK_EN.key.SIG_META=Verification

#values of identifiers
sig_obj.SIGNATUREBLOCK_EN.value.SIG_META=
Information about the verification of the electronic
    ↪ signature can be found at: https://www.
    ↪ signaturpruefung.gv.at
sig_obj.SIGNATUREBLOCK_EN.value.SIG_NOTE=
This document is signed with a qualified electronic
    ↪ signature. According to Art. 25 para. 2 of the
    ↪ Regulation (EU) No 910/2014 of 23. July 2014 (\eIDAS-
    ↪ Regulation\")_it_shall_have_the_equivalent_legal_
    ↪ effect_of_a_handwritten_signature.
```

Listing 5.1 describes the predefined style for one signature profile. Also, the listing outlines the identifiers of the signature block. In our application, a user has an option to choose between two signature profiles,

5. Implementation

which differ by the language. However, the architecture makes it easy to adopt a new profile according to the special requests, with different size of the signature table or different identifiers.

After we have gathered all relevant information like the signature position, profile, configurations, and the uploaded file, we create our signing parameters, as seen on listing 5.2.

Listing 5.2: Signing parameters of the file.

```
signParameter.setOriginalDoc(originalDoc);  
signParameter.setSignaturePosition(STANDARD_POSITION_STRING);  
signParameter.setSignatureProfileId(PROFILE_ID);  
signParameter.setSettings(impl.getSettings());
```

5.1.2. PDF signatures

In this subsection, we will explain how digital signatures have been integrated into a PDF file, but first, we need to define the PDF signature.

We refer to the PDF signature as a binary data object based on the CMS or related syntax, which contains a digital signature placed inside the PDF file according to the ISO 32000-1 standard, with other signature information applied when it was first created [14].

Digital signatures in ISO 32000-1 support adding signatures directly to the document, where the signature with additional information is held in a PDF structure called the signature dictionary. The signature value is encoded as a binary object using a CMS or related signature format, depending on the profile. The digest is calculated over the entire file since the signature is included in the document. The signature dictionary is included in the calculation, but the PDF signature is excluded from the calculation. The information about the range of bytes for the calculated digest is contained in the special entry called *ByteRange*. This ensures that all the bytes of the file have been included in a digest, except for the signature value. The signature dictionary

5. Implementation

contains multiple entries, which will be elaborated later on. However, from the structure illustrated in figure 5.3 one of those entries is *Contents* - the entry that contains the signature. ISO 32000-1 also defines multiple forms of CMS signature integration into a PDF file, and these implementations have been defined by the entries in the signature dictionary called *Filter* and *SubFilter*.

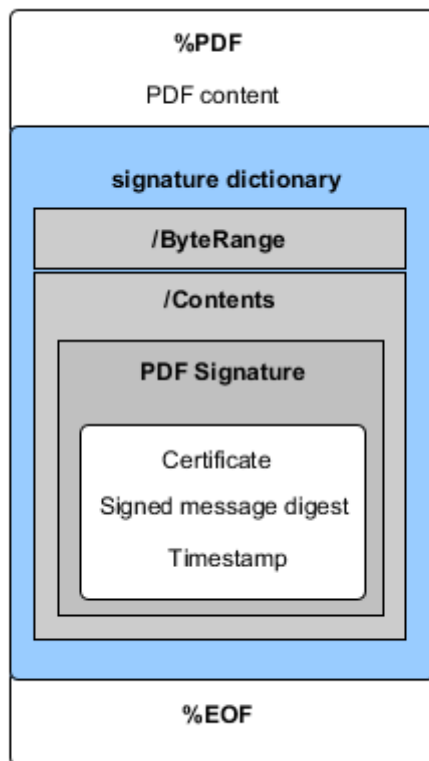


Figure 5.3.: Signature dictionary and the PDF signature.

PDF serial signatures are defined in ISO 32000-1. While other CMS signature profiles would support parallel signatures, the alternative PDF serial signatures define that each signature in PDF can contain a single signing certificate,

5. Implementation

but there can be many signature dictionaries, and each of them has its own byte-range values. This means that after the signer has signed the file once, a subsequent signer will also sign the previous signature [14]. Figure 5.4 illustrates the PDF serial signature.

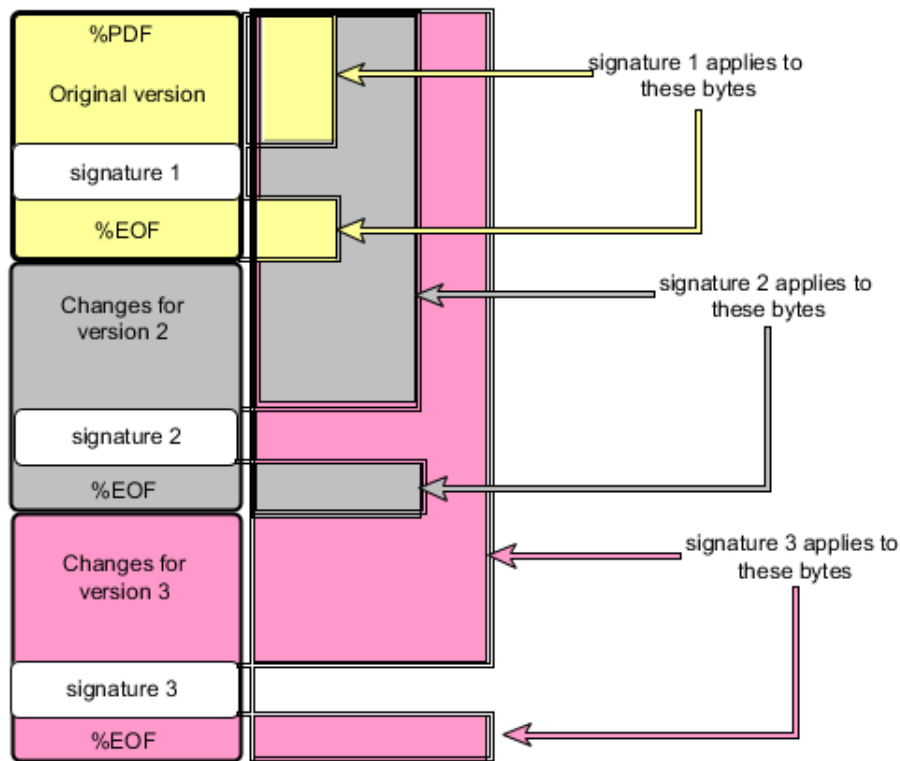


Figure 5.4.: PDF serial signatures.

In following, we will describe the most important features of PDF signature and relevant process steps for signature integration into the file.

Signature Appearance. Digitally signed PDF files with one signature handler can be verified with another. To achieve the interoperability between different signature handlers, a certain syntax for implementing signatures in a PDF file needs to be satisfied [4]. The standard syntax has two

5. Implementation

components:

1. **The signature dictionary.** The signature itself is stored under the signature dictionary. It contains parameters such as a name of the signer, time, signed hash of the file, and a certificate from the signer. The syntax for a signature dictionary is defined by the entry in the signature dictionary called *SubFilter*. Further description of the signature dictionary will be described in the following sections.
2. **The signature appearance.** The signature appearance refers to the way how the signature is represented to the user. The annotation field in a PDF file specifies the signature appearance. However, it is noteworthy to mention that the appearance of a signature in a file is not necessary for cryptographic purposes of verifying the signature but rather to enhance the user experience.

Document restrictions check. PDF signature cannot be applied if the document is marked as protected or has any of the accessibility restrictions; therefore, before the process of signing, we need to check permissions of the PDF file. The restricted file cannot proceed with the signature process.

Signature Dictionary. As we already said, the signature value, as well as all relevant information about the signature, is contained in a signature dictionary [11]. The signature dictionary contains entries that define the nature of the signature and its features. The signature value is a hex-encoded *PKCS#7*¹ object stored in the *Contents* entry. Alongside the signature value, the dictionary also contains *ByteRange* as an array of four 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 signature validator handlers and presents the internal name of a type of signature, such as *adbe.pkcs7.detached*.

Other entries can be *Reason*, a string which explains why the signatory is signing the file, as well as *Location*, *Name*, *Date*. In our application, we have defined, *filter*, *subfilter*, *date of signing* as well as *reason*. On

¹<https://tools.ietf.org/html/rfc2315>

5. Implementation

listing 5.3 and 5.4, we see how the signature dictionary is implemented and represented in PDF file.

Listing 5.3: The creation of the signature dictionary and its entries.

```
PDSignature signature = new PDSignature();
signature.setFilter(filter);
signature.setSubFilter(subfilter);
signature.setSignDate(Calendar.getInstance());
signature.setReason(signerReason);
```

Listing 5.4: The signature dictionary and its entries - shortened version.

```
20 0 obj
<<
/Type /Sig
/Filter /Adobe.PPKLite
/SubFilter /ETSI.CAdES.detached
/M (D:20190403142400+02'00')
/Reason (Signaturpruefung unter http://www.signaturpruefung.gv.at)
/Contents <308006092A864886F700000....000000000000>
/ByteRange [0 14587 22781 17992]
>>
endobj
```

Signature integration. As seen on listing 5.5, one of the last steps in PDF processing would be to add a created signature on a document and to save the altered document with an incremental update, so that further signatures could be applied. However, it is noteworthy to mention that the signature value should be retrieved from an authentication process before the process finishes.

5. Implementation

Listing 5.5: Signature integration and incremental update.

```
options.setPreferredSignatureSize(signatureSize);  
document.addSignature(signature, this, signatureOptions);  
document.saveIncremental(output);
```

5.1.3. Additional settings

We have implemented additional features as an addition to the basic functionality to assure user satisfaction. Therefore, a user has an option to choose whether the signature will be visible or invisible. Furthermore, the user can choose the language of the signature block, but she can also enter her name that will be visible on the signature block. Lastly, the user can opt to sign file in such a way that the file remains PDF/A or PDF/UA compliant. More details on this implementation will be provided in the following section.

Invisible Signature. A signature can be visible or invisible. In case of the visible signature, the signature block associated with the annotation is displayed on the file. On the other side, the invisible signature means that no signature block is presented on the file. However, an invisible signature is still associated with the annotation, which in this case has a rectangle array with zero values.

PDF/UA. The PDF/UA, or ISO 14289², is a PDF standard for the “Universal Accessibility”. The purpose of the PDF/UA is to define requirements for presenting documents in a form that is accessible to the persons who use assistive technology when reading digital content. The standard defines a reliable and accessible file, but at the same time hides technical details from the user.

PDF/UA standard is based on Tagged PDF³, meaning that we have a marked content, which follows a structural hierarchy. Every item of the content has its place in the hierarchy, but beside them, we have

²<https://www.iso.org/standard/54564.html>

³<https://www.iso.org/obp/ui/#iso:std:iso:32000:-1:ed-1:v1:en>

5. Implementation

“Artifact” - an item which does not belong to the vital document content. In addition, PDF/UA provides technical specifications as a complement to the WCAG 2.0⁴ conformance regarding the PDF content, by clarifying the low-level technical details for delivering accessible PDF content. [18] We will summarize some of the required conditions for the PDF/UA.

- The content of the document needs to be tagged in logical reading order.
- Fonts in the document need to be embedded, with exception to the fonts for invisible text.
- Some layer options are not allowed.
- Problematic content is not allowed.
- The document’s metadata needs to specify a title of the document.

PDF/UA specifications are mainly written for the software developers who are working with the PDF related software. [31].

However, PDF/UA compliant file has additional benefits and can be a powerful tool for providing navigation options, transforming the appearance of a text, improving search engine, and more.

PDF/A. PDF Archiving (PDF/A) stands alongside the many standards defined over the umbrella term “PDF” [2]. It is an ISO⁵ managed standard developed to assure the long-term preservation of documents.

This standard provides requirements that enable creation, view, and print of PDF documents, however, it does not specify any concrete strategy for archiving documents but rather a profile that enables reproduction of the same document in the future [1].

PDF/A can be seen as a guideline of the features that should be contained and the features that should be avoided for long-term archiving [21].

We will outline five main features of the PDF/A file.

1. **Self-contained.** Every feature of the file, which will be rendered or printed, like text, images, vector graphics, fonts, color information, needs to be included within the file, or in other words,

⁴<https://www.w3.org/TR/WCAG20/>

⁵International Organization for Standardization

5. Implementation

self-contained. This characteristics also prohibits some of the external content references, e.g., JavaScript, audio, and video content.

2. **Self-documenting.** Self-documenting is a characteristics of the PDF/A, which provides documentation about the file with the use of metadata. It suggests documenting identifiers like file provenance, font metadata, and file identifier.
3. **Device independent.** The device independence means that the processing of the file should be the same regardless of a device. For this purpose, it is suggested to use specific RGB or CMYK color profiles which are device-independent.
4. **Unfettered.** The PDF/A compliant file is encryption-free, which means that the file should not be protected but open and available for processing. This feature also defines that digital signatures are applied after the PDF/A content is created.
5. **Two levels of compliance.** We differ two levels of compliance - the core one PDF/A-1b, which refers to the basic requirements, and the higher level, PDF/A-1a. The PDF/A-1a requires *tags*, a special document structure which provides many features, where accessibility for the people with disabilities is among them.

5. Implementation

5.2. Authentication process

To generate a qualified electronic signature, a user first needs to authenticate herself. The user authentication and the signature acquisition require interaction between multiple components. This interaction is defined by the specifications, which is called the Security Layer 2.0. The SL2.0 presents the extension of the existing concept called Security Layer. These interaction steps are, on a high-level, depicted in figure 4.2.

In short, a user shares a PDF file with the PDF Signature App for processing. The App communicates with the Trust Service Provider component through the application interface to acquire the signature. Furthermore, the communication is transmitted to the TSP Client App, which interacts with both the user and TSP component via authentication interfaces. Lastly, the corresponding authentication result is delegated to the PDF Signature App.

5.2.1. Components

Three components are involved in the process. The first component is the PDF Signature App, the core application then follows the TSP component, and TSP App, which corresponds to the mobile client application from the Trust Service Provider. While PDF Signature App has been previously elaborated, it is noteworthy to mention the basic features of the remaining two applications. The TSP in our implementation is a prototype Java web service, which generates the qualified electronic signatures. The TSP App is client-side application from TSP which implements a graphical user interface for user authentication. TSP App is an Android application, which defines certain requirements, such as minimum API 23 and the support of alternative authentication methods such as a fingerprint.

5.2.2. Communication flow

The communication between components starts with the request, which requires the signature and ends with a signature value.

5. Implementation

(1) PDF Signature App to TSP. First step in the process is the interaction between PDF Signature App and the Trust Service Provider. PDF Signature App sends a request to the Trust Service Provider requesting a signature. The request contains a JSON object with corresponding parameters that are indicators for a required service.

Listing 5.6: The JSON body request from PDF Signature App to TSP.

```
1 {
2   "v": 10,
3   "reqID": "e7e1137d-f206-431e-a64e-60612d4b8a9c",
4   "payload": {
5     "name": "createCAAdES",
6     "params": {
7       "keyId": "SecureSignatureKeypair",
8       "content": "JVBERi0xLjQNJU9GCg==",
9       "contentMode": "detached",
10      "mimeType": "application/pdf",
11      "padesComatibility": true,
12      "excludedByteRange": [
13        [
14          14579,
15          22772
16        ]
17      ],
18      "cadesLevel": "cAdES"
19    }
20  }
21 }
```

The JSON body consists of the parameters defined by the Security Layer 2.0. Although we will not explain the specification in detail, tables below will describe the parameters. Table 5.1 depicts the main parameters of the response, while table 5.2 describes the *payload* body and table 5.3 describes the *params* body of the request.

5. Implementation

Name	Request	Response	Description
v	M	M	Describes the version protocol
reqID	M		Describes the unique request id
respID		M	Describes the unique response id
inResponseTo		M	Describes the unique ID of the request for the particular response ID
transactionID	O	O	Describes the unique ID of the transaction
payload	O	O	Unsigned payload from the communication interface
signedPayload	O	O	Signed payload BASE64 encoded

Table 5.1.: JSON parameters of the first request. "M" is mandatory field, while the "O" marks the optional fields in the request.

Name	Optional	Description
name		Name of the command. In our case the command is "createCades"
params		This is a JSON object which contains parameters associated with the name. Parameters that can be contained will be provided in one of the following tables.

Table 5.2.: Parameters of the *payload* field.

5. Implementation

Name	Optional	Description
keyID		This parameter is the ID of the signature key required for the signature creation.
content	X	Content present BASE64 encoded data to be signed.
contentType		Presents the mime type of the data to be signed
padesCompatibility	X	The value of this parameter can be either true or false. If the resulting signature is PAdES compatible, then we set this parameter to true.
excludedByteRange	X	This parameter describes the range of bytes to be excluded from the signature
caesLevel	X	This parameter refers to the different version of CADES standard.

Table 5.3.: Parameters of the *params* field.

(2) TSP to PDF Signature App. After the initial signature request, TSP sends a response to the PDF Signature App. The response, among other parameters, also contains a JSON object with the URL to the TSP App. The table 5.4 and 5.5 provide the description of the parameters contained the response.

5. Implementation

Listing 5.7: The JSON request from TSP to PDF Signature App.

```
1 {
2   "v":10,
3   "reqID":"03b2828b..371b",
4   "transactionID":"aacd815..376de8",
5   "payload":{
6     "name":"redirect",
7     "params":{
8       "command":{
9         "name":"redirect",
10        "params":{
11          "command":{
12            "name":"call",
13            "params":{
14              "url":"https://demo.egiz.gv.at/
15                vda/auth/initial",
16              "method":"get",
17              "includeTransactionID":true,
18              "reqParams":[
19                {
20                  "pendingId":"Ytd..
21                    jMtNjY5Zjk2MjU2ZjYx"
22                }
23              ]
24            }
25          },
26          "url":"APICALL://call.vda.in.app",
27          "IPCRedirect":true
28        }
29      },
30      "url":"https://demo.egiz.gv.at/vda/
31        generic/redirect",
32      "IPCRedirect":false
33    }
34  }
```

5. Implementation

Name	Optional	Description
command	X	The command to be transferred to the defined URL.
signedCommand	X	The signed command to be transferred to the defined URL
url	X	The URL where the request is addressed
IPC redirect	X	The true or false value. If true, it defines the interprocess communication between two apps.

Table 5.4.: The parameters contained in the response.

Name	Optional	Description
url		The URL where the request is addressed
method	X	The parameter that defines the HTTP request.
includeTransactionID		True or false value parameter. If set to true, the body has to contain transactionID.
reqParams	X	Describes the unique request identifier

Table 5.5.: The parameters of *params* field.

5. Implementation

(3) PDF Signature App to TSP App. PDF Signature App uses the forwarded URL to establish the IPC communication with TSP App. For this purpose, PDF Signature App sends a request to the TSP App and contains a JSON object with the following structure.

Listing 5.8: JSON request from PDF Signature App to TSP App.

```
1 {
2   "v": 10,
3   "reqID": "03b2828b-48c1-4e14-b87e-b78e8677371b",
4   "transactionID": "aacd815a-e61f-45e9-8a4f-c28796376de8",
5   "payload": {
6     "name": "call",
7     "params": {
8       "url": "https://demo.egiz.gv.at/vda/auth/initial",
9       "method": "get",
10      "includeTransactionID": true,
11      "reqParams": [
12        {
13          "pendingId": "YTdiM2E2...Yx"
14        }
15      ]
16    }
17  }
18 }
```

(4) and (5) PDF Signature App to TSP App. The steps 4 and 5 relate to the communication between TSP App and TSP. The authentication process is delegated to the TSP App, which provides a graphical interface for the user to authenticate. The user needs to provide her credentials in order to authenticate at the remote HSM. The multifactor authentication is satisfied when the user enters the password as the knowledge factor and applies fingerprint (an access to the mobile device hardware security

5. Implementation

element) as the possession factor.

(6) TSP to TSP App. After the user entered the credentials, TSP sends a response to the TSP App, which contains the signature value if authentication is performed successfully. The signature parameter in the response presents a CADES, BASE64 encoded signature value.

(7) TSP App to PDF Signature App. As a last step in the communication, the TSP App forwards the response from TSP to the PDF Signature App. The PDF Signature App handles the response and uses the signature value to finish signing the document. The response is shown on listing below.

Listing 5.9: JSON request from TSP App to PDF Signature App.

```
1 {
2   "v": 10,
3   "respID": "1ccc8d84-2dcf-4ded-a039-8
4     beb41ccdb19",
5   "inResponseTo": "d9312f16-b057-4f98-8dc7-
6     e9752d2f71d4",
7   "transactionID": "c01480ee-e85b-4d32-82c4-86
8     b45d428522",
9   "payload": {
10    "name": "createCADES",
11    "result": {
12      "signature": "MIAGCSqGSIB3DQ...AAAA"
```


5. Implementation

5.3. Summary of the chapter

In this section, we have described the implementation of our model for mobile PDF signing. The implemented solution allows any users from anywhere to select an arbitrary PDF file and sign it using just a single mobile device.

The implementation of our solution consists of three components that interact with each other to process PDF files but also to authenticate the user. The PDF processing is done inside the PDF Signature Application, which presents a core of our implementation. Additional used and modified implementation is from TSP, which presents a Trust Service Provider and TSP App, which is an Android App that represents as a client side of the Trust Service Provider.

As we aim to fulfill our objectives and build a user-friendly solution, the application for PDF processing provides an interface for signature positioning on the file. Furthermore, additional rich features have been added to the implementation. Those features are configurable by users and allow them to specify the language of the signature block and also the conformance with PDF/A and PDF/UA. The implementation of PDF processing is done inside the PDF Signature App as we aim to create a privacy-preserving solution and avoid relying on remote web services. The security of our solution is guaranteed by the multifactor authentication concept. The use of alternative methods for authentication, such as fingerprint, significantly impacts the user experience and makes the signing process faster and easier. The overall user involvement is decreased to a minimum number of interactions.

6. Demonstrator

In this section, we will illustrate an application flow from a user perspective. The prerequisites for the user demonstration is to have a mobile device that supports fingerprint authentication. Two Android applications need to be installed. First one is the PDF Signature App, and the other one TSP App. Figures present screenshots taken from the different stages of PDF signing. In the beginning, the user opens a PDF file with the PDF Signature App via Share Intent and obtains the following view.

6. Demonstrator

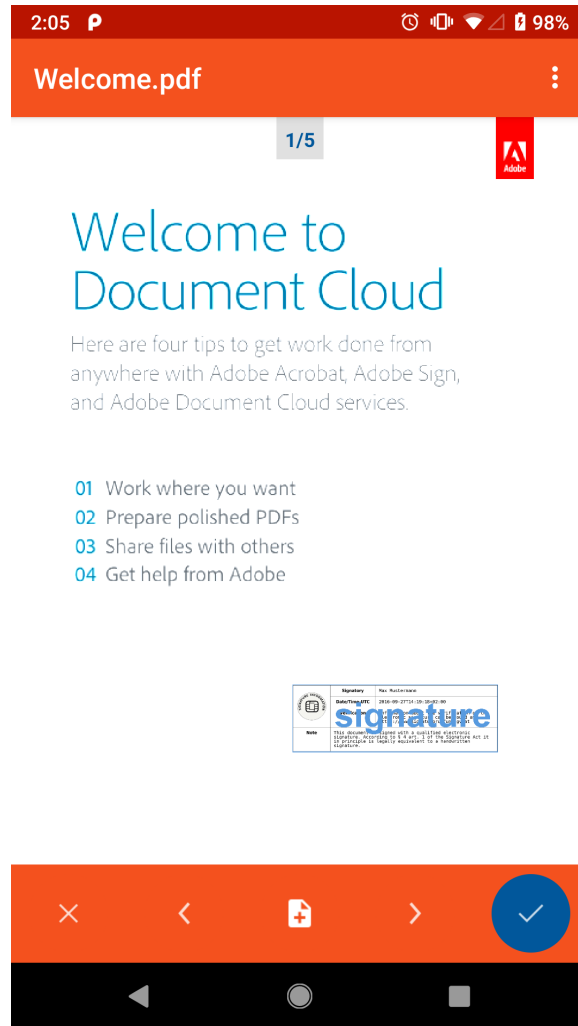


Figure 6.1.: The PDF file is open with the PDF signature application, and the user is presented with the graphically rendered file.

The interface consists of three layers. The first layer is the top layer implemented as the App Bar, which contains a name of the file as well as the menu icon. The menu will be explained later. The middle layer presents the bitmap of the PDF file with a small indicator in the top central position, which shows the page. This is the biggest part of the

6. Demonstrator

interface and has a dynamically draggable icon - signature block. The signature block can be dragged through the entire file until the user decides on the signature position. The latest layer is a bottom navigation bar. This bar consists of five buttons. The first button from the left side is "x" - for canceling the activity. The button in the middle is for creating an additional page so that the signature block could be on the entirely new and blank page. The first button from the right indicates that the user has finished the positioning part and the signing can proceed. The remaining two buttons are for navigating through the file.

The menu button will navigate the user to the "Settings" Activity. In this part, the user can choose between different options for the file, like the language of the signature block, whether the file should be compliant with PDF/A, or should the signature be invisible.

Another important option here is that the user can enter its name as the name of the signatory. The name entered in this field will be visible on the file and will correspond to the name of the signatory on the signature block. The decision for the approach, in which a user can enter its name in the signature block as a signatory name, was made for usability purposes. The standard procedure for acquiring the name of a signatory is by extracting it from an obtained certificate. However, to obtain the certificate would mean that the authentication process would take be repeated twice, first time for gaining the signatory name from the certificate, and a second time for obtaining the signature value. This case is related to the concrete communication between Trust Service Provider and PDF Signature App, and from technical reasons, the lack of session management does not allow this that two requests are processed through one transaction instead of two. Two times repeated authentication process, means two times the user is required to enter the set of credentials, which makes the use of application exhaustive and prolongs execution time.

6. Demonstrator

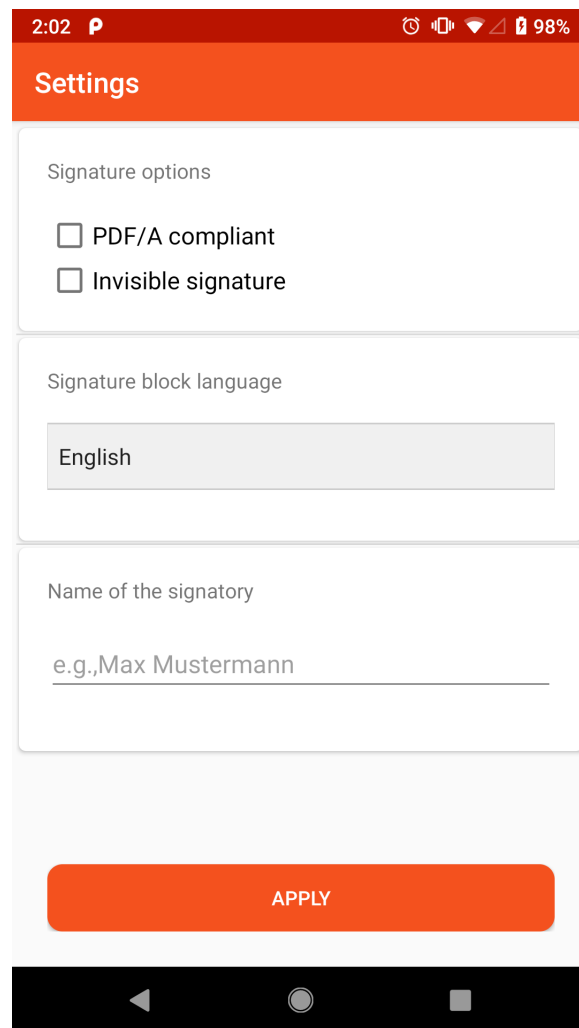


Figure 6.2.: Settings a user can select when signing a PDF file.

After the user chooses the preferred options, she starts the signing process. Users interaction is required again when the PDF Signature App established an IPC connection to the TSP App. As we mentioned, the TSP App opens and carries out the authentication of a user by providing an authentication interface as visible on figure 6.3.

6. Demonstrator

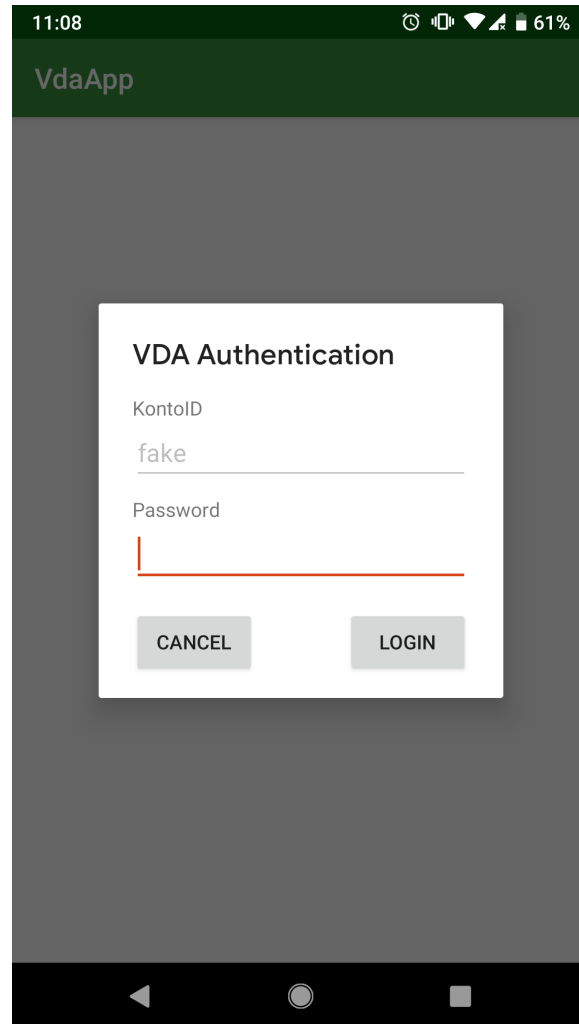
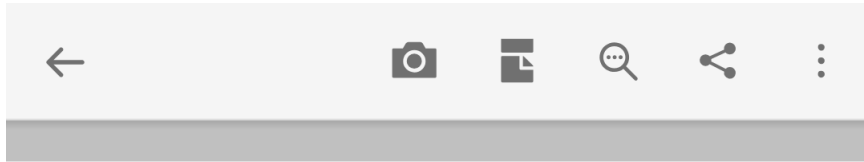



Figure 6.3.: Authentication interface provided by the TSP App.

After successful authentication, the file has been signed and saved. The preview of the signed file is shown on figure 6.4.

6. Demonstrator



This page is intentionally blank.

	Unterzeichner	Sanna Mauerer
	Datum/Zeit-UTC	2019-04-03T14:24:00+02:00
	Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://www.signaturprüfung.gv.at
Hinweis	Dieses mit einer qualifizierten elektronischen Signatur versicherte Dokument hat gemäß Art. 26 Abs. 2 der Verordnung (EU) Nr. 910/2014 von der EU die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.	

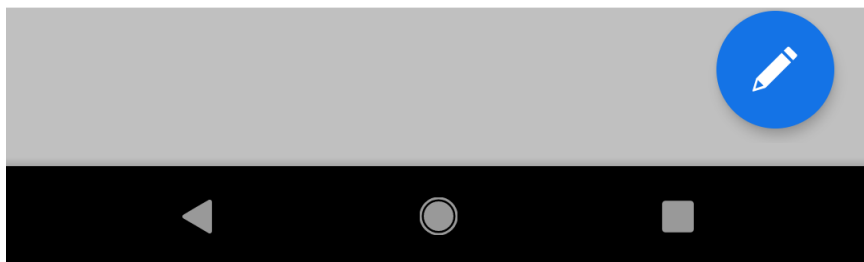


Figure 6.4.: The signed file opened with a PDF viewer.

7. Discussion

The importance of signing PDF documents is widely recognized in both the private and public sector. PDF files are commonly used for the exchange of electronic documents, while applied PDF signatures provide the integrity, authenticity, and non-repudiation of electronic documents. For this reason, there are many applications that offer signing PDF files. However, they are mostly desktop or web applications. As we move towards the mobile future, there is a need to provide a solution for obtaining PDF signatures on mobile devices as well. Nevertheless, designing such a solution has not been in the focus of research in the past, due to many reasons. One of them is the fact that it was not easy to enable multifactor authentication on mobile devices without using additional hardware. Nevertheless, the aim of this thesis is to bridge this significant gap in mobile technology and provide a reliable solution for enabling PDF signing on mobile devices, on the way that satisfies privacy, security and usability requirements.

During our work, we have designed a model for signing PDF files on mobile devices, where our concrete implementation has shown the practical applicability and feasibility of our model. However, it is also important to discuss how well and to what extent the challenges and the goals have been addressed and reached. Therefore, in this section, we will reflect on the requirements and discuss how well they perform.

This chapter is organized as follows: First in section 7.1, we provide a description of our evaluation where we deploy the productive e-Government components, then in section 7.2 we discuss how well the requirements of our model have been satisfied, and lastly, in section 7.3 we provide a conclusion of the chapter.

7. Discussion

7.1. Integration with Austrian productive solution

The proof-of-concept implementation of our solution, which has been explained in section Implementation, has shown the feasibility of our theoretical model. However, it is also required to evaluate the behavior of our solution in a real-world scenario. For this purpose, we have practically evaluated our solution by integrating it with Austrian productive solution. The created semi-productive environment has enabled us to further test the application.

Our implementation, as we already mentioned, consists of the three parts: PDF Signature App, prototype Trust Service Provider, and prototype TSP client mobile application. However, in our semi-productive environment, the components such as TSP and TSP app are replaced with already existing e-Government components.

Although we are still in a very early stage of the first pilot period, the results achieved so far demonstrate that our solution can be easily integrated into existing infrastructure. This is an important result, as one of the objectives of our model is applicability and modularity of our solution.

7.2. Evaluation of requirements

The represented requirements of our model in section 4 are quite general and broad; therefore, in this section, we will provide a bit narrowed definition and evaluate the concrete aspects of each of them.

7.2.1. Security

Security, as an important requirement of our model, can be derived into the following objectives:

Security on the application level. The security on application level refers to the steps and decisions, which have been taken while implementing the solution, to improve the security of an application. This aspect will not be thoroughly examined, as we assume that soft-

7. Discussion

ware components were implemented and installed correctly, taking care of the potential security vulnerabilities.

Protocol security. Protocol security refers to the correctness of the Security Layer 2.0 implementation. We argue that interfaces and communication between components of our system have been implemented according to the specifications. The integration of our solution with concrete instances of Austrian solution has shown that the communication based on the SL2.0 has been satisfied.

Security on authentication level. Although the authentication is a big part of our solution, as it is obligatory for obtaining QES, the security on authentication level is not in the scope of this topic.

PDF signature security. The security of PDF signatures became a big topic, since it has shown that little research has been done regarding the security of digital signatures embedded in PDF files in the prior work. The recent research in this domain introduced three most prominent attack classes regarding the PDF signatures- Signature Exclusion, Incremental Update Abuse, and Signature Wrapping [25]. The common feature of these attacks is the fact that they aim to manipulate PDF files in such a way that alterations after signatures are not detectable by the signature validation. Although the signature validation is not a part of the thesis, it is clear that the security of PDF signatures must be imperative in PDF processing applications. In this thesis, we have taken certain steps to harden the potential attacks on PDF signatures. One of these practices is already defined by the PAdES specifications and refers to including the entire document into the byte range calculation.

7.2.2. Privacy

In an environment such as e-Government, privacy is among major concerns. From the need to build a privacy-preserving solution, we derive additional objectives.

Data minimalization. The data minimalization refers to minimum data

7. Discussion

disclosure to the Service Provider to protect the privacy of the user. The local implementation of PDF processing aims to satisfy this objective. The transition of the implementation from the server-side to the client-side is made with the purpose of preserving users data by not sharing it with additional service providers.

However, if we evaluate the privacy features like privacy in terms of users data, tracking data or linking data in communication with service provider, we would agree that the objectives have not been satisfied since sharing the users data like PDF file with Trust Service Provider presents a normal behavior of our application and without this step it would not be possible to obtain QES. The file is sent to the Trust Service Provider with the aim that a user has a preview of the file she is signing. Nevertheless, even though in our concrete implementation, sharing file with TSP is obligatory, this certainly leaves a room for improvement, which will be discussed in the next chapter.

7.2.3. Usability

The general acceptance among users depends on the fact of how simple the system is to use. This usability requirement leads to additional objectives.

Signature positioning. The signing processes involves a user choosing the position for the signature. As this feature has a major role in overall user experience and it is an inevitable step in signing, the positioning of the signature block is made in a user-friendly way where the user can navigate through the file and easily select the position.

Minimal user interaction. A user aims to finish the signing process as easy as possible. This means that the time of signing should be minimized, as well as the user's interaction. The decision not to obtain the user's certificate contributes to fulfilling this objective, as we prevent of repeatedly entering the same credentials and performing the same process steps. Although we cannot influence the means for authentication in general, we argue that the involving biometric

7. Discussion

form of authentication contributes to the user-friendliness of the process. On the other hand, it is reasonable to conclude that the client-side PDF manipulation would extend the execution time, in comparison to the same server-side computations.

Additional features. Additional features have also been implemented to enhance users satisfaction. These feature mostly define the appearance of the signature block; however, they can also be extended to support additional requests. So far, the user has an opportunity to choose a language of the signature block text. She, also, can specify whether the signature will be visible or invisible. Furthermore, the user can also define that file conforms with PDF/A. Additional feature-rich options will be discussed in section 8.

7.3. Summary of the chapter

To summarize, we tackle different aspects of our objectives on different abstraction levels, as they have a wide scope of possibly derived sub-objectives. Therefore, the main aspects of security were the correctness of implementation together with satisfying SL 2.0 protocol and complying with PAdES standard to harden the potential attacks related to PDF signatures. Furthermore, preserving the privacy of data has been tackled by the local implementation of PDF processing. The client-side implementation for PDF processing serves to minimize data disclosure. The usability objective, or its derivatives, has been tackled by different optional features that our application offers, but most importantly, by embedding an easy mean of authentication on a single mobile device. However, we also indicate the possibility of our solution to be improved and further enhanced. In the following section, we will elaborate on the possible improvements.

8. Future work

As we already mentioned, further research on the matter of PDF signatures on mobile devices is necessary to yield improvements in this direction. The following ideas and approaches are based on the knowledge and experiences gathered during the design and implementation and can help to enhance the objectives.

Full privacy-preserving feature. The privacy feature of our solution has been discussed in many sections, and as it has been explained, we focused on this as an emerging objective for our model, however, preserving the privacy of our solution could be further enhanced. We propose that future work could go in the direction of taking the necessary steps to make our solution fully privacy-preserving, meaning that the data are not disclosed to any other components in the process. The current solution is based on the communication between PDF Signature App and Trust Service Provider, which requires, according to the protocol, that the file to be signed together with calculated byte ranges is sent to the server.

We propose that we do not send the entire document to the server-side, but only the hash value of the document. With this approach, not only we would have a fully privacy-preserving solution, but it would reduce certain computational steps at the server-side. The drawback of this approach is the fact that a user would not have an option to preview the document, which now exists. In that case, additional measurements need to be taken into account to ensure the trust between two application, as a guarantee that the correct file is to be signed on the server-side.

Signature verification. As our application checks the correctness of the signature only on integration level, a proper signature verification tool incorporated into PDF signing application would be a benefi-

8. Future work

cial function to have in the environment which handles the signed PDF document. Numerous desktop application for preview PDF documents offer these features; however, the situation is different on mobile devices.

The verification tool would provide users the ability to verify all signed PDF documents. The awareness about verifying documents, in this case, would be increased, as this topic on mobile devices is not quite in focus and does not provide a usable option. Integrating the signature validation tools can be done in two ways: First one is server-side validation, which means that we use an existing online web service for validation signatures. The other approach would be a privacy-preserving approach, in which we would implement the user validation on the client-side.

In addition to these main two targets for future work, we also suggest that the part of the future work should contain a proper and detailed usability evaluation. The usability evaluation would show how good and what else can be done regarding the speed of the process and graphical user interface.

In conclusion, this thesis presents a contribution towards building reliable m-Government by introducing a solution for generating qualified electronic signatures for signing PDF files on a single mobile device. Nevertheless, we find an opportunity to enhance the presented solution by conducting further research in the above-mentioned directions as well as other fields of study.

9. Conclusion

The chapter summarizes and concludes the content of this thesis. In this thesis, we have proposed and implemented a solution for signing PDF files on a single mobile device using remote authentication against HSM for generating QES. Our solution tackles not only the privacy challenges from related work but also provides a usable solution as an alternative to security-critical use cases for qualified signing PDF files in the mobile domain.

During our work, one of the first steps was to design a model for our solution. The model is built on well-established requirements, and in this thesis, we tackle concrete segments of security, usability, and privacy. Privacy is the first objective of our model. Building the privacy-preserving solution required moving the implementation to the client-side and implementing PDF processing inside the mobile client application. This way, we do not rely on the remote web services for PDF processing, and users data have minimal disclosure. The decision to move the implementation on the client side has also contributed to creating reliability in our solution, as we maintain only one application.

Furthermore, to satisfy an emerging usability objective, the application provides an easy and intuitive graphical user interface. A user can navigate through the file and position the signature block on the desired place. The user interaction is minimal, as the user aims to finish the signing process as fast as possible. The user is also provided with various options regarding the signature appearance.

Security, as an additional objective of the model, is also satisfied with the novel signing concept which incorporates the remote authentication at HSM for creating qualified electronic signatures. The novel contribution in mobile industry such as TTE and its implementation with secure

9. Conclusion

element allowed us to utilize the multi-factor authentication at the single mobile device, making the entire process usable by engaging alternative authentication mechanism such as a fingerprint.

As an addition to the mentioned requirements, the model we propose is designed on the way that applies to all mobile platforms, and modular so that it can be easily integrated into other solutions.

The implementation of our solution consists of three parts. First and the main part is the PDF Signature App, an Android application which implements PDF processing and authentication. Other components, whose implementation we have modified for the signing purpose, are demo TSP and demo TSP App. The demo TSP App provides an interface for the user's authentication.

The proof of concept implementation of the mobile PDF signature solution has shown the feasibility of the proposed model; however, we aim to evaluate the capabilities of our solution in a real-world scenario. For this purpose, we have deployed the semi-productive components of the Austrian e-Government infrastructure instead of our prototypes into our solution and created a semi-productive environment for a mobile PDF signature service, where we have shown by easily integrating our solution, the feasibility in a practical environment.

In summary, this thesis improved the state-of-the-art of the mobile transaction-based services, by recognizing the need, and proposing, designing and implementing a solution for obtaining qualified PDF signatures on mobile devices. Our implementation serves as a contribution to building flexible m-Government infrastructure and bridges the gap between the traditional and mobile approach in terms of PDF signatures.

Appendix

Appendix A.

A signed file

The listing below represents raw PDF data after the file has been signed. It demonstrates the creation of the qualified PDF signature. Specific parts of the PDF file have been intentionally outlined for a better overview of the file.

Listing A.1: Signed PDF file - source code shortened.

```
1 %PDF-1.4
2 %
3 6 0 obj<</H[516 141]/Linearized 1/E 4534/L 8357
   /N 1/O 9/T 8191>>
4 endobj
5
6 xref
7 6 11
8 0000004458 00000 n
9
10
11
12 trailer
13 <</Size 17/Prev 8181/Root 7 0 R/Info 5 0 R/ID[<
   db7775cce227f6b30c440df4221dc390><b0b3638dea
   568846897460db50f305e8>]>>
14 startxref
15 0
16 %%EOF
17
```

Appendix A. A signed file

```
18 8 0 obj<</Length 64/Filter/FlateDecode/L 75/S 3
    8>>stream
19
20 endstream
21 endobj
22 endstream
23 endobj
24 15 0 obj<</Type/FontDescriptor/FontBBox[-665 -3
    25 2028 1037]/FontName/Arial/Flags 32/StemV
    88/CapHeight 718/Ascent 905/Descent -211/
    ItalicAngle 0/FontFamily(Arial)/FontStretch/
    Normal/FontWeight 400>>
25 endobj
26 16 0 obj<</Type/ExtGState/SA false/OP false/SM
    0.02/op false/OPM 1>>
27 endobj
28 1 0 obj<</Nums[0 2 0 R]>>
29 endobj
30 2 0 obj<</S/D>>
31 endobj
32 3 0 obj<</Count 1/Kids[9 0 R]/Type/Pages>>
33 endobj
34 4 0 obj<</Length 3261/Type/Metadata/Subtype/XML
    >>stream
35 <?xpacket beom/pdf/1.3/' pdf:Producer='Acrobat
    Distiller 6.0 (Windows)''></rdf:Description>
36 ...
37 endstream
38 endobj
39 5 0 obj<</ModDate(D:20060306150633-05'00')/
    CreationDate(D:20060306150633-05'00')/Title(
    Microsoft Word - Document2)/Creator(AdobePS5
    .dll Version 5.2.2)/Producer(Acrobat
    Distiller 6.0 \ (Windows\))>>
40 endobj
41 xref
42 0 6
```

Appendix A. A signed file

```
43 0000000000 65535 f
44 0000004534 00000 n
45 0000004567 00000 n
46 0000004590 00000 n
47 0000004640 00000 n
48 0000007977 00000 n
49 trailer
50 <</Size 6>>
51 startxref
52 116
53 %%EOF
54 5 0 obj
55 <<
56 /ModDate (D:20060306151233-05'00')
57 /CreationDate (D:20060306150633-05'00')
58 /Title (Blank PDF Document)
59 /Creator (AdobePS5.dll Version 5.2.2)
60 /Producer (Acrobat Distiller 6.0 \ (Windows\))
61 /Author (Department of Justice \ (Executive
    Office of Immigration Review\))
62 >>
63 endobj
64 7 0 obj
65 <<
66 /Pages 3 0 R
67 /Type /Catalog
68 /PageLabels 1 0 R
69 /Metadata 17 0 R
70 >>
71 endobj
72 xref
73 0 1
74 0000000000 65535 f
75 5 1
76 0000008357 00000 n
77 7 1
78 0000008642 00000 n
```

Appendix A. A signed file

```
79 17 1
80 0000008732 00000 n
81 trailer
82 <<
83 /Size 18
84 /Info 5 0 R
85 /Root 7 0 R
86 /Prev 116
87 /ID[<db7775cce227f6b30c440df4221dc390><4a2b68
    cfc7a1eff56fed75af5d28bdac>]
88 >>
89 startxref
90 10850
91 %%EOF
92
93 7 0 obj
94 <<
95 /Pages 3 0 R
96 /Type /Catalog
97 /PageLabels 1 0 R
98 /Metadata 17 0 R
99 /AcroForm <<
100 /Fields [18 0 R]
101 /SigFlags 3
102 /DR <<
103 /XObject <<
104 /FRM1 19 0 R
105 >>
106 0 R
107 >>
108 /TU <48696E776569730...D656E742E0A>
109 >>
110 endobj
111 19 0 obj
112 <<
113 /Length 2171
114 /Type /XObject
```

Appendix A. A signed file

```
115 /Subtype /Form
116 /Resources <<
117 /XObject <<
118 /Im1 22 0 R
119 >>
120 /Font 23 0 R
121 /ProcSet [/PDF /Text /ImageC /ImageB /ImageI]
122 >>
123 /BBox [231.99998 84.0 0.0 0.0]
124 /FormType 1
125 >>
126 stream
127 1
128 endstream
129 endobj
130
131
132 20 0 obj
133 <<
134 /Type /Sig
135 /Filter /Adobe.PPKLite
136 /SubFilter /ETSI.CAdES.detached
137 /M (D:20190403142400+02'00')
138 /Reason (Signaturpruefung unter http://www.
          signaturpruefung.gv.at)
139 /Contents <308006...0000000000000000000000>
140 /ByteRange [0 14587 22781 17992]
141 >>
142 endobj
143
144
145 9 0 obj
146 <<
147 /Contents 12 0 R
148 /Type /Page
149 /Parent 3 0 R
150 /Rotate 0
```

Appendix A. A signed file

```
151 /MediaBox [0 0 612 792]
152 /CropBox [0 0 612 792]
153 /Resources 10 0 R
154 /Annots [18 0 R]
155 >>
156 ...
157 /Height 210
158 /ColorSpace /DeviceRGB
159 >>
160 stream
161 endstream
162 endobj
163 23 0 obj
164 <<
165 /F1 24 0 R
166 /F2 25 0 R
167 >>
168 endobj
169 24 0 obj
170 <<
171 /Subtype /Type1
172 /BaseFont /Helvetica-Bold
173 /Encoding /WinAnsiEncoding
174 >>
175 endobj
176 25 0 obj
177 <<
178 /Subtype /Type1
179 /BaseFont /Courier
180 /Encoding /WinAnsiEncoding
181 >>
182 endobj
183 xref
184 0199 00000 n
185 0000040290 00000 n
186 trailer
187 <<
```

Appendix A. A signed file

```
188 /Size 26
189 /Prev 10850
190 /Root 7 0 R
191 /Info 5 0 R
192 /ID [<DB7775CCE227F6B30C440DF4221DC390> <221E05
      06FD34FBB04D8ADEC354F61099>]
193 >>
194 startxref
195 40374
196 %%EOF
```


Bibliography

- [1] *3D PDF Consortium - PDF Standards Overview*. URL: <http://3dpdfconsortium.org/pdf-standards/>. Accessed: 2019-04-08 (cit. on p. 52).
- [2] *3D PDF Consortium - PDF/A*. URL: <http://3dpdfconsortium.org/pdf-a/>. Accessed: 2019-04-08 (cit. on p. 52).
- [3] *ADOBE FAST FACTS*. URL: <https://www.adobe.com/about-adobe/fast-facts.html>. Accessed: 2019-03-19 (cit. on p. 1).
- [4] *Adobe Systems Incorporated. Digital Signature Appearances*. URL: <https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/PPKAppearances.pdf>. Accessed: 2019-06-8 (cit. on p. 48).
- [5] *Adobe Systems Incorporated. Digital Signatures in Acrobat*. URL: https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/digisig_in_acrobat.pdf. Accessed: 2019-03-19 (cit. on pp. 17, 19).
- [6] *Adobe Systems Incorporated. PDF Reference — Document management — Portable document format — Part 1: PDF 1.7*. URL: https://www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/PDF32000_2008.pdf. Accessed: 2019-03-19 (cit. on p. 12).
- [7] *Android Security Documentation*. URL: <https://source.android.com/security/keystore>. Accessed: 2019-03-19 (cit. on pp. 2, 29).
- [8] *Apache PDFBox® - A Java PDF Library*. URL: <https://pdfbox.apache.org>. Accessed: 2019-04-08 (cit. on p. 43).
- [9] *Cloud Signature Consortium VZW - Architectures and protocols for remote signature applications Published version 1.0.3.0 (2018-12)*. URL: https://cloudsignatureconsortium.org/wp-content/uploads/2019/02/CSC_API_V1_1.0.3.0.pdf. Accessed: 2019-03-19 (cit. on p. 1).

Bibliography

- [10] *Digital Signatures in a PDF*. URL: https://www.adobe.com/devnet-docs/etk_deprecated/tools/DigSig/%20Acrobat_DigitalSignatures_in_PDF.pdf. Accessed: 2019-03-19 (cit. on p. 19).
- [11] *Digital Signatures in the PDF Language - Developer Technical Note*. URL: <https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/DigitalSignaturesInPDF.pdf>. Accessed: 2019-04-08 (cit. on p. 49).
- [12] *eIDAS Regulation - 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*. URL: <https://www.eid.as/home/#article25>. Accessed: 2019-03-19 (cit. on p. 1).
- [13] *eIDAS Regulation - 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*. URL: <https://www.eid.as/Regulation>. Accessed: 2019-06-17 (cit. on pp. 6–8).
- [14] *Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES*. URL: https://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf. Accessed: 2019-04-08 (cit. on pp. 11, 46, 48).
- [15] *ETSI TS 101 903 V1.4.2 (2010-12) Technical Specification Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)*. URL: https://www.etsi.org/deliver/etsi_ts/101900_101999/101903/01.04.02_60/ts_101903v010402p.pdf. Accessed: 2019-04-08 (cit. on p. 10).
- [16] *ETSI TS 103 173 V2.1.1 (2012-03) - Electronic Signatures and Infrastructures (ESI); CAdES Baseline Profile*. URL: https://www.etsi.org/deliver/etsi_ts/103100_103199/103173/02.01.01_60/ts_103173v020101p.pdf. Accessed: 2019-04-08 (cit. on p. 10).
- [17] Andreas Gregor Fitzek et al. "Fortgeschrittene PDF Signaturen mit PAdES." In: *eGovernment review* 15 (2015), pp. 16–17 (cit. on p. 23).

Bibliography

- [18] *Importance of WCAG 2.0 and PDF/UA*. URL: <https://commonlook.com/WCAG-20-and-PDF-UA-Your-Questions-Answered/>. Accessed: 2019-04-08 (cit. on p. 52).
- [19] *Introduction to Trusted Execution Environments*. URL: <https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf>. Accessed: 2019-07-29 (cit. on p. 27).
- [20] *iOS Security - iOS 12.1*. URL: https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. Accessed: 2019-03-19 (cit. on pp. 2, 28).
- [21] *ISO 19005-1:2005 Preview Document management – Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1)*. URL: <https://www.iso.org/obp/ui/#iso:std:iso:19005:-1:ed-1:v2:en>. Accessed: 2019-04-08 (cit. on p. 52).
- [22] Herbert Leitold, Reinhard Posch, and Thomas Rössler. “Media-break resistant eSignatures in eGovernment: an Austrian experience.” In: *IFIP International Information Security Conference*. Springer. 2009, pp. 109–118 (cit. on p. 22).
- [23] Thomas Lenz and Lukas Alber. “Towards cross-domain eid by using agile mobile authentication.” In: *2017 IEEE Trustcom/Big-DataSE/ICCESS*. IEEE. 2017, pp. 570–577 (cit. on p. 2).
- [24] Desta Mengistu, Hangjung Zo, and Jae Jeung Rho. “M-government: opportunities and challenges to deliver mobile government services in developing countries.” In: *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*. IEEE. 2009, pp. 1445–1450 (cit. on p. 2).
- [25] Vladislav Mladenov et al. “1 Trillion Dollar Refund—How To Spoof PDF Signatures.” In: () (cit. on pp. 1, 71).
- [26] *Mobile Phone Signature*. URL: <https://www.egiz.gv.at/en/e-government/4-buergerkarte#sub-handysignatur>. Accessed: 2019-05-11 (cit. on p. 22).

Bibliography

- [27] *Multi-factor Authentication Overview paper about state-of-the-art multi-factor authentication systems and mechanisms*. URL: <https://www.egiz.gv.at/files/projekte/2016/whitepaperMFA/Whitepaper-Multi-Faktor-Authentifizierung.pdf>. Accessed: 2019-03-19 (cit. on p. 26).
- [28] *PDF REFERENCE AND ADOBE EXTENSIONS TO THE PDF SPECIFICATION*. URL: https://www.adobe.com/devnet/pdf/pdf_reference.html. Accessed: 2019-03-19 (cit. on p. 1).
- [29] *PDF-AS*. URL: <https://www.egiz.gv.at/en/schwerpunkte/16-pdf-as>. Accessed: 2019-03-19 (cit. on pp. 1, 23).
- [30] *PDF-Over*. URL: <https://joinup.ec.europa.eu/solution/pdf-over>. Accessed: 2019-03-19 (cit. on pp. 1, 24).
- [31] *PDFlib whitepaper: "A Technical Introduction to PDF/UA"*. URL: <https://www.pdfliib.com/fileadmin/pdfliib/pdf/whitepaper/Whitepaper-Technical-Introduction-to-PDFUA.pdf>. Accessed: 2019-04-08 (cit. on p. 52).
- [32] Karl Christian Posch et al. "Secure and Privacy-Preserving eGovernment—Best Practice Austria." In: *Rainbow of computer science*. Springer, 2011, pp. 259–269 (cit. on pp. 22, 32).
- [33] *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*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG. Accessed: 2019-03-19 (cit. on pp. 5, 6).
- [34] *The AdES family of standards: CAdES, XAdES, and PAdES Implementation guidance for using electronic signatures in the European Union*. URL: https://blogs.adobe.com/security/91014620_eusig_wp_ue.pdf. Accessed: 2019-03-19 (cit. on p. 20).
- [35] Kevin Theuermann, Arne Tauber, and Thomas Lenz. "Mobile-Only Solution for Server-Based Qualified Electronic Signatures." In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7 (cit. on pp. 2, 27, 31).

Bibliography

- [36] Kevin Theuermann et al. "Flexible und benutzerfreundliche Authentifizierungsverfahren zur Umsetzung transaktionaler E-Government-Services auf mobilen Geräten." In: *Handbuch E-Government: Technikinduzierte Verwaltungsentwicklung* (2019), pp. 1–30 (cit. on p. 1).
- [37] Thomas Zefferer. "Towards Transactional Electronic Services on Mobile End-User Devices-A Sustainable Architecture for Mobile Signature Solutions." In: *11th International Conference on Web Information Systems and Technologies*. . 2015 (cit. on p. 2).
- [38] Thomas Zefferer and Vesna Krnjic. "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*. Springer. 2012, pp. 88–102 (cit. on p. 33).
- [39] Thomas Zefferer and P Teuf. "Opportunities and forthcoming challenges of smartphone-based mgovernment services." In: *Eduard Aibar* (2011), p. 56 (cit. on p. 2).