RUHR-UNIVERSITÄT BOCHUM

RUB

RUHR-UNIVERSITÄT BOCHUM

Sicherheitsanalyse und Evaluierung von signierten PDF Dokumenten

Simon Rohlmann

Masterarbeit – 14. November 2019.
Lehrstuhl für Netz- und Datensicherheit.

1. Prüfer: Prof. Dr. Jörg Schwenk
Betreuer: Dr.-Ing. Vladislav Mladenov
Betreuer: Dr.-Ing. Christian Mainka

hgi Lehrstuhl für Netz- und Datensicherheit

Zusammenfassung

Diese Arbeit befasst sich mit der Sicherheit von digitalen Signaturen in PDF Dokumenten. Hierfür wurde die Funktionsweise eingebauter Methoden zur Modifikationserkennung betrachtet und das Verhalten verschiedener PDF Betrachtungsprogramme im Manipulationsfall analysiert. Darüber hinaus wurden Manipulationsmöglichkeiten in Verbindung von signierten PDF Dokumenten und enthaltenen Schriftarten entwickelt. Weitere Analysefelder dieser Arbeit stellen die Angriffsklassen Incremental Saving und Shadow Attack dar. Beide bieten die Möglichkeit den Inhalt von signierten PDF Dokumenten zu manipulieren.
Um die Ziele dieser Arbeit zu erreichen, wurden Exploits für jede vorgestellte Angriffstechnik entwickelt und unter 18 PDF Betrachtungsprogrammen ausgewertet. Dabei ließen sich für jedes PDF Betrachtungsprogramm mindestens zwei Schwachstellen ausmachen.

Eidesstattliche Erklärung

Ich erkläre, dass ich keine Arbeit in gleicher oder ähnlicher Fassung bereits für eine andere Prüfung an der Ruhr-Universität Bochum oder einer anderen Hochschule eingereicht habe.

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen, die anderen Quellen dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen kenntlich gemacht. Dies gilt sinngemäß auch für verwendete Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Ich erkläre mich damit einverstanden, dass die digitale Version dieser Arbeit zwecks Plagiatsprüfung verwendet wird.

Datum	Unterschrift

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel wird die Motivation zu dieser Masterarbeit dargelegt. Es wird aufgezeigt, weshalb die Sicherheit von digitalen Signaturen in PDF Dokumenten eine bedeutende Rolle einnimmt und Schwachstellen im Validierungsprozess relevant für Wirtschaft und Verwaltung sind. Darüber hinaus werden die Ziele dieser Arbeit festgelegt und verwandte Arbeiten in einer Übersicht dargestellt.

1.1 Motivation

Im Juni 1993 veröffentlichte Adobe Systems das Referenzhandbuch zum Portable Document Format (PDF). Das Format baut auf der PostScript-Sprache auf und soll Texte sowie Grafiken in Form von digitalen Dokumenten plattformunabhängig für Benutzer verfügbar machen [32, S. 5].
Seit dem Jahr 2008 wird das Format in der Norm ISO-32000-1 der International Organization for Standardization spezifiziert [4].
Mit der Veröffentlichung des Referenzhandbuchs zu PDF 1.3 werden offiziell Signaturen unterstützt [3, S. 4]. Adobe unterscheidet zwischen digitalen Signaturen auf Basis asymmetrischer Kryptographie, welche zur Berechnung Schlüsselpaare, bestehend aus einem öffentlichen und privaten Teil, heranziehen sowie elektronischen Signaturen, die bspw. mit handschriftlichen Unterschriften gleichzusetzen sind [3, S. 451]. Eine digitale Signatur soll die drei Schutzziele: Integrität, Authentizität und Nicht-Zurückweisbarkeit erfüllen [11, S. 328]. So kann der Empfänger eines PDF Dokuments mit digitaler Signatur feststellen, wer das Dokument digital unterschrieben hat und ob es nachträglich verändert wurde. Der Unterzeichner wiederum kann wegen der digitalen Signatur, welche unter Verwendung seines privaten Schlüssels erstellt wurde, nicht abstreiten das Dokument unterschrieben zu haben.
Im Jahr 2018 wurden alleine in Adobe Produkten 250 Milliarden PDF Dokumente geöffnet, wovon 8 Milliarden eine elektronische oder digitale Signatur beinhalteten [1, S. 1]. Dazu kommen viele weitere PDF Dokumente die mit Betrachtungsprogrammen anderer Hersteller geöffnet wurden. Diese Zahlen zeigen die weite Verbreitung von PDF Dokumenten und digitaler Signaturen.
Verwendung finden digitale Signaturen unter anderem im E-Government, E-Justice oder im Onlinehandel. So führen beispielsweise die sächsischen Finanzämter PDF Signaturen explizit als zugelassenes Signaturaustauschformat auf [15]. Die gesetzliche Grundlage für rechtsgültige digitale Signaturen bietet in der Europäischen Uni-

on (EU) die eIDAS-Verordnung über elektronische Identifizierung und Vertrauensdienste [13]. Ergänzend dazu findet in Deutschland §17 des Vertrauensdienstegesetz (VDG) Anwendung [10].
Für digitale Signaturen im Allgemeinen und PDF Signaturen im Speziellen gibt es somit ein weitgefächertes Einsatzgebiet. Da digitale Signaturen rechtsgültigen Charakter besitzen, ist die Sicherheit und Unfälschbarkeit von entscheidender Bedeutung. Das Erforschen von Möglichkeiten, um nachträglich den Inhalt eines digital signierten PDF Dokuments zu ändern, ohne dabei die Signatur zu invalidieren, stellt somit eine wichtige Forschungsaufgabe dar.

1.2 Zielsetzung

In der Masterarbeit „Security of PDF Signatures“ von Karsten Meyer zu Selhausen [21] sowie im Paper „1 Trillion Dollar Refund - How To Spoof PDF Signature“ von Mladenov et al. [35] wurden verschiedene Techniken vorgestellt, die zu einer gültigen Signaturprüfung führen, obwohl der Inhalt des PDF Dokuments verändert wurde. Auf diesen Ergebnissen aufbauend, soll in dieser Masterarbeit weiter zur Sicherheit von digitalen Signaturen in PDF Dokumenten geforscht werden.
Adobe unterscheidet bei digitalen Signaturen zwischen den zwei Signaturtypen Approval und Certification. Während es vom Typ Approval mehrere Signaturen geben kann, darf eine Certification Signatur nur einmal im Dokument vorkommen und muss gleichzeitig die erste Signatur sein [8, S. 6-7]. Diese Unterscheidung wurde bisher nur oberflächlich betrachtet. Daher ist ein Ziel dieser Arbeit, die bereits veröffentlichten Exploits zu Re-Evaluieren und sie auf Certification signaturbasierte Dokumente zu übertragen.
Ein weiteres Ziel ist es, die Funktionsweise von Transformationsmethoden und -parametern der digitalen Signaturen zu erforschen. Diese Methoden und Parameter dienen für PDF Betrachtungsprogramme als Restriktion, um erlaubte Änderungen nach dem Signieren zu beschränken [8, S. 2]. In dieser Arbeit soll überprüft werden, wie die verschiedenen Betrachtungsprogramme auf Manipulationen an diesen Parametern und auf nachträgliche Änderungen reagieren.
Als letztes Ziel sollen weitere Schwachstellen in Hinblick auf digitale Signaturen in PDF Dokumenten identifiziert werden, welche Änderungen am Quelltext, bei gleichzeitig erfolgreicher Signaturvalidierung durch das Betrachtungsprogramm, ermöglichen.
Um die Auswertung der Manipulationsversuche zu unterstützen, soll ein Werkzeug entwickelt werden, dass die Evaluierung maßgeblich unterstützt.

1.3 Verwandte Arbeiten

Im Folgenden werden verwandte Arbeiten im Themengebiet der PDF Sicherheit vorgestellt.
Im Jahr 2011 präsentierte Dan-Sabin Popescu in seinem Paper „Hiding Malicious Content in PDF Documents“ [12] eine Möglichkeit Inhalte in einem PDF Dokument zu verstecken. Dafür erstellt der Angreifer eine Datei, die sowohl ein PDF Dokument als auch eine TIFF-Bilddatei enthält. Nachdem das Opfer die Datei digital signiert hat, kann der Angreifer den versteckten Inhalt nachträglich sichtbar machen, ohne dabei die Signatur zu invalidieren.
Im Jahr 2017 demonstrierten Stevens et al. in ihrem Paper „The first collision for full SHA-1“ [22] einen Kollisionsangriff auf die Hashfunktion SHA-1. Ihnen gelang es zwei unterschiedliche PDF Dokumente mit einem gemeinsamen Hashwert zu erzeugen. SHA-1 ist eine von mehreren möglichen Hashfunktionen, welche während des Erstellungsprozesses einer PDF Signatur genutzt werden können.
Im gleichen Jahr stellte Tomáš Stefan in seiner Bachelorarbeit „Digital Signature Verification in PDF“ [33] eine Bibliothek zur Validierung von PDF Signaturen vor und wies in diesem Zusammenhang auf potentielle Risiken hin.
Ebenfalls im Jahr 2017 veröffentlichten Markwood et al. das Paper „PDF Mirage: Content Masking Attack Against Information-Based Online Services“ [18]. Die Autoren demonstrierten in drei Angriffsvarianten Methoden, um den Inhalt von PDF Dokumenten zu verschleiern. Dabei präsentierten sie unter anderem, wie die Darstellung des PDF Dokuments durch manipulierte Schriftarten beeinflusst werden kann.
Im Jahr 2018 erschien die Masterarbeit „Security of PDF Signatures“ von Karsten Meyer zu Selhausen [21] und 2019 das Paper „1 Trillion Dollar Refund - How To Spoof PDF Signature“ von Mladenov et al. [35]. Die Autoren stellten drei verschiedene Angriffsklassen auf digitale Signaturen in PDF Dokumenten vor und demonstrierten in Exploits für 21 PDF Betrachtungsprogramme, wie sich Schwachstellen im Validierungsprozess von PDF Signaturen ausnutzen lassen, um den Inhalt von signierten PDF Dokumenten nachträglich zu manipulieren.
Ebenfalls im Jahr 2019 zeigten Müller et al. in ihrem Paper „Practical Decryption exFiltration: Breaking PDF Encryption“ [19] Möglichkeiten auf, um an den Inhalt eines verschlüsselten PDF Dokuments zu gelangen. Wird ein manipuliertes PDF Dokument durch das Opfer geöffnet und entschlüsselt, sendet das PDF Betrachtungsprogramm den Klartext an einen vom Angreifer kontrollierten Webserver. Hierfür manipuliert der Angreifer das PDF Dokument entweder in einem unverschlüsselten Teil oder nutzt Cipher Block Chaining (CBC) Gadgets, um die Manipulation im verschlüsselten Teil des Dokuments durchzuführen.

1.4 Aufbau dieser Arbeit

Diese Masterarbeit ist in sieben Hauptkapitel gegliedert. Das erste Kapitel leitet die Arbeit ein und gibt einen Überblick über Motivation und Zielsetzung. Durch das zweite Kapitel werden dem Leser ein grundlegendes Basiswissen zum Aufbau von PDF Dokumenten und enthaltenen digitalen Signaturen vermittelt. Diese Grundlagen sind nötig, um die Funktionsweise der Manipulationen und verarbeitenden Betrachtungsprogramme zu verstehen. Im dritten Kapitel werden die beiden Angreifermodelle vorgestellt, die den Manipulationen zu Grunde liegen. Die in Kapitel vier dargestellten Angriffsklassen unterscheiden die verschiedenen Manipulationsmethoden der in dieser Arbeit behandelten Angriffe. Das im Rahmen dieser Masterarbeit entstandene Programm zur Unterstützung des Evaluationsprozesses wird in Kapitel fünf beschrieben. Dabei wird auf Funktionsweise und Aufbau der einzelnen Module eingegangen. Das Kapitel sechs enthält die Ergebnisse der Auswertung. Es ist in mehrere Abschnitte eingeteilt, welche die Ergebnisse der verschiedenen Manipulationsvarianten in den überprüften PDF Betrachtungsprogrammen diskutieren. Den Abschluss bildet das Kapitel sieben mit einem Fazit zu den erreichten Ergebnissen dieser Arbeit. Dabei wird außerdem ein Ausblick auf weitere Forschungsarbeiten mit einem ähnlichen Themengebiet gegeben.

2 Grundlagen

Das Grundlagenkapitel führt in den Aufbau von PDF Dokumenten und digitaler Signaturen im PDF Kontext ein. Als Basis dient hierbei der ISO-Standard 32000-1:2008 [4], welcher die Funktionalität der PDF Version 1.7 spezifiziert.

2.1 Allgemeiner Aufbau von PDF Dokumenten

Ein PDF Dokument gliedert sich in vier Teile: Header, Body, Cross-Reference Table und Trailer. Der Aufbau ist anhand eines minimalen Beispieldokuments in Abbildung 2.1 dargestellt. Der Header bildet die erste Zeile des Dokuments und legt die verwendete PDF-Version fest. Falls das Dokument Binärdaten enthält, sollte nach dem Header eine Kommentarzeile mit mindestens vier binären Zeichen folgen. Diese dient dazu, dass Programme, die das Dokument für den Dateitransfer durchleuchten, den Inhalt als Binärdaten und nicht als Textdaten interpretieren. Der Inhalt des PDF Dokuments ist in Objektform organisiert, welche in der Gesamtheit den Body bilden. Über die Cross-Reference Table werden die indirekten Objekte im Body referenziert. Dadurch erhält ein PDF Betrachtungsprogramm Informationen über die Position der enthaltenen Objekte, ohne den Inhalt des gesamten Dokuments einlesen zu müssen. Der Trailer leitet das Ende des Dokuments ein und enthält unter anderem Informationen zur Position der Cross-Reference Table und die Nummer von speziellen Objekten [4, S. 39-42].

2.1.1 Objekte

Bei Objekten wird unter acht verschiedenen Typen unterschieden: Boolesche Werte, Integer und reale Zahlen, Zeichenketten, Namen, Arrays, Dictionaries, Streams sowie das Null-Objekt. Ein Boolescher Wert kann innerhalb eines Dokuments `true` oder `false` annehmen. Innerhalb der Zahlenobjekte unterscheidet man zwischen ganzzahligen Werten (Integer Objekte) und Zahlenwerten mit Nachkommastellen (Real Objekte). Dabei können beide Objektarten positive oder negative Vorzeichen führen. Eine Zeichenkette wird mit einer öffnenden runden Klammer begonnen und enthält eine endliche Anzahl von Zeichen. Mit der schließenden runden Klammer wird das Ende der Zeichenkette angegeben. Bei Zeichenketten, welche von spitzen Klammern

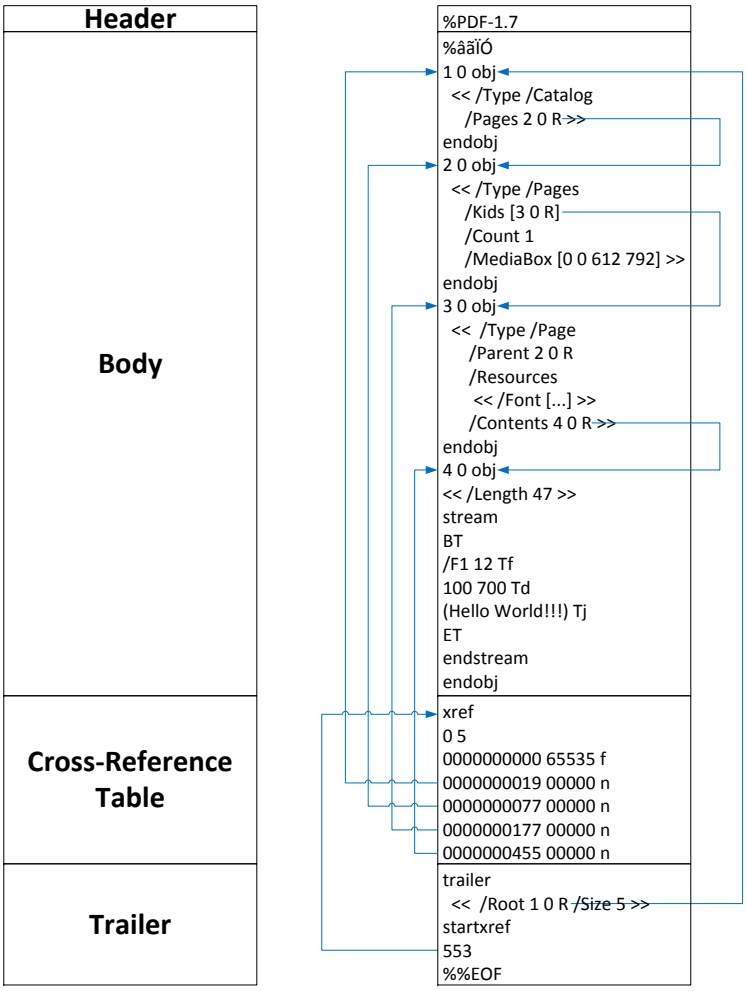

Abbildung 2.1: Aufbau eines PDF Dokuments [4, S. 39].

umschlossen werden, muss der Inhalt eine hexadezimale Darstellung aufweisen. Dabei wird eine hexadezimale Zeichenkette mit einer Null am rechten Ende aufgefüllt, falls sie eine ungerade Anzahl von Zeichen enthält. So wird bspw. `<A0F91>` als `A0, F9, 10` interpretiert. Durch einen vorangestellten Schrägstrich, wird ein Namens-Objekt eingeleitet und durch eine ununterbrochene Zeichenfolge fortgeführt. Eine weitere Objektvariante bilden Arrays. Diese können verschiedene Objekte, unterschiedlichen Typs, aufnehmen. Der Inhalt wird dabei durch eckige Klammern umschlossen. Ein Dictionary-Objekt kann verschiedene Paare von Objekten aufnehmen, welche als Dictionary-Eintrag bezeichnet werden. Ein solcher Eintrag gliedert sich in Schlüssel- und Werteelement. Die Einträge werden durch doppelte spitze Klammern umschlossen, wobei es innerhalb eines Dictionary-Objekts auch weitere Dictionary-Objekte geben darf. Ein solches Objekt ist bspw. nach der Zeile `1 0 obj` in Abbildung 2.1 dargestellt. Um Objekte mit großer Datenmenge in ein PDF Dokument

zu integrieren, eignet sich das Stream-Objekt. Einem solchen Objekt vorangestellt ist ein Dictionary-Objekt, welches mindestens den Parameter `Length`, gefolgt von der Byte-Länge des Streams, enthält. Die Byte-Werte werden durch `stream` und `endstream` umschlossen. Ein Beispiel hierfür ist in Abbildung 2.1 nach der Zeile `4 0 obj` dargestellt. Ein Null-Objekt besteht lediglich aus dem Eintrag `null`. Enthält bspw. ein Dictionary-Eintrag den Wert `null`, dann soll dieser Eintrag von der Verarbeitungslogik übergangen werden [4, S. 13-21].
In Tabelle 2.2 sind die beschriebenen Objekte mit Beispielen dargestellt.

Tabelle 2.2: Übersicht der Objekttypen des PDF Standards mit Beispielen [4, S. 13-21].

Objekttyp	**Syntax**	**Beispiel**
Boolescher Wert	`true` oder `false`	siehe Syntax-Spalte

Zahl:
• Integer
• Real
	
	ganzzahlig
	Fließkommazahl
		
		`123`
		`12.3`

Zeichenkette:
• beliebige Zeichen
• hexadezimal
	
	von runden Klammern umschlossen
	von spitzen Klammern umschlossen
		
		`(abc123)`
		`<A0F9>`

Name	ununterbrochene Zeichenkette mit vorangestelltem Schrägstrich	`/Type`
Array	Objekte von eckigen Klammern umschlossen	`[0 1 (abc123)]`
Dictionary	Paare von Objekten mit doppelten spitzen Klammern umschlossen	`« /Type /Catalog /Pages 2 0 R »`

Stream	Byte-Werte von `stream` und `endstream` umschlossen sowie vorangestelltem Dictionary	`« /Length 13 »`
		`stream`
		`(Hello World)`
		`endstream`

Null	`null`	siehe Syntax-Spalte

2.1.2 Cross-Reference Table

Die Cross-Reference Table wird mit `xref` eingeleitet, gefolgt von zwei Zahlen in der nächsten Zeile. Dabei steht die erste Zahl für die erste Objektnummer und die zweite Zahl für die Anzahl der direkt darauffolgenden Objektnummern. Lautet die zweite Zeile bspw. `0 5`, so erwartet die Verarbeitungslogik die Einträge für die Objektnummern 0, 1, 2, 3, 4. Es ist möglich diesen Eintrag auch an späterer Stelle zusätzlich einfließen zu lassen. So wird die Verarbeitungslogik im Fall der `Tabelle 1` in Abbildung 2.3 die indirekten Objekte 0-4 erwarten, während bei `Tabelle 2` in selbiger Abbildung die indirekten Objekte 0-2 und 10-11 erwartet werden. Die weiteren Abschnitte der Tabelle sind wie folgt aufgebaut:

```
nnnnnnnnnn ggggg x eol
```

Die Position des indirekten Objekts im Dokument wird durch Angabe des dezimalen Byte-Wertes beschrieben. Dabei wird der Positionswert (hier durch `n` dargestellt) linksseitig mit Nullen aufgefüllt, bis eine zehnstellige numerische Zeichenkette erreicht ist. Als nächster Wert folgt die fünfstellige dezimale Generationsnummer `g`. Das nachfolgende `x` kann entweder den Wert `n` annehmen, falls das Objekt in Verwendung ist oder `f`, um den Eintrag als frei zu markieren. Der letzte Teil kennzeichnet das Ende der Zeile. Für Einträge von indirekten Objekten, welche sich in Verwendung befinden, sollte die Generationsnummer 0 gesetzt werden. Wird ein Eintrag durch die Angabe des Wertes `f` als frei markiert, soll die Generationsnummer um eins erhöht werden, wobei der Wert 65535 die maximal mögliche Generationsnummer bildet. Ein- bis vierstellige Werte werden entsprechend linksseitig mit Nullen aufgefüllt, bis eine fünfstellige numerische Zeichenkette erreicht ist. Der erste Objekteintrag innerhalb der Cross-Reference Table sollte stets die Objektnummer 0 tragen, als frei markiert sein und die Generationsnummer 65535 besitzen [4, S. 40-42].

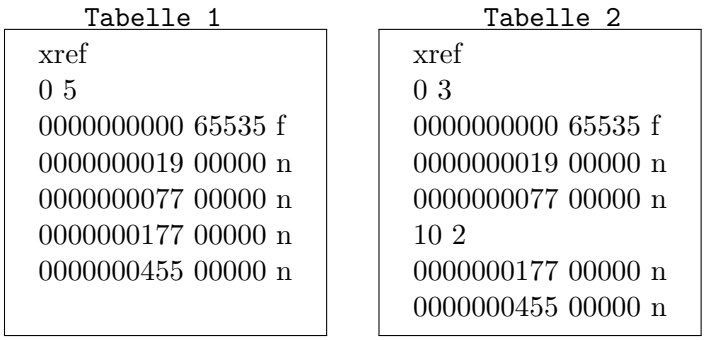

Abbildung 2.3: Darstellung einer Cross-Reference Table mit Referenzierung der Objekte 0-4 (`Tabelle 1`) bzw. der Objekte 0-2 und 10-11 (`Tabelle 2`).

2.1.3 Trailer

Durch die Zeichenkette `trailer` wird selbiger eingeleitet. Die nächste Zeile öffnet ein Dictionary-Objekt, in dem mindestens die Einträge `Size` und `Root` enthalten sein müssen. Der Wert von `Size` sollte immer um eins höher sein, als die höchste Objektnummer. In der Abbildung 2.1 lautet die höchste Objektnummer 4, weshalb für `Size` der Wert 5 gewählt wird. Der `Root`-Eintrag referenziert auf das Wurzelelement des Dokuments, welches den `Type`-Eintrag `Catalog` enthält. Nach der Zeichenkette `startxref` befindet sich ein Byte-Wert, welcher auf die Position der Cross-Reference Table innerhalb des Dokuments zeigt. Der anschließende Wert `%%EOF` leitet das Ende des Dokuments bzw. bei Dokumenten mit enthaltenen Incremental Updates (siehe Abschnitt 2.2.3) das Ende des Teilabschnitts ein [4, S. 42-43].

2.1.4 Kommentarfunktion

Das Prozentzeichen dient zum Öffnen eines Kommentars und weist die Verarbeitungslogik an die restliche Zeile zu ignorieren. Bei einer Zeile, wie „Sicherheit von% PDF Signaturen“, wird bspw. nur der Teil vor dem Prozentzeichen, also „Sicherheit von“, interpretiert. Eine Ausnahme bilden die Header-Zeile sowie der `%%EOF`-Eintrag, welche trotz Prozentzeichen verarbeitet werden sollen [4, S. 13].

2.2 Dokumentstruktur

2.2.1 Indirekte Objekte und Referenzierung

Ein indirektes Objekt kann verschiedene Objekte beinhalten. Es beginnt mit der Objekt- und Generationsnummer, gefolgt von `obj` und endet mit `endobj`. Die Generationsnummer soll für neu erstellte Objekte 0 sein, wobei sie bei Aktualisierungen des Dokuments erhöht werden kann. Indirekte Objekte lassen sich über Ihre Objekt- und Generationsnummer referenzieren [4, S. 21].
In Abbildung 2.1 wird nach der Zeile `trailer` auf das Objekt `1 0` referenziert. Dieses indirekte Objekt enthält ein Dictionary-Objekt mit den Einträgen `/Type /Catalog` und `/Pages 2 0 R`. Dieses indirekte Objekt bildet die Wurzel des Dokuments. D.h. jedes weitere referenzierte Objekt hat seinen Ursprung im Wurzelobjekt, wobei keine direkte Beziehung zwischen diesen beiden Objekten bestehen muss [4, S. 71].
Mit dem Eintrag `/Pages 2 0 R` wird auf ein indirektes Objekt referenziert, welches die Basisinformationen zu den Seiten innerhalb des PDF Dokumentes bereithält. Es referenziert über den Eintrag `Kids` die enthaltenen Objekte mit Typ `Page`. Der Eintrag `Count` gibt die Anzahl der enthaltenen Seiten im PDF Dokument an und sollte den Einträgen im Array-Objekt des `Kids`-Eintrag entsprechen. Über den Eintrag `MediaBox` wird der Verarbeitungslogik die Koordinaten des Seitenmediums mitgeteilt [4, S. 75-76].
Das nächste referenzierte indirekte Objekt besitzt die Objektnummer 3 und einen `Type`-Eintrag `Page`. Innerhalb dieses Objekts wird über `Parent` auf das übergeordnete indirekte Objekt referenziert. Darüber hinaus kann über das `Resources`-Dictionary bspw. eine Schriftart referenziert oder beschrieben werden. Der `Contents`-Eintrag referenziert auf den Inhalt der Seite [4, S. 77-79].
Das Inhaltsobjekt, nach der Zeile `4 0 obj` in Abbildung 2.1, beginnt mit der Angabe der Byte-Länge des darauffolgenden Streams. Innerhalb des Streams finden sich Parameter, um Position und Aussehen der enthaltenen Zeichenkette zu steuern [4, S. 81-83].

2.2.2 Relevante Elemente

Für die Verwendung von digitalen Signaturen sind spezielle Elemente nötig, die im Folgenden kompakt beschrieben werden. Die Abbildung 2.4 stellt das Zusammenspiel dieser Elemente dar.

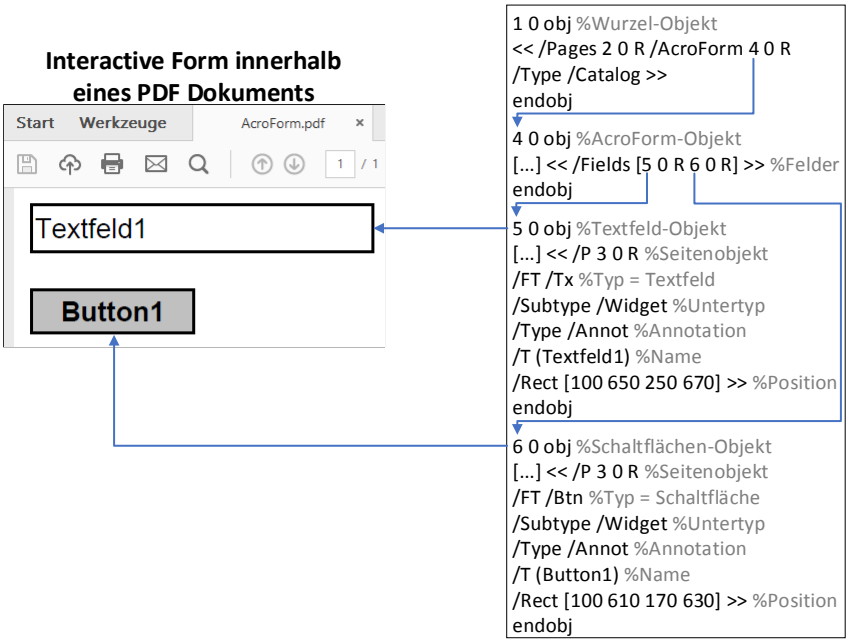

Abbildung 2.4: Beispiel eines PDF Dokuments, welches ein Textfeld sowie eine Schaltfläche über eine Interactive Form bereithält.

Annotations

PDF Seiten bieten unter anderem die Möglichkeit Texte, Grafiken, Notizen, Tonaufnahmen, Videos usw. aufzunehmen. Diese werden über Annotations adressiert und können über die Benutzeroberfläche des PDF Betrachtungsprogramms mit dem Anwender interagieren. Durch ein Annotation-Dictionary werden die Eigenschaften dieser Objekte festgelegt. Hier bietet bspw. der Eintrag `P` die Möglichkeit das Seitenobjekt auszuwählen und über `Rect` die Position der Annotation innerhalb des Seitenobjekts anzugeben [4, S. 381].

Field-Dictionaries

Bestehende Felder, bspw. innerhalb eines Formulars, werden in Field-Dictionaries organisiert. Jedes Feld wird durch den `Field Type (FT)` bestimmt. Dabei sind folgende Zuordnungen möglich:

- `Btn`: Schaltfläche
- `Tx`: Text
- `Ch`: Auswahl
- `Sig`: Signatur

Die Bearbeitung eines solchen Feldes kann durch den Eintrag `Field Flag (Ff)` eingeschränkt werden. So veranlasst der Wert `1` das PDF Betrachtungsprogramm dazu das Feld als `ReadOnly` zu betrachten und keine Änderungen zuzulassen. `2` markiert das Feld als Pflichtfeld und `3` verbietet den Export des Feldes [4, S. 432-433]

Interactive Forms

Eine Interactive Form wird im Quelltext eines PDF Dokuments als `AcroForm` bezeichnet und fasst verschiedene Felder in einem Formular zusammen. Interactive Forms nutzen Annotations für die Darstellung von Feldern und die Interaktion mit dem Anwender. Hierfür wird der Annotation Untertyp `Widget` verwendet. Innerhalb des Interactive Form Dictionary werden die Formularfelder als Array im Eintrag `Fields` aufgeführt. Der optionale Eintrag `SigFlags` weist das PDF Betrachtungsprogramm auf existierende Signaturfelder hin [4, S. 430-432].

2.2.3 Incremental Updates

Der PDF Standard sieht vor, dass Änderungen, wie bspw. das Hinzufügen eines Kommentars, einer Signatur oder einer Änderung am Inhalt, via Updates durchgeführt werden. Der Ursprungsinhalt des Dokuments bleibt dabei unverändert. Damit bei einer Änderung nicht immer der gesamte Inhalt des Dokuments neu hinzugefügt werden muss, sind die Updates inkrementell durchzuführen. D.h. ändert sich bspw. der Inhalt des Dokuments von „Hello World“ zu „Hello Attacker“, beinhaltet der Body-Bereich des Incremental Updates lediglich das betreffende Inhaltsobjekt. Neben dem neuen Body-Bereich wird auch die Cross-Reference Table und der Trailer als Update hinzugefügt. Die Tabelle enthält die neuen Byte-Werte der Objekte im Body-Update, während der neue Trailer auf die neue Cross-Reference Table referenziert und die alte Tabelle als Vorgänger im `Prev`-Eintrag festhält. Die Abbildung 2.5 stellt den beschriebenen Vorgang dar. Ein PDF Dokument kann dabei mehrere Incremental Updates enthalten, die immer ans Ende des Dokuments angefügt werden [4, S. 44-41].

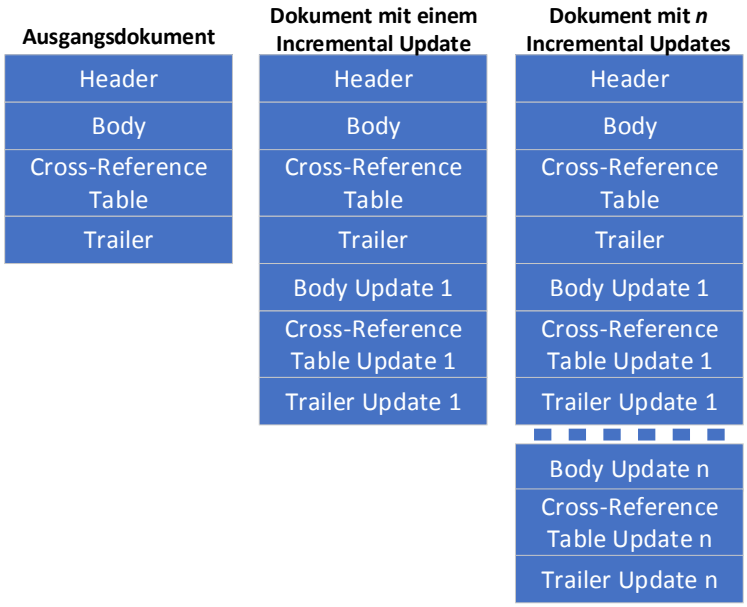

Abbildung 2.5: Angewendete Incremental Updates [4, S. 45].

2.3 Digitale Signaturen im PDF Kontext

2.3.1 Aufbau

Die Signaturinformationen werden in einem Signatur-Dictionary festgehalten. Es besteht aus vier Pflichteinträgen und kann weitere optionale Einträge aufnehmen. Diese werden im Kapitel 12.8 der PDF 1.7 Spezifikation über digitale Signaturen beschrieben [4, S. 466-478].
Zu den Pflichteinträgen zählen:

- `Filter`: Legt die Form der Authentifizierung durch den Signatur-Handler fest. Als Wert kann bspw. einer der folgenden Signatur-Handler festgelegt werden: `Adobe.PPKLite`, `Entrust.PPKEF`, `CICI.SignIt` oder `VeriSign.PPKVS`.
- `Contents`: Beinhaltet die Signaturdaten als Distinguished Encoding Rules (DER)-kodiertes Binärdatenobjekt vom Typ Public-Key Cryptography Standards (PKCS)#1 oder PKCS#7. Dabei soll die Variante PKCS#1 bei Verwendung des `SubFilters adbe.x509.rsa.sha1` gewählt werden. Im Fall des `SubFilters adbe.pkcs7.detached` oder `adbe.pkcs7.sha1` soll die PKCS#7 Variante genutzt werden. Die `SubFilter` sind in Tabelle 2.7 dargestellt. Im Fall einer PKCS#7 Kodierung soll das Objekt konform zur Spezifikation im Request for Comments (RFC) 3852 [29] gestaltet werden. Ein Verweis zur RFC Konformität, beim Einsatz einer PKCS#1 Kodierung, findet sich nicht in der Spezifikation zum PDF 1.7 Standard.

- `Cert`: Ist nur unter Verwendung des `adbe.x509.rsa.sha1 SubFilters` ein Pflichtfeld. Im Verwendungsfall soll dieses Feld die X.509[1] Zertifikatskette beinhalten.
- `ByteRange`: Beinhaltet die Byte-Werte zu den Teilen des Dokuments, welche durch die Signatur geschützt sind. Es wird in Form eines Array-Objekts angegeben: `[a b c d]`. Die Eingabe zum Bilden der Signatur wird in zwei Teile aufgeteilt. Der erste Teil beginnt beim Byte-Wert `a` und besitzt die Byte-Länge `b`. Der zweite Teil beginnt entsprechend beim Byte-Wert `c` und besitzt die Byte-Länge `d`. Der ausgelassene Byte-Wert zwischen `a + b` und `c` besteht aus dem Wert des `Contents`-Objekts, also den zu erstellenden Signaturdaten. Die Abbildung 2.6 stellt den durch die Signatur geschützten Bereich grafisch dar.

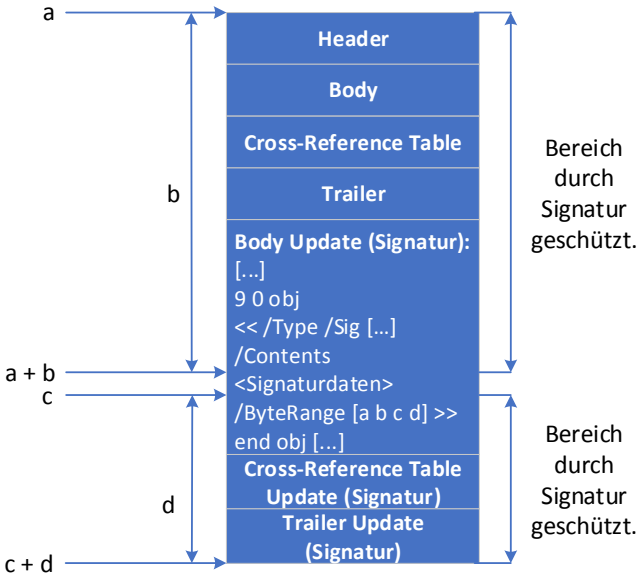

Abbildung 2.6: Darstellung der ByteRange innerhalb eines signierten PDF Dokuments [35, S. 6].

Neben den Pflichteinträgen sind weitere optionale Einträge möglich:

- `Type`: Für das Signatur-Dictionary sollte `Sig` gewählt werden.
- `SubFilter`: Gibt Auskunft über die verwendete Kodierung der Signatur- und Schlüsseldaten. Die unterstützten Algorithmen sind in Tabelle 2.7 aufgelistet.
- `Reference`: Array des Signatur-Reference-Dictionarys (siehe Abschnitt 2.3.3).
- `Changes`: Hält Informationen zu durchgeführten Änderungen am Dokument zwischen zwei angewendeten Signaturen bereit.
- `Name`: Name der Person, die die Signatur angewendet hat.

[1] X.509: Spezifikation zum Beschreiben das Formats und der Semantik von Zertifikaten und Zertifikatssperrlisten in einer Public Key Infrastructure (PKI) [14].

- `M`: Signierungszeitpunkt.
- `Location`: Name des Computers oder des geografischen Ortes der Signaturerstellung.
- `Reason`: Der Grund, weshalb das Dokument signiert wurde.
- `ContactInfo`: Kontaktinformationen der Person, die die Signatur angewendet hat.
- `R`: Version des Signatur-Handlers.
- `V`: Version des Signatur-Dictionary Formats.
- `Prop_Build`: Informationen zur Computerumgebung, welche zum Signieren verwendet wurde.
- `Prop_AuthTime`: Sekunden seit der letzten Authentifizierung der signierenden Person.
- `Prop_AuthType`: Methoden zur Authentifizierung der signierenden Person: Persönliche Identifikationsnummer (PIN), Passwort oder Fingerabdruck.

Tabelle 2.7: Unterstützte Algorithmen des `SubFilter`-Eintrags [4, S. 476].

	adbe.pkcs7.detached	**adbe.pkcs7.sha1**	**adbe.x509.rsa.sha1***
Message Digest (Hashfunktion)	SHA1 (PDF 1.3) SHA256 (PDF 1.6) SHA384 (PDF 1.7) SHA512 (PDF 1.7) RIPEMD160 (PDF 1.7)	SHA1 (PDF 1.3)	SHA1 (PDF 1.3) SHA256 (PDF 1.6) SHA384 (PDF 1.7) SHA512 (PDF 1.7) RIPEMD160 (PDF 1.7)
RSA Algorithmus Unterstützung	Bis zu 1024-bit (PDF 1.3) Bis zu 2048-bit (PDF 1.5) Bis zu 4096-bit (PDF 1.5)	Bis zu 1024-bit (PDF 1.3) Bis zu 2048-bit (PDF 1.5) Bis zu 4096-bit (PDF 1.5)	Bis zu 1024-bit (PDF 1.3) Bis zu 2048-bit (PDF 1.5) Bis zu 4096-bit (PDF 1.5)
DSA Algorithmus Unterstützung	Bis zu 4096-bit (PDF 1.6)	Bis zu 4096-bit (PDF 1.6)	Nein

*Trotz der Zeichenkette „sha1“ im Namen, ist dieser `SubFilter` nicht auf die Verwendung der SHA1 Hash-Funktion limitiert.

Neben dem Signatur-Dictionary, ist auch das Signatur-Field-Dictionary ein Teil der digitalen Signatur innerhalb eines PDF Dokuments. Es beinhaltet die Position des grafischen Teils der Signatur und ist vom Field-Typ `Sig`. Der Eintrag `V` innerhalb des Field-Dictionarys referenziert auf das Signatur-Dictionary. Darüber hinaus beinhaltet es die optionale Referenz auf das Lock-Dictionary. Hierüber lässt sich die Bearbeitung von Formularfelder nach der Signierung restriktieren. Es lassen sich über den `Action`-Eintrag wahlweise alle Formularfelder sperren oder nur einzelne Felder ein- bzw. ausschließen [4, S. 446-447].
Das Listing 2.8 enthält Beispielobjekte für das Field und Lock-Dictionary.

```
8 0 obj %Field-Dictionary einer Signatur
<< /FT /Sig %Field-Typ = Signatur
/Type /Annot %Annotation
/Subtype /Widget %Untertyp der Annotation
/T (Signature1) %Name
/P 3 0 R %Referenz auf das Seitenobjekt
/Rect [150.0 400.0 450.0 480.0] %Position
/Lock 7 0 R %Referenz auf das Lock-Dictionary
/V 9 0 R >> %Referenz auf das Signatur-Dictionary
endobj

7 0 obj %Lock-Dictionary
<< /Action /Include %In Fields aufgeführte Felder werden eingeschlossen
/Type /SigFieldLock %Dictionary-Typ
/Fields [(Button1) (Textfield1)] >> %Namen der eingeschlossenen Felder
endobj
```

Listing 2.8: Beispiel für Field und Lock-Dictionary.

2.3.2 Signaturtypen

Innerhalb von digitalen Signaturen unterscheidet Adobe zwischen Certification und Approval Signaturen, welche verschiedene Anwendungsfälle abdecken. Darüber hinaus lassen sich auch beide Signaturtypen kombinieren. Die Abbildung 2.9 zeigt die grafische Anwenderinformation seitens Adobe für Certification und Approval Signaturen sowie die Kombination der beiden Signaturtypen.

Abbildung 2.9: Anwenderinformation beim Öffnen eines signierten PDF Dokuments. (1) Certification Signatur, (2) Approval Signatur, (3) Kombination beider Signaturtypen.

Certification Signaturen

Eine Certification Signatur muss immer die erste Signatur innerhalb eines PDF Dokuments sein. Des Weiteren darf ein Dokument jeweils nur eine Signatur vom Typ Certification aufweisen. Mit der Signierung legt der Autor die erlaubten nachträglichen Änderungen am Dokument fest. Diese Restriktionen werden über den `DocMDP`-Parameter festgelegt (siehe Abschnitt 2.3.3) [7].

Approval Signaturen

Ein PDF Dokument kann mehrere Approval Signaturen beinhalten. So kann bspw. ein Vertrag von mehreren Parteien signiert werden. Über den `FieldMDP`-Parameter lässt sich das Bearbeiten von Formularfelder nach der Signierung einschränken (siehe Abschnitt 2.3.3). Falls ein PDF Dokument mit Certification Signatur das nachträgliche Signieren erlaubt, können Approval Signaturen auf Dokumente mit Certification Signatur angewendet werden [4, S. 467].

2.3.3 Transformationsmethoden

Der PDF Standard sieht die Möglichkeit vor nachträgliche Nutzereingaben zu einem signierten Dokument hinzuzufügen. Diese werden durch die Parameter der Modification Detection and Prevention (MDP) gesteuert und in einem Signatur-Reference-Dictionary gespeichert, welches durch den `Reference`-Eintrag im Signatur-Dictionary referenziert wird. Das Signatur-Reference-Dictionary besitzt den `Type`-Eintrag `SigRef` und legt über `TransformMethod` die Art der Methode fest (DocMDP, FieldMDP oder UR3). Das zur Methode zugehörige Parameter-Dictionary wird über den Eintrag `TransformParams` festgelegt. Bei gewählter FieldMDP-Methode soll der `Data`-Eintrag eine Referenz zum Objekt enthalten, auf das die Modifikationsanalyse angewendet werden soll. Dies ist in der Regel das Wurzelobjekt vom Typ `Catalog`. Im `DigestMethod`-Eintrag wird der verwendete Hash-Algorithmus hinterlegt. Nicht erlaubte nachträgliche Nutzereingaben sollen durch die jeweilige Methode erkannt werden und zu einer ungültigen Signatur führen [4, S. 469-470].

DocMDP

Über die DocMDP-Methode lässt sich steuern, welche Aktionen nach einer Signierung auf Certification-Basis erlaubt bleiben, ohne dabei die Signatur zu invalidieren. Die Festlegung erfolgt über den `P`-Eintrag des Transform-Parameter-Dictionarys, der die Werte `1-3` akzeptiert. Mit `1` wird festgelegt, dass keine Änderungen erlaubt sind. Der Parameter `2` erlaubt das Ausfüllen von Formularen sowie das Anfügen von Approval Signaturen. Mit `3` besteht zusätzlich zu den Berechtigungen aus `2` die Möglichkeit Kommentare über Annotation-Objekte hinzuzufügen. DocMDP bezieht sich immer auf das gesamte Dokument, daher kann dieser Parameter nur in Verbindung mit einer Certification Signatur verwendet werden [8, S. 2, 9-10].
In Tabelle 2.10 sind die verschiedenen Parameter zusammengefasst. Darüber hinaus findet sich in Listing 2.11 ein Beispiel zur Verwendung der DocMDP-Methode.

Tabelle 2.10: Übersicht zu den verfügbaren DocMDP-Parametern [8, S. 2, 9-10].

Parameter	Erlaubte Änderungen
`P 1`	Keine Änderungen erlaubt.
`P 2`	Formulare ausfüllen und digital signieren.
`P 3`	Annotations hinzufügen (bspw. Kommentare), Formulare ausfüllen und digital signieren.

```
10 0 obj %Signatur-Reference-Dictionary
<< /Type /SigRef %Typ = Signatur-Reference
/DigestMethod /SHA1 %Verwendeter Hashalgorithmus
/TransformParams 11 0 R %Referenz auf das Transform-Parameter-Dictionary
/TransformMethod /DocMDP >> %Auswahl der DocMDP-Transformationsmethode
endobj

11 0 obj %Transform-Parameter-Dictionary
<< /Type /TransformParams %Typ = Transform-Parameter
/V /1.2 %Versionsnummer
/P 1 >> %Parameter 1 = Keine Änderungen erlaubt
endobj
```

Listing 2.11: Beispiel für ein DocMDP Signatur-Reference und Transform-Parameter-Dictionary.

FieldMDP

Bei Verwendung von Approval Signaturen lässt sich die Bearbeitung von Formularfeldern nach dem Signieren einschränken. Über den `Action`-Eintrag können alle Formularfelder geschützt werden (`All`), nur bestimmte Felder (`Include`) oder Felder ausgeschlossen werden (`Exclude`). Im Schutz ein- bzw. ausgeschlossene Formularfelder werden über den Eintrag `Fields` per Feldname adressiert [8, S. 2, 9]. Bei Signierung soll der Inhalt für die Einträge `Action` und `Fields` aus den entsprechenden Einträgen des Lock-Dictionary (Abschnitt 2.3.1) kopiert werden [4, S. 474].
In Tabelle 2.12 sind die verschiedenen Einschränkungen zusammengefasst. Darüber hinaus findet sich in Listing 2.13 ein Beispiel zur Verwendung der FieldMDP-Methode.

Tabelle 2.12: Übersicht zu den verfügbaren FieldMDP-Einschränkungen [8, S. 2, 9].

Action	Zu schützende Formularfelder
`All`	Alle Formularfelder.
`Include`	Nur die unter `Fields` angegebenen Formularfelder.
`Exclude`	Alle außer die unter `Fields` angegebenen Formularfelder.

```
10 0 obj %Signatur-Reference-Dictionary
<< /Type /SigRef %Typ = Signatur-Reference
/DigestMethod /SHA1 %Verwendeter Hashalgorithmus
/TransformParams 11 0 R %Referenz auf das Transform-Parameter-Dictionary
/TransformMethod /FieldMDP %Auswahl der FiledMDP-Transformationsmethode
/Data 1 0 R >> %Referenz auf das Wurzelobjekt
endobj

11 0 obj %Transform-Parameter-Dictionary
<< /Type /TransformParams %Typ = Transform-Parameter
/V /1.2 %Versionsnummer
/Action /All >> %Action All = Es darf kein Formularfeld nach dem Signieren verändert werden
endobj
```

Listing 2.13: Beispiel für ein FieldMDP Signatur-Reference und Transform-Parameter-Dictionary.

UR3

Die UR3-Methode erlaubt es zusätzliche Rechte (Usage Rights) zu aktivieren. Auf diese Weise kann der Autor eines Dokuments einem Anwender, mit Adobe Reader als Betrachtungsprogramm, zusätzliche Rechte einräumen. Folgende Rechteausprägung ist möglich: Bearbeitung und Verwendung von Formularfelder und Kommentarfunktionen, Hinzufügen und Entfernen von Signaturen sowie eingebetteten Dateien. Damit die gewährten Rechte nicht beliebig vom Anwender verändert werden können, sind selbige durch eine digitale Signatur geschützt. Wobei das Zertifikat von einer durch Adobe autorisierten Zertifizierungsstelle stammen muss. Im Gegensatz zu digitalen Signaturen, welche durch den Anwender platziert werden, erfolgt innerhalb des Betrachtungsprogramms für die UR3-Methode kein Signaturhinweis [8, S. 2, 10]. Bei Verwendung von Adobe Reader XI und höher ist es nicht mehr notwendig zusätzliche Rechte für das Ausfüllen von Formularen, Hinzufügen von Kommentaren oder das digitale Unterschreiben von PDF Dokumenten zu erteilen [2].

2.3.4 Signierungsprozess

Soll ein PDF Dokument digital signiert werden, benötigt der Anwender eine digitale Identifikation (ID), welche als Zertifikatsdatei, Smartcard / Token, ID Server, in der Mac Schlüsselbundverwaltung (Apple) oder im Windows Zertifikatsspeicher (Microsoft) vorliegen muss. Sie beinhaltet unter anderem einen privaten und öffentlichen Schlüssel. Darüber hinaus legt sie den Algorithmus fest mit dem der während des Signierungsprozesses erzeugte Hashwert signiert wird. Offiziell unterstützt werden in der PDF 1.7 Spezifikation die Algorithmen Rivest–Shamir–Adleman (RSA) und Digital Signature Algorithm (DSA). Zu Beginn der Signierung wird ein Incremental Update mit den benötigten Signaturobjekten, wie in Abschnitt 2.3.1 beschrieben, erstellt und an das Original PDF Dokument angefügt. Die im Signatur-Dictionary

enthaltene `ByteRange` gibt die Byte-Werte vor über die der Hashwert[2] gebildet wird. Die für den Hashwert genutzte Hashfunktion wird durch den `SubFilter` und die verwendete PDF Version vorgegeben (siehe Tabelle 2.7). Der Signaturwert wird durch die Signierung des Hashwertes, unter Verwendung des privaten Schlüssel aus der digitalen ID, erzeugt. Falls keine Zeitstempeldaten, bspw. durch einen vertrauensvollen Zeitstempelserver, vorliegen, wird die lokale Systemzeit als Signierungszeitpunkt verwendet. Für die Signaturdaten wird je nach `SubFilter` ein PKCS#1 oder PKCS#7 Binärobjekt erzeugt, welches unter anderem den Signaturwert, öffentlichen Schlüssel und Daten zur Identität der signierenden Person enthält. Anschließend werden die Signaturdaten als Wert des `Contents`-Eintrag in hexadezimalen Form innerhalb des Signatur-Dictionarys abgelegt. Falls mehrere Signaturen zu einem PDF Dokument hinzugefügt werden, wiederholt sich der beschriebene Prozess, wobei jede neue Signatur immer über das gesamte Dokument inklusive enthaltener Signaturen gebildet wird. Davon ausgenommen sind die für den aktuellen Signierungsprozess erzeugten Signaturdaten des `Contents`-Eintrag [8, S. 8, 12].
Der beschriebene Prozess ist in Abbildung 2.14 dargestellt.

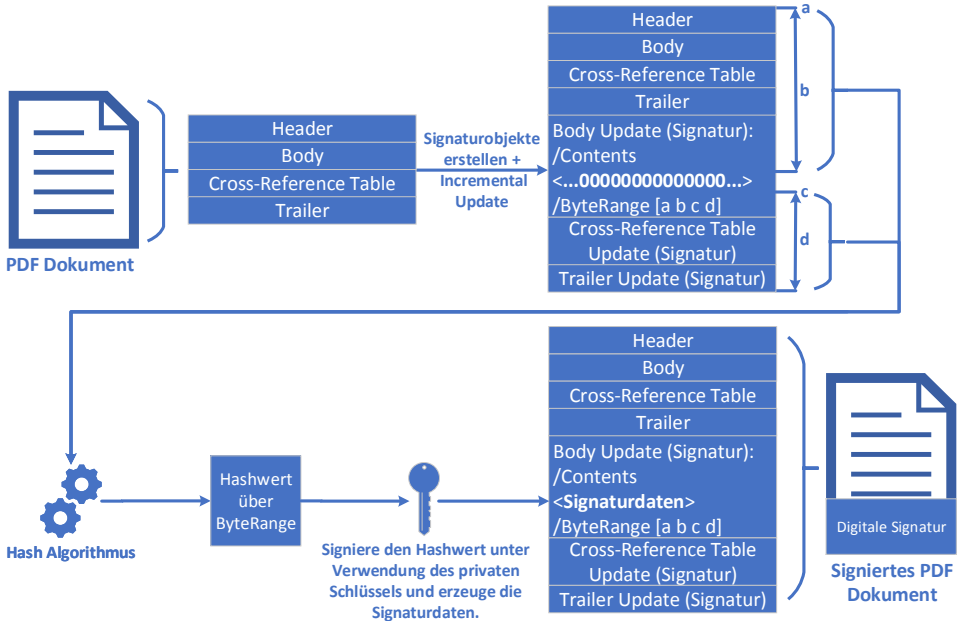

Abbildung 2.14: Schritte des PDF Signierungsprozesses.

2.3.5 Validierungsprozess

Wird ein digital signiertes PDF Dokument durch ein PDF Betrachtungsprogramm geöffnet, kann der Validierungsprozess automatisiert gestartet werden. Dabei extra-

[2]Eine (Einweg-)Hashfunktion errechnet aus einer beliebig langen Eingabe eine kryptographische Prüfsumme fester Länge, den Hashwert [20, S. 14].

hiert das Betrachtungsprogramm zunächst die Signaturdaten, welche als PKCS#1 oder PKCS#7 Binärobjekt im Wert des `Contents`-Eintrag gespeichert sind. Die Berechnung des Hashwertes aus dem Signaturwert, erfolgt unter Verwendung des enthaltenen öffentlichen Schlüssels. Parallel wird wie beim Signierungsprozess über die in der `ByteRange` hinterlegten Werte ein Hashwert erzeugt. Der erzeugte Hashwert wird mit dem berechneten Hashwert verglichen. Sind diese beiden Werte nicht identisch, kam es entweder zu einem Fehler bei der Berechnung / Erzeugung oder der Signaturwert bzw. das PDF Dokument wurden nach dem Signieren verändert. Die Folge ist eine ungültige Signatur. Stimmen die beiden Werte überein, wird das Dokument auf Incremental Updates nach dem Signieren überprüft. Sind Incremental Updates vorhanden, werden durch die DocMDP bzw. FieldMDP Methoden (siehe Abschnitt 2.3.3) die Legitimität der Updates überprüft. Sind unerlaubte Änderungen durch ein enthaltenes Incremental Update vorgenommen worden, führt dies zu einer ungültigen Signatur. Andernfalls wird die Signatur als gültig ausgegeben, falls das enthaltene Zertifikat der signierenden Person als vertrauenswürdig eingestuft wurde. Dieser Prozess wird für jede enthaltene Signatur innerhalb des PDF Dokuments durchgeführt [9, S. 33-42].
Die Einstufung als vertrauenswürdig kann abhängig vom PDF Betrachtungsprogramm über den Zertifikatsspeicher des Systems oder des Betrachtungsprogramms erfolgen. Die Abbildung 2.15 stellt den Validierungsprozess grafisch dar.

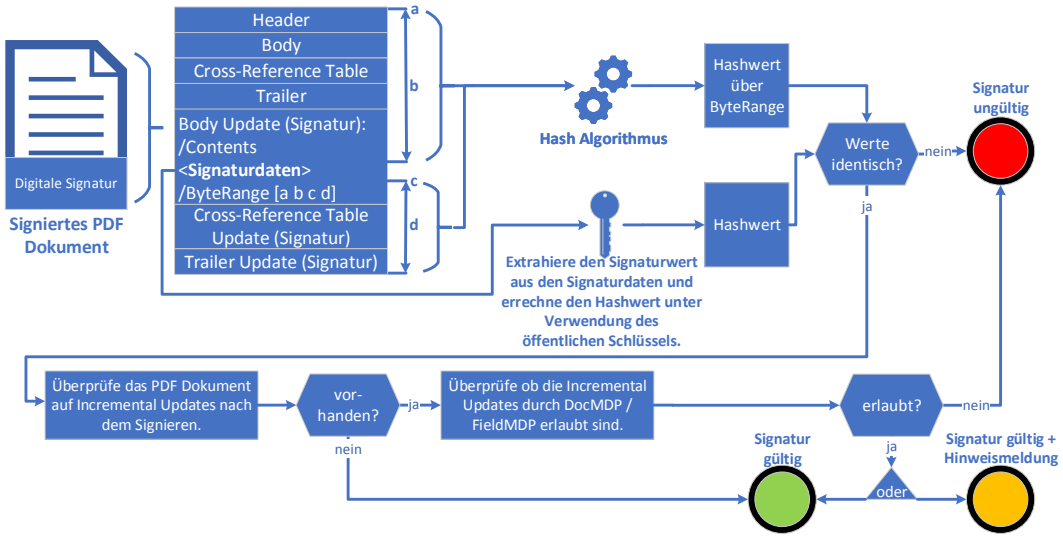

Abbildung 2.15: Schritte des PDF Validierungsprozesses.

3 Angreifermodelle

In diesem Kapitel werden die Angreifermodelle beschrieben, welche als Basis für die Angriffsklassen in Kapitel 4 dienen.

3.1 Übersicht

Für die in dieser Masterarbeit genutzten Angriffsklassen aus Kapitel 4 liegen die zwei im folgenden beschriebenen Angreifermodelle zu Grunde.
Für einen erfolgreichen Angriff ist die korrekte Darstellung des Signaturvalidierungsstatus von Bedeutung. Für viele PDF Betrachtungsprogramme lässt sich die Benutzeroberfläche, in Hinblick auf die Signaturdarstellung, in zwei Ebenen unterteilen. Die erste Ebene `User Interface (UI)-Layer 1` wird in der Regel unmittelbar nach dem Öffnen dargestellt. Sie kann grafisch, in Textform oder in Kombination visualisiert werden. Die zweite Ebene `UI-Layer 2` wird durch eine Nutzeraktion, wie bspw. ein Mausklick, geöffnet und bietet eine detaillierte Darstellung des Signaturvalidierungsstatus [35, S. 4-5].

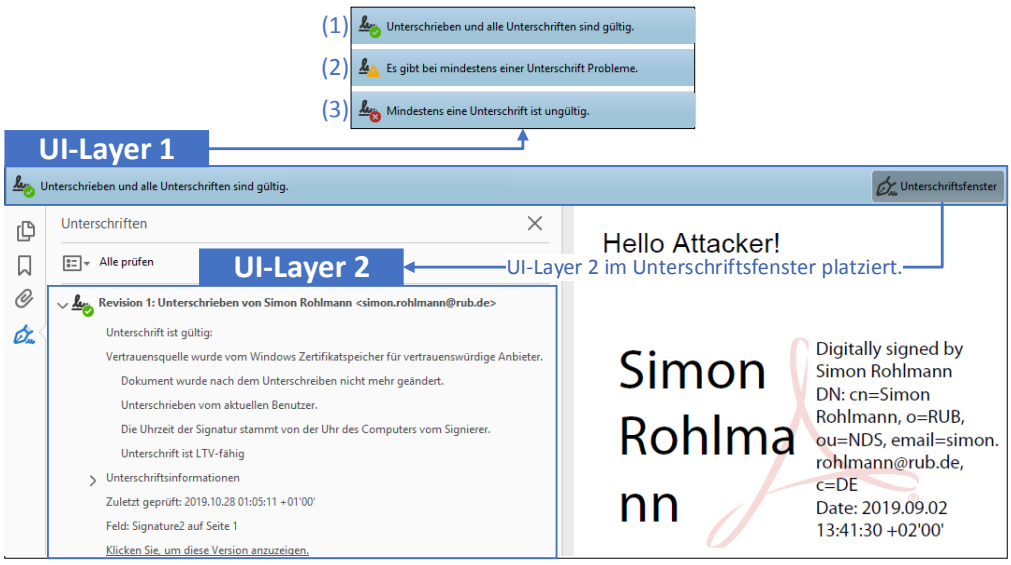

Abbildung 3.1: UI-Layer 1 und 2 am Beispiel einer Approval Signatur in Adobe Acrobat Pro 2017 [35, S. 5].

Die Abbildung 3.1 zeigt die verschiedenen Ebenen am Beispiel einer Approval Signatur in Adobe Acrobat Pro 2017. Die Ausgabe innerhalb des `UI-Layer 1` weist den Anwender unmittelbar auf eine (1) gültige, (2) nicht vertrauenswürdige oder (3) ungültige Signatur hin. Der `UI-Layer 2` wird nach einem Mausklick auf den Button „Unterschriftsfenster“ geöffnet und enthält detailliertere Signaturinformationen, wie bspw. die ausstellende Person oder die Information zu nachträglichen Änderungen am Dokument.

3.2 Angreifermodell A

Im Angreifermodell A manipuliert `Oskar` ein von `Alice` signiertes PDF Dokument. Anschließend validiert `Bob` erfolgreich das manipulierte Dokument. `Bob` kann hierbei eine beliebige Person, Gruppe oder Dienst sein, welche dem von `Alice` verwendeten Zertifikat vertraut. Ein Angriff ist nur möglich, wenn `Oskar` Zugang zu einem von `Alice` signierten PDF Dokument besitzt. `Oskar` und `Bob` haben dabei lediglich Zugriff auf den öffentlichen Schlüssel von `Alice`. In diesem Angreifermodell bildet `Oskar` den Angreifer und `Bob` das Opfer. Der beschriebene Ablauf ist in Abbildung 3.2 grafisch dargestellt.

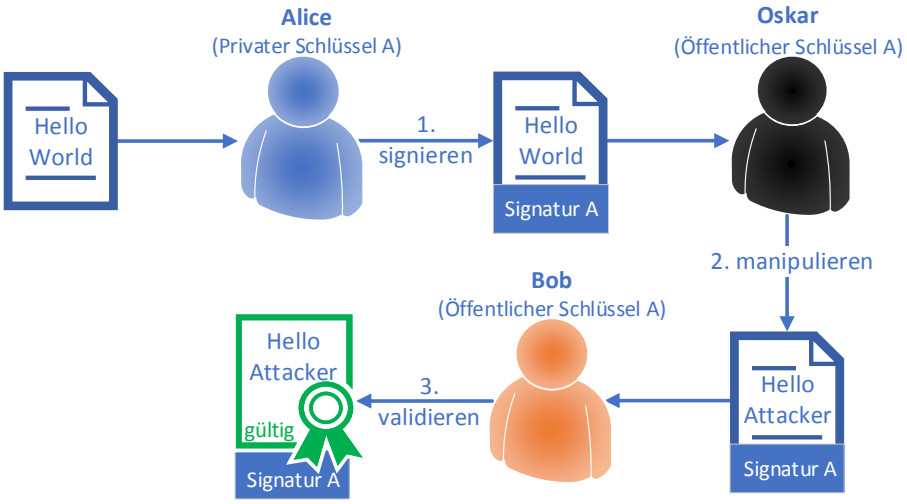

Abbildung 3.2: Angreifermodell A: Oskar manipuliert ein von Alice signiertes PDF Dokument, welches von Bob erfolgreich validiert wird.

Beispiel
Empfängt ein Mitarbeiter eine signierte Lieferantenrechnung, könnte er die enthaltene Bankverbindung durch seine eigene ersetzen und anschließend die manipulierte

Rechnung an die Finanzabteilung weiterleiten. Die Finanzabteilung veranlasst aufgrund der gültigen Signaturvalidierung die Zahlung an die manipulierte Bankverbindung.

Erfolgsbedingungen

Für einen erfolgreichen Angriff in diesem Modell müssen folgende Bedingungen erfüllt sein:

1. Beim Öffnen des manipulierten PDF Dokuments durch das PDF Betrachtungsprogramm wird dem Anwender die enthaltene Signatur als gültig angezeigt.

2. Das PDF Betrachtungsprogramm gibt den manipulierten Inhalt wieder.

Der Angriff ist eingeschränkt erfolgreich, falls zusätzlich zu Punkt 1 ein Hinweis auf Modifikationen durch das PDF Betrachtungsprogramm erfolgt. Der Angriff ist ebenfalls eingeschränkt erfolgreich, falls sich die beiden `UI-Layer 2` für das manipulierte und das unveränderte Dokument unterscheiden.

3.3 Angreifermodell B

Im Angreifermodell B erstellt `Oskar` ein PDF Dokument mit einem sichtbaren und einem versteckten Inhalt. Anschließend signiert `Alice` dieses Dokument. Es muss daher eine Vertrauensbeziehung zwischen `Oskar` und `Alice` vorliegen. Indem `Oskar` das signierte PDF Dokument manipuliert, wird der versteckte Inhalt sichtbar. Das manipulierte Dokument wird daraufhin von `Bob` erfolgreich validiert. `Bob` kann hierbei eine beliebige Person, Gruppe oder Dienst sein, welche dem von `Alice` verwendeten Zertifikat vertraut. `Oskar` und `Bob` haben dabei lediglich Zugriff auf den öffentlichen Schlüssel von `Alice`. In diesem Angreifermodell bildet `Oskar` den Angreifer und `Bob` das Opfer. Der beschriebene Ablauf ist in Abbildung 3.3 grafisch dargestellt.

Beispiel

Ein Geschäftsführer möchte einen Mitarbeiter ohne Abfindung entlassen. Hierfür erstellt er ein PDF Dokument mit einer Vereinbarung zur Gehaltserhöhung im sichtbaren Teil und einer Kündigung mit Verzicht auf eine Abfindungszahlung im versteckten Teil. Der Mitarbeiter signiert die vermeintliche Gehaltserhöhung. Anschließend manipuliert der Geschäftsführer das Dokument, sodass nur noch die vormals versteckte Kündigung sichtbar ist. Der Geschäftsführer legt die signierte Kündigung dem Mitarbeiter und dem Betriebsrat vor.

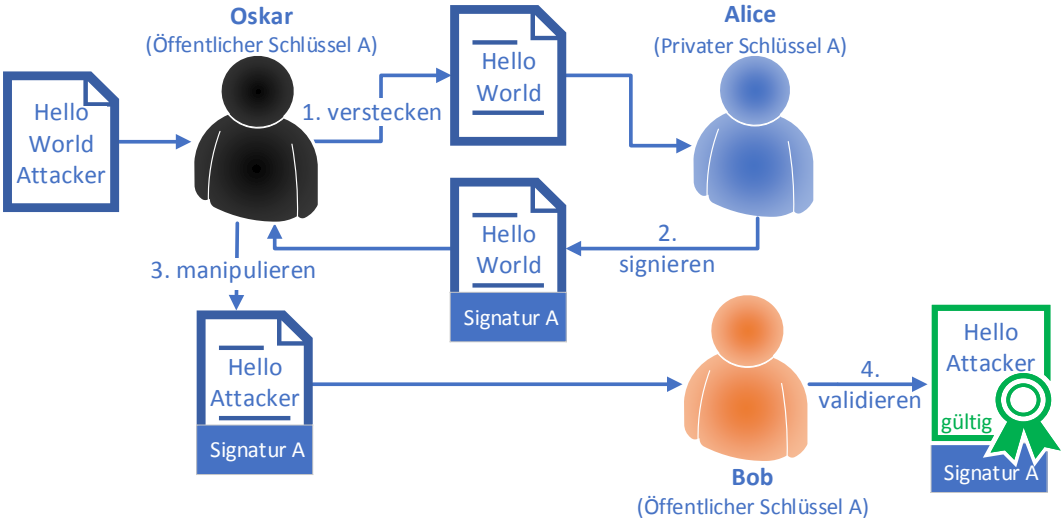

Abbildung 3.3: Darstellung der Manipulationsabfolge im Angreifermodell B.

Erfolgsbedingungen

Für einen erfolgreichen Angriff in diesem Modell müssen folgende Bedingungen erfüllt sein:

1. Beim Öffnen des manipulierten PDF Dokuments durch das PDF Betrachtungsprogramm wird dem Anwender die enthaltene Signatur als gültig angezeigt.
2. Das PDF Betrachtungsprogramm gibt den manipulierten Inhalt wieder.

Der Angriff ist eingeschränkt erfolgreich, falls zusätzlich zu Punkt 1 ein Hinweis auf Modifikationen durch das PDF Betrachtungsprogramm erfolgt. Der Angriff ist ebenfalls eingeschränkt erfolgreich, falls sich die beiden `UI-Layer 2` für das manipulierte und das unveränderte Dokument unterscheiden.

4 Angriffsklassen

Dieses Kapitel behandelt die verschiedenen Angriffsklassen auf digitale Signaturen in PDF Dokumenten. Die Angriffe Universal Signature Forgery (USF), Incremental Saving Attack (ISA) und Signature Wrapping Attack (SWA) werden im Paper „1 Trillion Dollar Refund - How To Spoof PDF Signatures“ von Mladenov et al. [35] beschrieben. Die in der Masterarbeit „Security of PDF Signatures“ von Karsten Meyer zu Selhausen [21] beschriebenen Angriffe Signature Exclusion und Incremental Update Abuse entsprechen dabei den Angriffen USF und ISA. Eine kompakte Übersicht zu den Angriffsklassen sowie den dazugehörigen Angreifermodellen ist in 4.1 dargestellt.

Tabelle 4.1: Übersicht zu den Angriffsklassen und dazugehörigen Angreifermodellen.

Angriffsklasse	**Angreifermodell**
Universal Signature Forgery	A
Incremental Saving Attack • DocMDP • FieldMDP • Font-Masking	A
Signature Wrapping Attack	A
Shadow Attack	B

4.1 Universal Signature Forgery

4.1.1 Angriffsidee

Für Angriffe auf USF-Basis gilt das in Abschnitt 3.2 beschriebene Angreifermodell A. Die Grundidee besteht darin das signierte PDF Dokument an Stellen zu manipulieren, welche für die Signaturvalidierung erforderlich sind. Dadurch kann die Signatur nicht durch die Validierungslogik verifiziert werden. Die Verarbeitungslogik des Programms findet hingegen die Signatur und zeigt selbige als gültig an. Als Ergebnis, sind Manipulationen am Inhalt des PDF Dokuments möglich [21, S. 23].

4.1.2 Manipulationsvarianten

Die digitale Signatur eines PDF Dokuments bietet verschiedene Stellen, welche sich als Ausgangspunkt einer Manipulation eignen. Das Signatur-Dictionary mit `Type`-Wert `Sig` enthält den Eintrag `SubFilter`, welcher das verwendete Encoding der Signaturwerte und Schlüsselinformationen vorgibt. Dieser Wert kann bspw. durch einen anderen gültigen bzw. ungültigen Wert ersetzt oder ganz entfernt werden. Weitere Ansatzpunkte sind der `Contents` oder `ByteRange`-Eintrag. Durch eine Manipulation am `Contents`-Wert, fehlen dem PDF Betrachtungsprogramm Teile bzw. die gesamten Signatur- und Zertifikatswerte. Dies kann zu Fehlern während der Signaturverifikation führen. Wird der `ByteRange`-Eintrag manipuliert, findet die Validierungslogik keine geeigneten Werte als Eingangsinformationen für die Hashbildung vor. Auch dies kann Fehler verursachen, die zum Abbruch der Verifikation, bei gleichzeitiger Ausgabe einer gültigen Signatur, führen können. Das angesprochene Signatur-Dictionary wird innerhalb des Field-Dictionary mit Field-Typ `Sig` referenziert. Hierfür wird der Eintrag `V` innerhalb des Field-Dictionary verwendet. Des Weiteren findet sich die Referenz auf das `Page`-Objekt, welche verwendet wird, um den sichtbaren Teil der Signatur für den Benutzer anzuzeigen. Hierfür wird der Eintrag `P` innerhalb des enthaltenen Dictionarys verwendet. Manipulationen an einem dieser Einträge können ebenfalls zu Fehlern während der Signaturverifikation führen, welche der Angreifer nutzen kann, um den Inhalt des PDF Dokuments bei gleichzeitig gültiger Signatur zu verändern [21, S. 23-24].
Die in Listing 4.2 markierten Zeilen weisen die angesprochenen Einträge aus. Die Tabelle 4.3 führt die verwendeten Manipulationen in den von Mladenov et al. veröffentlichten USF-Exploits[1] auf.

```
1  8 0 obj %Field-Dictionary einer Signatur
2  << /FT /Sig %Field-Typ = Signatur
3  /Type /Annot %Annotation
4  /Subtype /Widget %Untertyp der Annotation
5  /T (Signature1) %Name
6  /P 3 0 R %Referenz auf das Seitenobjekt
7  /Rect [150.0 400.0 450.0 480.0] %Position
8  /V 9 0 R >> %Referenz auf das Signatur-Dictionary
9  endobj
10
11 9 0 obj %Signatur-Dictionary
12 << /Type /Sig %Typ = Signatur
13 /Filter /Adobe.PPKLite %Signatur-Handler
14 /SubFilter /adbe.pkcs7.detached %Hashfunktion und Signierungsalgorithmus
15 /Name (PDF Insecurity Team) %Name des Signierers
16 /Contents <...> %Signaturdaten
17 /ByteRange [0 1633 20579 2437] >> %Durch Signatur geschützter Byte-Bereich
18 endobj
```

Listing 4.2: Beispiel für Field- und Signatur-Dictionary (Zeile 16 gekürzt).

[1] `https://pdf-insecurity.org/signature/evaluation_2018.html#download-pocs`

Tabelle 4.3: Beispiele für Manipulationen bei USF-Angriffen.

Originaleintrag	**Manipulation**
`/ByteRange [0 1633 20579 2437]`	Gesamte Zeile entfernt.
`/ByteRange [0 1633 20579 2437]`	`/ByteRange null`
`/Contents <Signaturdaten>`	Gesamte Zeile entfernt.
`/Contents <Signaturdaten>`	`/Contents`
`/Contents <Signaturdaten>`	`/Contents <>`
`/Contents <Signaturdaten>`	`/Contents null`
`/Contents <Signaturdaten>`	`/Contents <00>`

4.2 Incremental Saving Attack

4.2.1 Angriffsidee

Für die ISA-Angriffe gilt das in Abschnitt 3.2 beschriebene Angreifermodell A. Nachdem eine Signatur zu einem bestehenden PDF Dokument hinzugefügt wurde, lassen sich Änderungen per Incremental Update durchführen. Veränderungen, die explizit den Inhalt bzw. Seiten verändern, sind nicht gestattet und sollen durch das PDF Betrachtungsprogramm mit einer invaliden Signatur geahndet werden. Im Kontext von ISA sollen die Updates dabei so angepasst werden, dass sie die Überprüfung durch das Betrachtungsprogramm überstehen und Änderungen am Inhalt des Dokuments herbeiführen. Der signierte Bereich des PDF Dokuments bleibt bei diesem Angriff unverändert [21, S. 25].
Dabei reagieren manche Programme generell bei jeder Art von Incremental Update mit einem Hinweis an den Benutzer, dass es zu Modifikationen nach der Signierung gekommen ist. Eine detaillierte Unterscheidung zum Verhalten der Betrachtungsprogramme ist in Abschnitt 6.2.2 dargestellt.

4.2.2 Manipulationsvarianten

Die verschiedenen Varianten beinhalten Manipulationen in einem oder mehreren Bereichen des Incremental Updates. Das Entfernen oder nur teilweise Hinzufügen von Objekten oder Werten kann dazu führen, dass der Inhalt des signierten PDF Dokuments verändert werden kann. Darüber hinaus limitieren manche PDF Betrachtungsprogramme Incremental Updates innerhalb eines signierten PDF Dokuments kaum oder gar nicht. Speziell Restriktionen, die durch gesetzte Transformationsparameter festgelegt wurden, werden nicht immer ausreichend durch das Betrachtungsprogramm geprüft (siehe Abschnitt 6.4.1 und 6.4.2). Das Ziel eines ISA-Angriffs ist

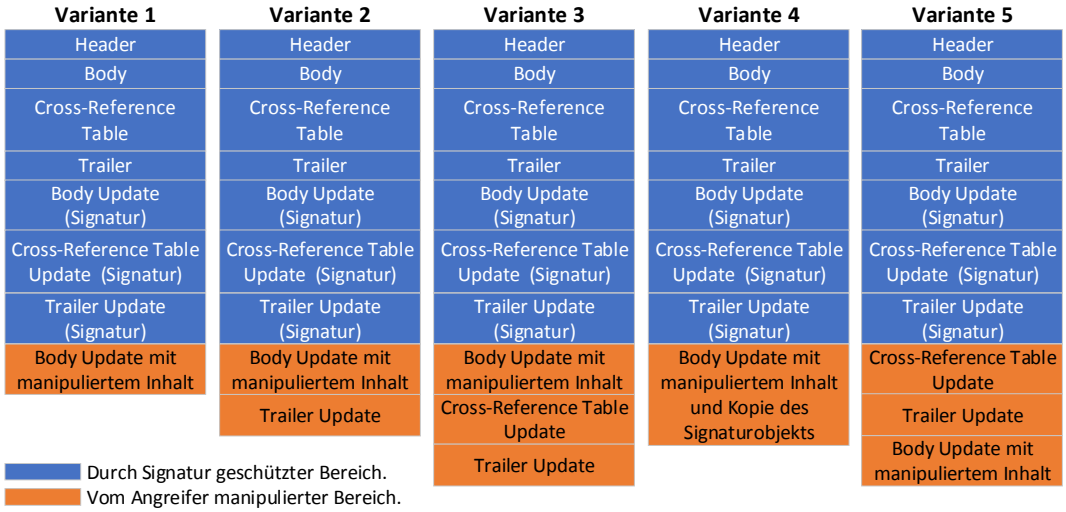

Abbildung 4.4: PDF Dokumentaufbau nach Anwendung der verschiedenen ISA-Varianten [35, S. 5].

die Manipulation am Inhalt des signierten PDF Dokuments, unter Verwendung eines Incremental Updates und gleichzeitig gültiger Signaturvalidierung. Die Abbildung 4.4 stellt verschiedene Möglichkeiten von Manipulationen auf ISA-Basis dar. Im Folgenden werden die Varianten 1-4 [21, S. 25-26] und eine neue Variante 5, die im Rahmen dieser Masterarbeit entstanden ist, vorgestellt.

Variante 1

Bei der ersten Variante wird lediglich ein Body Update an das vorhandene Dokument angehängt. Der manipulierte Body kann bspw. einen komplett neuen Objektpfad für das PDF Dokument beinhalten oder auch nur einzelne Objekte, welche den direkt sichtbaren Inhalt des PDF Dokuments enthalten. Diese Variante ist erfolgreich, falls das Betrachtungsprogramm den Inhalt wiedergibt, aber das unvollständige Incremental Update nicht von der Validierungslogik als unerlaubt eingestuft wird.

Variante 2

Bei der zweiten Variante wird ein Body und Trailer Update an das signierte Dokument angehängt. Der manipulierte Body kann bspw. einen komplett neuen Objektpfad für das PDF Dokument beinhalten oder auch nur einzelne Objekte, welche den direkt sichtbaren Inhalt des PDF Dokuments enthalten. Falls in Variante 1 der manipulierte Inhalt im Body Update nicht durch das PDF Betrachtungsprogramm dargestellt wird, kann das Trailer Update diesen Missstand ausgleichen. Darüber

hinaus muss das unvollständige Incremental Update von der Validierungslogik nicht als unerlaubt eingestuft werden.

Variante 3
Bei der dritten Variante wird ein Body, eine Cross-Reference Table und ein Trailer Update an das vorhandene Dokument angehängt. Der manipulierte Body kann bspw. einen komplett neuen Objektpfad für das PDF Dokument beinhalten oder auch nur einzelne Objekte, welche den direkt sichtbaren Inhalt des PDF Dokuments enthalten. Im Bereich der Cross-Reference Table sind verschiedene Änderungen denkbar. So kann der Bereich zwar mit `xref` eingeleitet werden, aber die eigentliche Tabelle leer sein und verweist somit nicht auf die verwendeten Objekte. Der Validierungslogik wird auf diese Weise vermittelt, dass keine neuen Objekte enthalten sind und das Incremental Update daher als harmlos eingestuft werden soll. Im zweiten Fall wird eine vollständige Tabelle angefügt, welche alle enthaltenen Objekte im Body korrekt referenziert.

Variante 4
Die vierte Variante enthält neben dem Body Update weitere Signaturobjekte. So werden neben dem veränderten Inhalt im Body-Bereich auch die im Listing 4.2 aufgeführten indirekten Objekte (Field- und Signatur-Dictionary) als Update an das PDF Dokument angehängt. Falls eine Validierungslogik Incremental Updates für signierte PDF Dokumente nur dann genehmigt, wenn das Update ebenfalls über eine Signatur verfügt, kann diese Restriktion durch die Variante 4 umgangen werden. Die erfolgreiche Manipulation ist möglich, da nicht zwingend vorausgesetzt wird, dass die Signatur das gesamte Dokument schützen muss.

Variante 5
Die fünfte Variante enthält ein Body, eine Cross-Reference Table und ein Trailer Update, jeweils in vollständiger und gültiger Ausprägung. Allerdings wird hierbei die Reihenfolge der Updates vertauscht, sodass die Cross-Reference Table auf Objekte nach `%%EOF` referenziert. Die Idee dieser Variante besteht darin, dass die Validierungslogik eines PDF Betrachtungsprogramms ein Incremental Update nur bis `%%EOF` auf Legitimität überprüft, die Verarbeitungslogik aber auch Objekte nach dem Trailer akzeptiert und darstellt.

4.2.3 Spezielle Angriffsziele

Neben den ISA-Angriffen auf beliebige Stellen des PDF Inhalts, werden in dieser Masterarbeit auch ISA-Angriffe auf spezielle Ziele des PDF Dokuments behandelt. Dies sind zum einen Manipulationen, welche durch die Restriktionen der Transformationsmethoden nicht gestattet sind und zum anderen Manipulationen an enthaltenen

Schriftarten des PDF Dokuments. Diese Angriffe lassen sich ebenfalls mit den im Abschnitt 4.2.2 beschriebenen Varianten durchführen.

DocMDP

ISA-Angriffe auf die DocMDP Transformationsmethode setzen ein PDF Dokument mit Certification Signatur und gesetztem DocMDP Parameter `P 1` bzw. `P 2` voraus. Bei einem Angriff wird ein Incremental Update erzeugt, welches jeweils gegen die in Tabelle 2.10 beschriebenen Restriktionen verstößt. Dies kann bspw. eine Annotation in Form einer Bilddatei sein, welche den eigentlichen Inhalt des signierten PDF Dokuments überdeckt oder eine Inhaltsmanipulation an einem Formularfeld.

FieldMDP

Für ISA-Angriffe auf die FieldMDP Transformationsmethode muss das PDF Dokument mindestens über eine Approval Signatur verfügen, welche über einen gesetzten FieldMDP Parameter mindestens ein Formularfeld schützt. Die verschiedenen Parameterausprägungen sind in Tabelle 2.12 beschrieben. Bei einem Angriff wird ein Incremental Update erzeugt, welches den Inhalt eines geschützten Formularfelds manipuliert.

Font-Masking

Font-Masking im PDF Kontext wurde in Arbeiten von Markwood et al. [18, S. 836] und Mladenov et al. [35, S. 12] thematisiert.
Die Angriffe lassen sich für beide Signaturtypen durchführen. Das Ziel dieser Angriffe ist es, den Inhalt eines signierten PDF Dokuments zu maskieren. Dabei wird nicht der eigentliche Inhalt verändert, sondern über eine Manipulation Einfluss auf die Darstellung genommen. Auf diese Weise lassen sich Buchstaben, Zahlen oder sonstige Zeichen vertauschen oder ausblenden. Eine Einschränkung dieses Angriffes ist es, dass eine manipulierte Schriftart nicht nur ein bestimmtes Zeichen im Text verändert, sondern sich die Darstellung für sämtliche Zeichen des gleichen Typs ändert. Um den Angriff durchzuführen, wird zunächst eine Schriftart manipuliert und anschließend durch ein Incremental Update eingespielt. Dabei wird die gleiche Objektnummer, wie für die enthaltene Schriftart verwendet. Die Verarbeitungslogik des PDF Betrachtungsprogramms erwartet dabei für die manipulierte Schriftart den gleichen Namen, wie für die bereits enthaltene Schriftart.

4.3 Signature Wrapping Attack

4.3.1 Angriffsidee

Für Angriffe auf SWA-Basis gilt das in Abschnitt 3.2 beschriebene Angreifermodell A. Der durch eine digitale Signatur geschützte Bereich wird durch die Byte-Werte der `ByteRange` bestimmt. Der Wert des `Contents`-Eintrag innerhalb des Signatur-Dictionary wird hierbei immer ausgeschlossen. Dies ist zwingend notwendig, da es unter anderem den signierten Hashwert über das Dokument enthält, welcher verständlicherweise noch nicht zum Zeitpunkt des Erstellens der Signaturdaten verfügbar ist. Die `ByteRange` teilt das Dokument in zwei Teile auf und lässt zwischen diesen Teilen genug Platz, um die Signaturdaten nachträglich einzufügen. Die Idee der Signature Wrapping Attack basiert darauf manipulierte Objekte zwischen diesen Teilen zu platzieren und anschließend den `c`-Wert innerhalb einer neuen `ByteRange` um die hinzugekommenen Byte-Werte zu erhöhen. So sollen die Eingabedaten der Signaturberechnung gleich bleiben, während das PDF Betrachtungsprogramm die manipulierten Objekte verarbeitet und dem Opfer präsentiert. Alternativ zu der Möglichkeit die Signature Wrapping Attack zwischen den beiden Dokumententeilen zu platzieren, können die manipulierten Objekte auch oberhalb des Original-Headers eingefügt werden. Hierfür werden alle Werte der neuen `ByteRange` entsprechend angepasst [35, S. 6].

4.3.2 Manipulationsvarianten

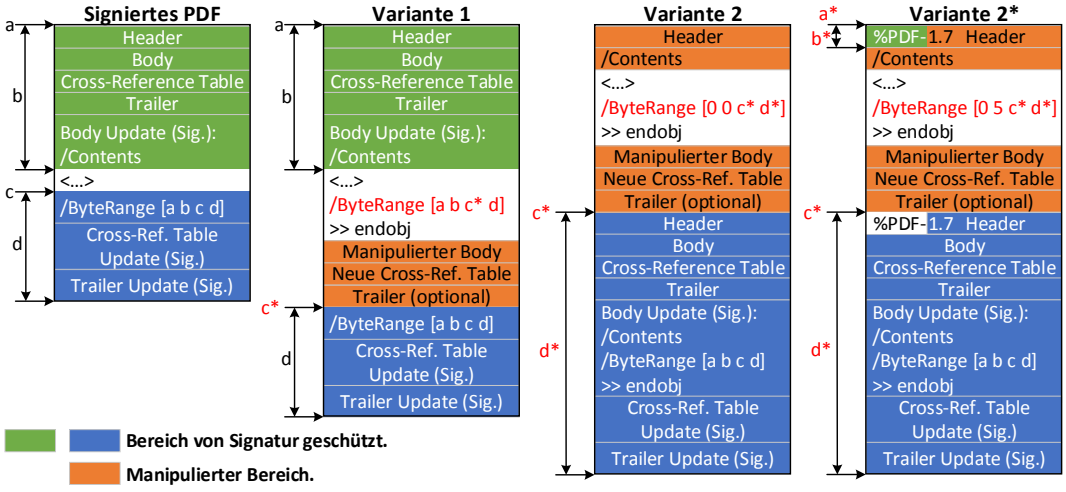

Abbildung 4.5: Vergleich der Manipulationsvarianten auf SWA-Basis [35, S. 7].

Das detaillierte Vorgehen für die in der Angriffsidee beschriebenen beiden Möglichkeiten zur Umsetzung eines SWA-Angriffs werden sowohl in Abbildung 4.5 als auch im Folgenden dargestellt.

Variante 1

Die Variante 1 basiert darauf die manipulierten Objekte zwischen den beiden beschriebenen Dokumentteilen zu platzieren. Hierfür wurden folgende Schritte von Mladenov et al. [35, S. 6-7] beschrieben:

1. (Optional) Die eingefügten Nullen des Paddings im Wert des `Contents`-Eintrag können teilweise entfernt werden, um anschließend den freigewordenen Platz für manipulierte Objekte zu nutzen.
2. Oberhalb der Original-`ByteRange` wird eine neue `ByteRange` erzeugt. Hierfür werden die Werte `a`, `b`, `d` übernommen und der `c`-Wert um den Byte-Wert erhöht, welcher für die Manipulation benötigt wird. Der erhöhte `c`-Wert wird nachfolgend `c*` genannt.
3. Es wird eine neue Cross-Reference Table für die manipulierten Objekte eingefügt. Da sich die Referenzierung des Original-Trailers nicht ändern lässt, muss die neue Tabelle auf Byte-Höhe der Original-Cross-Reference Table liegen.
4. Die manipulierten Objekte werden eingefügt. Falls der erste Schritt ausgelassen wurde, müssen die manipulierten Objekte nach der neuen Cross-Reference Table eingefügt werden. Andernfalls können die manipulierten Objekte auch oberhalb der neuen Cross-Reference Table platziert werden.
5. (Optional) Damit das Dokument korrekt geöffnet und die Manipulation erfolgreich umgesetzt wird, benötigen einige PDF Betrachtungsprogramme unterhalb der neuen Cross-Reference Table den Original-Trailer.
6. Der im Original-Dokument ab Wert `c` befindliche Teil wird an die Position `c*` verschoben. Optional besteht die Möglichkeit diesen Teil des Dokuments in ein `Stream`-Objekt zu kapseln.

Variante 2

In der zweiten Variante befinden sich die manipulierten Objekte nicht zwischen den beiden Dokumentteilen, sondern werden oberhalb des Dokuments eingefügt. Darüber hinaus wird auch der Wert des `Contents`-Eintrag in den manipulierten Body verschoben. Hieraus ergibt sich für Schritt 2 die Änderung, dass alle Werte der `ByteRange` angepasst werden müssen. `a*` und `b*` erhalten den Wert 0. `c*` erhält den Byte-Wert direkt nach den einfügten Manipulationen, während sich `d*` aus dem Summenwert von `b` + `d` bildet. In Schritt 6 werden die beiden Original-Teile des Dokuments ab Byte-Wert `c*` platziert. Der Wert des `Contents`-Eintrag wird dabei ausgelassen, wodurch sich die Lücke zwischen beiden Teilen schließt. Falls ein PDF

Betrachtungsprogramm für `b` einen Wert größer als 0 verlangt, kann dies durch die Teilintegration des PDF Headers umgangen werden. Jedes reguläre PDF Dokument beginnt mit der Zeichenkette „%PDF-“, gefolgt von der PDF Versionsnummer. Die Zeichenkette „%PDF-“ liegt also immer im Byte-Bereich von 0 bis 5. Somit wird für `b` der Wert 5 gewählt, wodurch ein Teil des Headers aus dem manipulierten Bereich in die Signaturberechnung einfließt. Dieser Teil des Headers ist ebenfalls im Original-Header zu finden, wodurch die Berechnung einen korrekten Signaturwert liefert [35, S. 7].

4.4 Shadow Attack

4.4.1 Angriffsidee

Für Shadow Attacks gilt das in Abschnitt 3.3 beschriebene Angreifermodell B. Die Angriffe auf Shadow Attack Basis wurden von Vladislav Mladenov und Christian Mainka während der Bearbeitung dieser Masterarbeit entwickelt und in die Untersuchung aufgenommen. Die Grundidee ist es zwei verschiedene Inhalte in einem PDF Dokument unterzubringen. Das PDF Betrachtungsprogramm soll beim Öffnen des Dokuments der signierenden Person aber nur den Inhalt `X` anzeigen und den zweiten Inhalt `Y` ausblenden. Das Opfer unterschreibt das PDF Dokument anschließend auf Grundlage des Inhalts `X`. Im nächsten Schritt blendet der Angreifer den Inhalt `X` aus und gleichzeitig den Inhalt `Y` ein.

4.4.2 Manipulationsvarianten

Diese Angriffsklasse lässt sich in zwei Varianten aufteilen:

Variante 1

In Variante 1 werden während der Erstellung des PDF Dokuments weitere Objekte hinterlegt, aber zunächst nicht direkt angesprochen. So ist im Quelltext des Dokuments ein regulären Objektpfad hinterlegt und korrekt über die Cross-Reference Table referenziert. Dieser Objektpfad enthält Objekte, die das PDF Betrachtungsprogramm anweisen bspw. den Inhalt „Hello World“ auszugeben. Neben diesem regulären Objektpfad befindet sich ein weiterer Pfad mit Objekten im Dokument, die das Betrachtungsprogramm anweisen bspw. den Inhalt „Hello Attacker“ auszugeben. Der zweite Objektpfad wird allerdings nur teilweise über die Cross-Reference Table referenziert. So ist der Byte-Wert innerhalb der Tabelle korrekt angegeben, aber die Objektnummer des Tabelleneintrags entspricht nicht der Nummer des adressierten Objekts. Dadurch werden die Objekte im zweiten Objektpfad korrekt referenziert, aber nicht durch das Betrachtungsprogramm angesprochen, da die referenzierten

Objektnummern im Dokument nicht genutzt werden. In der Abbildung 4.6 wird die Dokumentstruktur vor dem Signieren dargestellt.

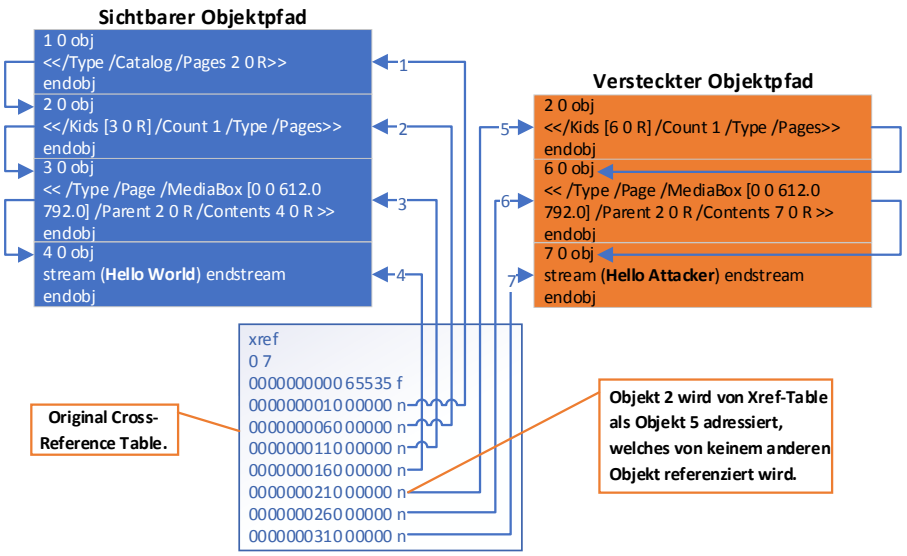

Abbildung 4.6: Shadow Attack Variante 1: Dokumentstruktur vor dem Signieren.

Nachdem das PDF Dokument mit dem für das Opfer sichtbaren Inhalt „Hello World“ signiert wurde, wird ein Update der Cross-Reference Table und des Trailers eingefügt. Innerhalb der Tabelle wird nun der zweite Objektpfad referenziert und das Betrachtungsprogramm zeigt nun den vorher versteckten Inhalt „Hello Attacker“ an. Gleichzeitig wird dabei die Referenz zum ersten Objektpfad ersetzt und somit der vormals angezeigte Inhalt „Hello World“ ausgeblendet. Die Abbildung 4.7 zeigt die Struktur nach Signierung und Manipulation des Dokuments.

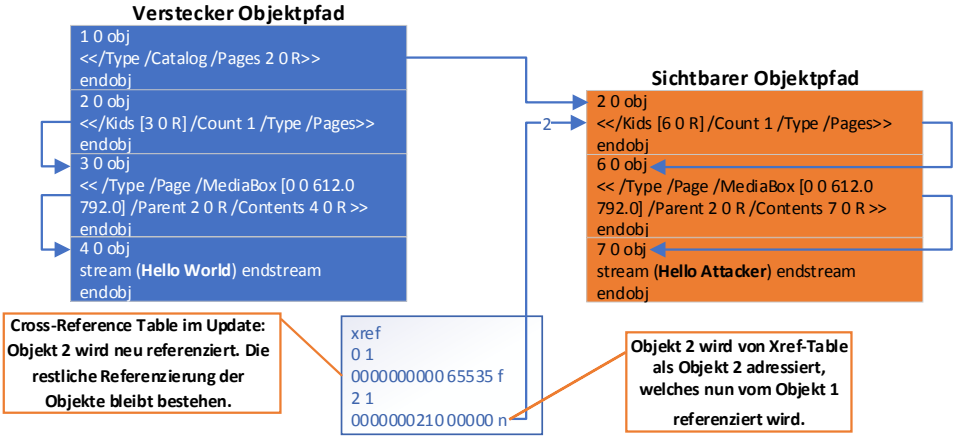

Abbildung 4.7: Shadow Attack Variante 1: Struktur nachdem das Dokument signiert und der zweite Objektpfad referenziert wurde.

Variante 2

In der zweiten Variante wird bei Erstellung der zu versteckende Inhalt regulär im Quelltext eingefügt und korrekt durch die Cross-Reference Table referenziert. Damit das PDF Betrachtungsprogramm den in diesem Beispiel verwendeten Inhalt „Hello Attacker“ nicht für das Opfer sichtbar macht, wird ein weiteres Objekt über diesen Inhalt platziert. Dies kann bspw. eine Bilddatei mit dem Inhalt „Hello World“ sein. Der eigentliche Inhalt des Dokuments wird somit hinter dieser Bilddatei versteckt. Nachdem das Opfer das PDF Dokument signiert hat, wird durch den Angreifer ein Incremental Update erzeugt, welches die Bilddatei überschreibt und somit den eigentlichen Inhalt „Hello Attacker“ des PDF Dokuments sichtbar macht. In Abbildung 4.8 wird die Variante 2 grafisch dargestellt. Der blau eingefärbte Teil stellt dabei die Bilddatei dar, welche über den eigentlichen Inhalt des PDF Dokuments platziert wird.

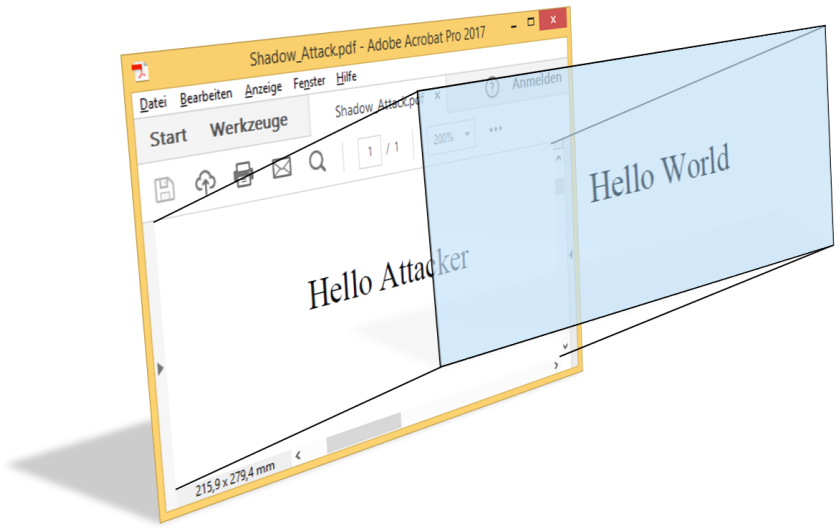

Abbildung 4.8: Shadow Attack Variante 2: Objektüberlagerung durch Bilddatei.

5 Programmentwicklung

Während der Masterarbeit sind über 100 PDF Dokumente mit verschiedenen Manipulationsvarianten entstanden. Um die Auswirkungen der Manipulationen in allen PDF Betrachtungsprogrammen zu überprüfen, muss jedes manipulierte PDF Dokument in jedem Betrachtungsprogramm gestartet werden, das Ergebnis der Verifikation sowie die Darstellung des Inhaltes evaluiert werden und zu Dokumentationszwecken ein Screenshot angefertigt werden. Insgesamt resultierten daraus über 1800 Screenshots. Um das Anfertigen und Auswerten dieser Screenshots zu automatisieren, ist im Rahmen dieser Masterarbeit das Programm *PDF Tester* entstanden. Es bietet Funktionen zum automatisierten Öffnen von PDF Dokumenten in verschiedenen Betrachtungsprogrammen, Anfertigen von Screenshots und unterstützt den Anwender bei der Ergebnisauswertung. Die Resultate lassen sich per Screenshot-Vergleich auf Pixelebene oder durch eine Texterkennung auswerten. Das Programm wurde in der Programmiersprache C# mit dem .NET Framework 4.7.2 unter Visual Studio 2019 entwickelt. Für die Bedienung des Programms wurde eine grafische Oberfläche auf Basis von Windows Forms [25] gestaltet (siehe Abbildungen im Anhang A.3).

5.1 Funktionsumfang

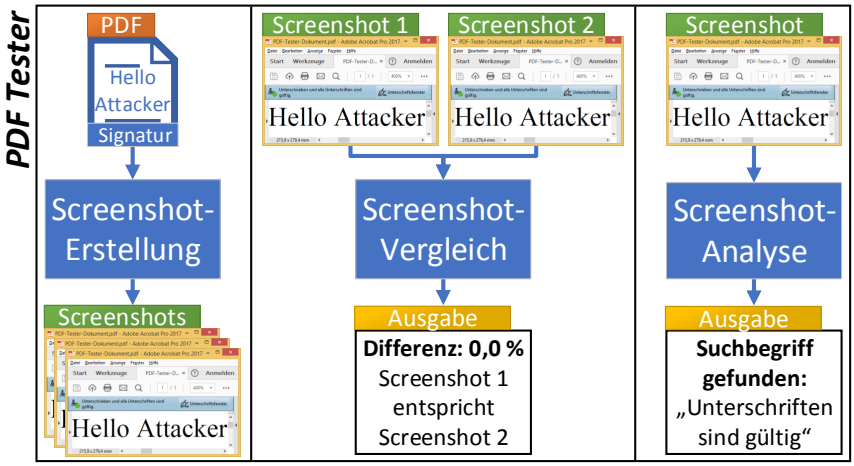

Abbildung 5.1: Kompakte Darstellung der drei enthaltenen Grundfunktionen.

In Abbildung 5.1 werden die drei Grundfunktionen des Programms kompakt dargestellt. Um Manipulationsauswirkungen in den einzelnen PDF Betrachtungsprogrammen zu dokumentieren, wird die Screenshot-Erstellung verwendet. Der Anwender teilt dem Programm die Pfade zu den PDF Dateien und Betrachtungsprogrammen mit und erhält als Ausgabe jeweils einen Screenshot je geöffneten Dokuments in jedem Betrachtungsprogramm. Um die Manipulationsauswirkungen zu verifizieren, wird der Screenshot-Vergleich verwendet. Hierbei werden zwei Screenshots miteinander verglichen. Dabei dient ein Screenshot als Referenz und muss das unmanipulierte PDF Dokument mit gültiger Signatur enthalten. Als Ausgabewert erhält der Anwender die Differenz beider Screenshots. Liegen keine geeigneten Referenz-Screenshots vor, kann alternativ die Screenshot-Analyse verwendet werden. Als Eingabe dienen die zu analysierenden Screenshots sowie darin zu suchende Begriffe. Das Programm extrahiert den gesamten Text aus den Bilddateien und vergleicht ihn mit den Suchbegriffen. Der Anwender wird über gefundene Suchbegriffe innerhalb der Screenshots informiert und erhält den extrahierten Inhalt als Textdatei. In den nachfolgenden Abschnitten wird detaillierter auf den Aufbau und Funktionsweise der einzelnen Programmteile eingegangen.

5.2 Programmaufbau

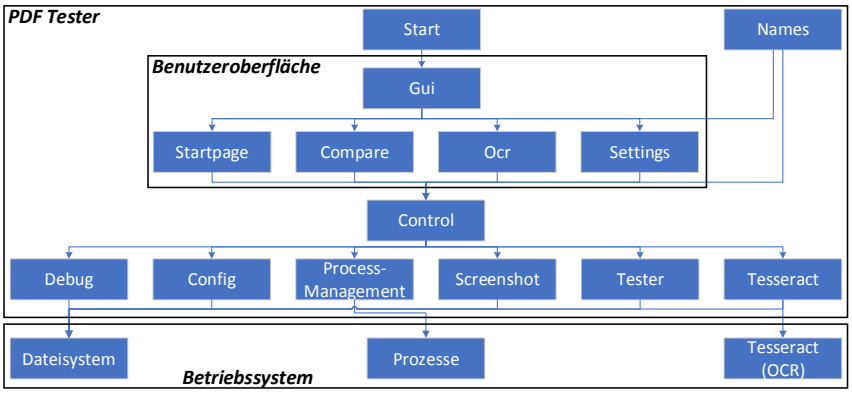

Abbildung 5.2: Diagramm der enthaltenen Klassen und Verbindungen untereinander sowie Darstellung der Schnittstellen zum Betriebssystem.

In Abbildung 5.2 sind die verwendeten Klassen und Zugriffe untereinander bzw. auf das Betriebssystem dargestellt. Nach Programmstart wird die grafische Oberfläche geöffnet und ermöglicht den Zugriff auf die Startseite, Vergleichsseite, OCR-Seite sowie zu den Einstellungen. Die Klasse „Gui“ bildet hierbei den Multiple Document Interface (MDI)-Container [24], welcher den Rahmen für die Form-Seiten der Oberfläche bildet. Über die Klasse „Control“ wird die Kommunikation zwischen den Klassen der grafischen Oberfläche und der Verarbeitungslogik hergestellt. Einzelne Programmkomponenten können auf diese Weise leicht ausgetauscht werden, ohne

Änderungen an anderen Klassen durchführen zu müssen. Die statische Klasse „Names“ wird innerhalb der grafischen Oberfläche und der „Control“-Klasse eingesetzt. Sie enthält zentrale Konfigurationsangaben, wie Dateinamen, Pfadangaben und zu verwendende Standardwerte. In den folgenden Abschnitten wird detaillierter auf das Zusammenspiel der verschiedenen Klassen eingegangen.

5.3 Programmstart

Dem Programm übergebene Pfade, Listen und Einstellungen werden in Konfigurationsdateien auf dem Rechner des Anwenders abgelegt. Hierfür dient das Verzeichnis `C:\Users\%username%\AppData\Local\PdfTester`, wobei der Platzhalter `%username%` durch den Benutzernamen des aktuellen Anwenders ersetzt wird. Bei jedem Programmstart wird überprüft, ob das genannte Verzeichnis vorhanden ist und die Dateien aus Tabelle 5.8 bereits existieren. Ist dies nicht der Fall, wird der Ordner und die entsprechenden Dateien mit Standardwerten angelegt. Falls bereits vom Anwender Variablen hinterlegt wurden, werden sie im Rahmen der Startsequenz in die entsprechenden Textboxen der Oberfläche importiert.

5.4 Screenshot-Erstellung

Das entwickelte Programm *PDF Tester* dient vor allem dazu die Screenshot-Erstellung zu automatisieren und so eine angemessene Dokumentation der Manipulationsergebnisse zu gewährleisten. Auf der Startseite lässt sich über eine Textbox der Pfad zu den einzelnen PDF Betrachtungsprogrammen hinzufügen. Darüber hinaus ist die Eingabe von optionalen Namen und Zeitangaben möglich. Jede Eingabeoption wird dabei durch ein Semikolon getrennt. Der Aufbau einer Zeile sieht bspw. wie folgt aus:

```
C:\Program Files (x86)\Adobe\Acrobat 2017\Acrobat.exe;Adobe;10
```

Die in diesem Beispiel verwendete optionale Namensgebung „Adobe“ im mittleren Teil wird für die Erstellung des Screenshot-Ordners sowie für den Dateinamen der Bilddatei genutzt. Falls an dieser Stelle kein Name eingetragen wurde, wird die Bezeichnung der entsprechenden Anwendungsdatei verwendet. Für das im Beispiel verwendete PDF Betrachtungsprogramm also „Acrobat“. Im dritten Teil der Eingabezeile legt der Anwender die Wartezeit in Sekunden für den Programmaufbau fest. D.h. erst nach dieser Zeit wird der Screenshot durch den *PDF Tester* angelegt. Diese Zeit kann individuell für jedes PDF Betrachtungsprogramm festgelegt werden. Wird dieser Wert nicht gesetzt, dient der unter „Einstellungen“ festgelegte Standardwert als Basiswert. Im rechten Teil der Oberfläche lässt sich die Pfadangabe zu den PDF Dateien hinzufügen, dabei wird der angegebene Pfad sowie vorhandene Unterordner

nach PDF Dateien durchsucht. Die zweite Pfadangabe dient zur Angabe des übergeordneten Verzeichnisses für die zu erstellenden Screenshots. Durch die Schaltfläche „PDF Tester starten“ wird die Verarbeitungslogik ausgeführt. Zunächst wird die eingegebene Programmliste sowie die übergebenen Pfadangaben in die dafür vorgesehenen Konfigurationsdateien gespeichert (siehe Tabelle 5.8). Wobei leere Angaben zu einem entsprechenden Hinweis an den Anwender und zum Abbruch der Verarbeitung führen. Bei korrekten Angaben wird unter Verwendung von zwei „foreach“-Schleifen die Dateipfade zu den gefundenen PDF Dokumenten sowie die Programmliste abgearbeitet. Für jedes Dokument wird jeweils ein Screenshot in jedem Betrachtungsprogramm erstellt. Damit die grafische Oberfläche weiterhin funktional bleibt, wird dieser Vorgang asynchron unter Verwendung der „Backgroundworker“-Klasse [23] durchgeführt. Nach jeder Screenshot-Erstellung wird der Fortschrittsbalken inkrementiert. Bei auftretenden Fehlern, während des Verarbeitungsvorgangs, wird die entsprechende Fehlermeldung unter Verwendung eines aktuellen Zeitstempels, der betroffenen Funktion und der genauen Fehlerbeschreibung in die debug.txt-Datei geschrieben. Eine Hinweismeldung informiert den Anwender über neue Einträge in der genannten Datei. Das Flussdiagramm in Abbildung 5.3 zeigt den beschriebenen Prozessablauf.

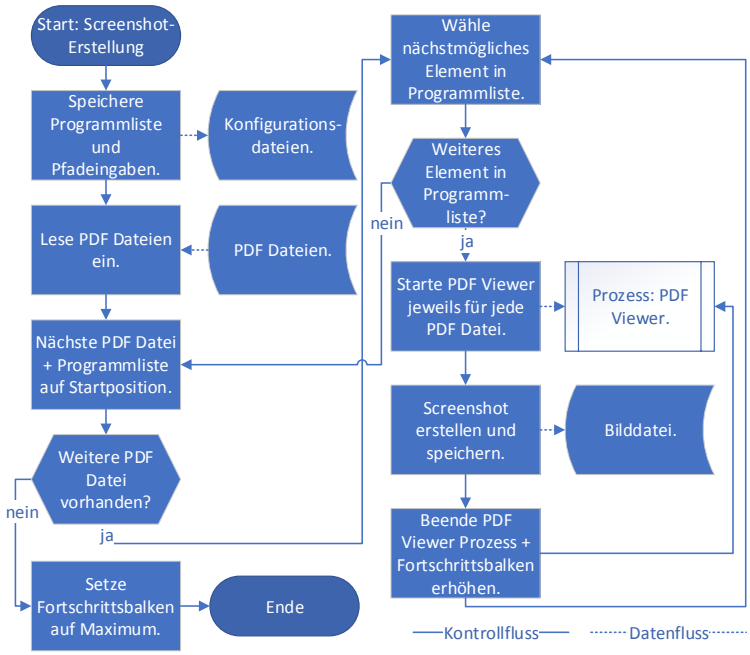

Abbildung 5.3: Flussdiagramm zur Funktion der Screenshot-Erstellung. Als Eingabe erhält das Programm die Pfadangaben zu PDF Dateien und Betrachtungsprogrammen.

5.5 Screenshot-Vergleich

Neben der Screenshot-Erstellung in Abschnitt 5.4, bietet der *PDF Tester* die Möglichkeit zum Vergleichen von Screenshots auf Pixelebene. Hierfür ist es nötig, dass im Vorfeld für jedes PDF Betrachtungsprogramm ein Screenshot von einem PDF Dokument mit valider Signatur vorliegt. Über die grafische Oberfläche unter Menüpunkt „Vergleich“ hat der Anwender die Möglichkeit Pfadangaben zu Verzeichnissen mit validen und zu testenden Screenshots anzugeben. Nachdem der Screenshot-Vergleich gestartet wurde, werden beide Pfadangaben und enthaltende Unterordner nach Bilddateien durchsucht. Das Flussdiagramm in Abbildung 5.5 stellt den Programmablauf grafisch dar. Für jeden zu testenden Screenshot wird anhand des im Dateinamen vorhandenen Programmnamens, bspw. “Adobe“, ein passender validen Screenshot gesucht, welcher den gleichen Programmnamen im Dateinamen trägt. Kann ein passender valider Screenshot ermittelt werden, wird die Differenz zwischen den beiden Bilddateien in Prozent, wie im Listing 5.4 dargestellt, ermittelt.

```
1  /*
2  Durchlaufe jedes Pixel der Bilddateien bis zur angegeben Maximalhöhe
3  und bis zur Breite der ersten Bilddatei.
4  Berechne die Differenz zwischen beiden Bilddateien in Prozent.
5  */
6  for (int y = 0; y < intMaxHeight; y++)
7  {
8    for (int x = 0; x < img1.Width; x++)
9    {
10     Color pixel1 = img1.GetPixel(x, y);
11     Color pixel2 = img2.GetPixel(x, y);
12
13     diff = diff + Math.Abs(pixel1.R - pixel2.R);
14     diff = diff + Math.Abs(pixel1.G - pixel2.G);
15     diff = diff + Math.Abs(pixel1.B - pixel2.B);
16   }
17 }
18 string result = (100 * (diff / 255) / (img1.Width * intMaxHeight * 3)).ToString();
```

Listing 5.4: Algorithmus zum Ermitteln der Differenz zweier Bilddateien [28].

Eine gültige Signatur wird von einem PDF Betrachtungsprogramm anders dargestellt, als eine ungültige Signatur. Diese unterschiedliche Darstellung führt zu einer Differenz im Screenshot-Vergleich. Anhand dieses Wertes können erfolgreiche bzw. nicht erfolgreiche Manipulationsversuche erkannt werden. Auf diese Weise ist eine Auswertung der in Abschnitt 5.4 erstellten Screenshots automatisiert möglich. In der Regel befinden sich die bereitgestellten Informationen über eine valide bzw. invalide Signatur im oberen Teil des PDF Betrachtungsprogramms. Daher ist es nicht nötig den Screenshot-Vergleich über die gesamte Pixelhöhe durchzuführen. Die Angabe über den maximal zu berücksichtigen Pixelbereich lässt sich zentral über die Einstellungen steuern. Dies ermöglicht eine Beschleunigung der Verarbeitungslogik, da auf diese Weise bereits nach weniger als der Hälfte der zu vergleichenden Pixel ein aussagekräftiges Ergebnis erzielt werden kann. Der entsprechende Menüpunkt wird

in Abschnitt 5.7 behandelt. Im gleichen Menüpunkt wird die Differenzgrenze für einen erfolgreichen Screenshot-Vergleich festgelegt. Liegt der errechnete Differenzwert zweier Bilddateien unterhalb bzw. auf gleicher Höhe dieser festgelegten Grenze wird der Vergleich als erfolgreich eingestuft und in der Textbox im linken Teil der grafischen Oberfläche ausgegeben. Neben dem Differenzwert wird ebenfalls der vollständige Dateiname des getesteten Screenshots visualisiert. Darüber hinaus werden sämtliche Ergebnisse im rechten Teil der grafischen Oberfläche als Gesamtverlauf dargestellt. Da für die Berechnung des Differenzwertes jedes Pixel der ersten Bilddatei mit dem Pixel an gleicher Stelle der zweiten Bilddatei verglichen wird, ist es notwendig, dass beide Screenshots die gleiche Pixelbreite aufweisen. Ist dies bei zwei zu vergleichenden Bilddateien nicht der Fall oder besitzt eine der beiden Screenshots eine geringere Pixelhöhe, als die definierte maximale Pixelhöhe in Abschnitt 5.7, so wird der Vergleich abgebrochen und eine entsprechende Hinweismeldung im Gesamtverlauf ausgegeben.

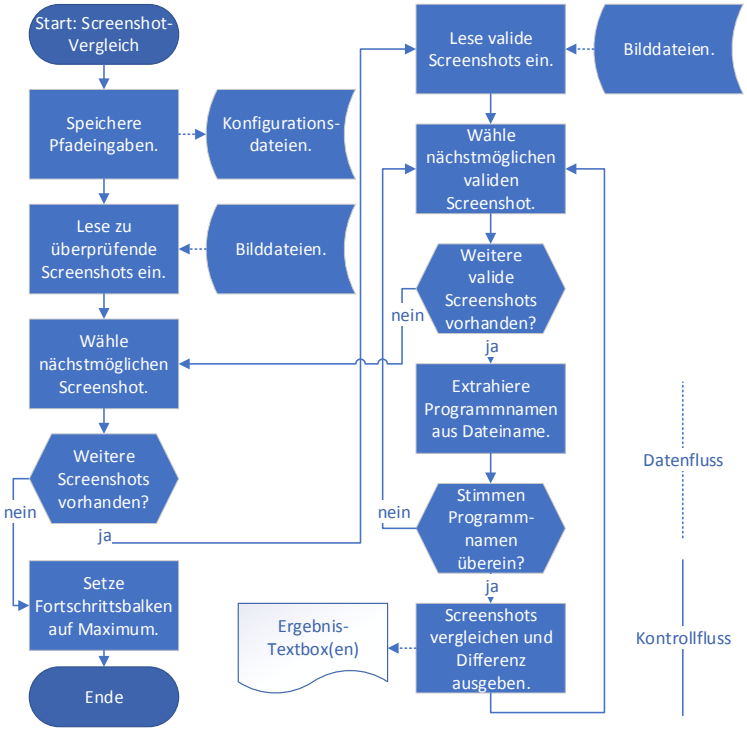

Abbildung 5.5: Flussdiagramm zur Funktion des Screenshot-Vergleichs.

5.6 Screenshot-Analyse

Der Screenshot-Vergleich setzt eine einheitliche Bildvorlage voraus und bietet keine Flexibilität bei Form und Ausgestaltung des Inhalts. Zu vergleichende Screenshots

dürfen sich in der Grundauflösung nicht unterscheiden. Darüber hinaus führen bspw. geöffnete Untermenüs innerhalb des PDF Betrachtungsprogramms zu einem deutlich erhöhten Differenzwert. Um Screenshots zu analysieren, die nicht die Eigenschaften für einen direkten Pixelvergleich mitbringen, kann die implementierte Texterkennung genutzt werden. Hierfür wird die Tesseract Optical Character Recognition (OCR) Engine verwendet, welche von 1985 bis 1994 durch Hewlett-Packard entwickelt wurde und seit 2006 von Google weiterentwickelt wird [17, 27]. Die Tesseract-Einheit wird extern eingebunden und ist somit kein unmittelbarer Teil des *PDF Tester* Programms. Dies hat den Vorteil, dass die Einheit unabhängig aktualisiert und ausgetauscht werden kann. Für die Auswertung innerhalb dieser Masterarbeit wurde die Version Tesseract 5.0.0 (alpha) verwendet, welche von der Universitätsbibliothek Mannheim bereitgestellt wird [34].

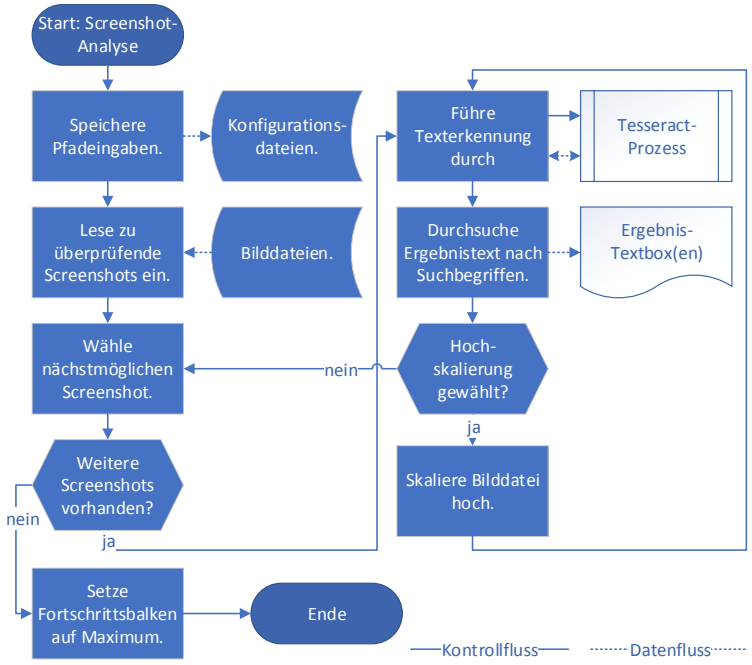

Abbildung 5.6: Flussdiagramm zur Screenshot-Analyse durch Texterkennung.

Der nachfolgend beschriebene Prozessablauf ist in Abbildung 5.6 als Flussdiagramm dargestellt. Bevor der Analyseprozess gestartet werden kann, ist es nötig den Pfad zu den zu überprüfenden Screenshots anzugeben. Der gesamte erkannte Text innerhalb der Bilddatei wird in eine Textdatei ausgelagert, deren Pfad vom Anwender anzugeben ist. Über eine Textbox innerhalb der grafischer Oberfläche können Zeichenketten übermittelt werden, welche als Suchbegriff innerhalb des erkannten Textes verwendet werden sollen. Wird ein Suchbegriff gefunden, erfolgt eine Ausgabe mit Hinweis auf die Bilddatei im rechten Teil der Oberfläche. Falls innerhalb eines Screenshots kein Suchbegriff gefunden wird, erfolgt diese Informationen über

die untere Textbox der Oberfläche. Bei der Analyse von Bilddateien hat es sich teilweise als Vorteil gezeigt, wenn die Bilddatei in einer hohen Auflösung vorlag. Über ein Optionsfeld kann ein zusätzlicher Durchlauf, mit einem auf ca. 6000 Pixel in der Breite hochskalierten Screenshot, aktiviert werden. Nachdem die Analyse gestartet wurde, wird die Tesseract-Einheit als externer Prozess aufgerufen und der zu analysierende Screenshot als Parameter, wie in Listing 5.7 zu sehen, übergeben. In einem weiteren Schritt wird der zurückgegebene Text auf das Vorkommen der spezifizierten Suchbegriffe hin durchleuchtet. Falls kein Suchbegriff gefunden wurde und die Option für einen zweiten Durchlauf gewählt wurde, wird die Tesseract-Einheit mit der hochskalierten Bilddatei erneut aufgerufen und abermals durchleuchtet. Der zu verwendende Tesseract-Programmordner und das für die Analyse zu nutzende Sprachpaket wird unter dem Menüpunkt „Einstellungen“ hinterlegt (siehe hierzu Abschnitt 5.7).

```
1  /*
2  Starte Tesseract als Prozess und übergebe:
3  - Screenshot
4  - Textdatei für das Ergebnis
5  - zu verwendendes Sprachpaket
6  Danach lies Ergebnisdatei ein.
7  */
8  using (Process proc = new Process())
9  {
10   proc.StartInfo.WorkingDirectory = tesseractPath;
11   proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
12   proc.StartInfo.CreateNoWindow = true;
13   proc.StartInfo.UseShellExecute = false;
14   proc.StartInfo.FileName = "cmd.exe";
15   proc.StartInfo.Arguments = "/c tesseract.exe " + tmpImg + " " + tmpResultText + " -l " +
         language;
16   proc.Start();
17   proc.WaitForExit();
18   if (proc.ExitCode == 0)
19   {
20     using (StreamReader sr = new StreamReader(tmpResultText + ".txt"))
21     {
22       string line;
23       while ((line = sr.ReadLine()) != null)
24       {
25         if (line.Trim() != "")
26           result += line + "\r\n";
27       }
28     }
29   }
30   else
31     result = "Fehler bei Verarbeitung mit tesseract.exe." + Environment.NewLine + "Status: "
           + proc.ExitCode;
32 }
```

Listing 5.7: Verwendung von Tesseract als externer Prozess [26].

5.7 Einstellungen

Allgemeine Konfigurationsvariablen lassen sich unter dem Menüpunkt „Einstellungen“ festlegen. Die Werte werden in die vorgegebenen Konfigurationsdateien „settings.conf“ und „tesseractPath.conf“ gespeichert, welche in Tabelle 5.8 beschrieben werden.
Die Konfigurationswerte lassen sich wie folgt zuordnen:

1. Feld: Wartezeit 1 | 5.4 Screenshot-Erstellung
2. Feld: Wartezeit 2 | 5.4 Screenshot-Erstellung
3. Feld: Differenzgrenze | 5.5 Screenshot-Vergleich
4. Feld: Maximale Pixelhöhe | 5.5 Screenshot-Vergleich
5. Feld: Sprachauswahl | 5.6 Screenshot-Analyse
6. Feld: Pfadangabe Tesseract | 5.6 Screenshot-Analyse

Tabelle 5.8: Verwendete Konfigurationsdateien und Standardwerte des *PDF Tester*-Programms.

Dateiname	**Verwendung**	**Standardwert**
debug.txt	Enthält aufgetretene Fehlermeldungen.	Leer
pdfPath.conf	Enthält die Pfadangabe zu den PDF Dateien (Ort: Start).	Leer
programList.conf	Enthält die Liste der PDF Betrachtungsprogramme (Ort: Start).	Leer
screenshotPathStart.conf	Enthält die Pfadangabe zum Screenshot-Verzeichnis (Ort: Start).	Leer
screenshotPathCompare.conf	Enthält die Pfadangabe zum Screenshot-Verzeichnis (Ort: Vergleich).	Leer
screenshotPathValid.conf	Enthält die Pfadangabe zum Verzeichnis mit Screenshots, die eine valide Signatur aufweisen (Ort: Vergleich).	Leer
screenshotPathOcr.conf	Enthält die Pfadangabe zum Screenshot-Verzeichnis (Ort: OCR).	Leer
searchStrings.conf	Enthält die Suchbegriffe der Texterkennung (Ort: OCR).	Leer
txtPath.conf	Enthält die Pfadangabe zur Textdatei, in der die Ausgabe der Texterkennung gespeichert werden soll (Ort: OCR).	Leer
settings.conf	Enthält die Werte der Einstellungsseite (Ort: Einstellung).	50;2;0,00001;500;eng
tesseractPath.conf	Enthält die Pfadangabe zum Tesseract-Programmordner (Ort: Einstellung).	C:\Users\%username%\ AppData\Local\ PdfTester\tesseract

6 Evaluation

In Rahmen dieser Masterarbeit wurden verschiedene Manipulationsversuche in unterschiedlichen Angriffsklassen ausgewertet. In diesem Kapitel werden die Ergebnisse dieser Auswertung vorgestellt. Es enthält darüber hinaus Hinweise zu den verwendeten PDF Betrachtungsprogrammen und der Testumgebung. Im letzten Teil werden weitere Erkenntnisse vorgestellt, welche sich im Rahmen der stattgefundenen Forschung zu digitalen Signaturen in PDF Dokumenten ergeben haben.

6.1 Testumgebung

Aktuell[1] liegt der Anteil von Windows Betriebssystemen im Desktop-Bereich bei 77,93 % [30]. Aus diesem Grund wird Windows 10 in Version 1809 als Basisbetriebssystem für die Testumgebung genutzt. Insgesamt werden drei Systeme verwendet: Signier-, Angreifer- und Opfer-System. Das Signier-System beinhaltet als einziges System den privaten Schlüssel zum digitalen Signieren. Um die Signaturen der PDF Dokumente zu erstellen, wurde das von Vladislav Mladenov entwickelte Programm *PDFSigner* auf Basis der Apache PDFBox Bibliothek [31] verwendet. Darüber hinaus sind weitere Signaturen mit Adobe Acrobat Professional 8 in Version 8.0.0 und Adobe Acrobat Pro 2017 in Version 2017.011.30143 generiert worden. Die Manipulationen wurden im Angreifer-System vorgenommen. Hierfür wurde das Programm Notepad++ in der Version 7.5.9 eingesetzt. Die darin enthaltenen Funktionen zum Kopieren und Einfügen von Binär-Inhalten bieten die Möglichkeit komprimierte Daten sowie Zeilenumbrüche korrekt zu übertragen. Die Auswertung der Manipulationen wurde im Opfer-System vorgenommen. Es beinhaltet alle in diesem Kapitel erwähnten PDF Betrachtungsprogramme sowie das während dieser Masterarbeit entstandene Programm *PDF Tester*, welches für die Teilautomatisierung der Auswertung eingesetzt wurde. Alle Systeme liegen als virtuelle Maschine unter Oracle VirtualBox in Version 5.2.32 vor. Falls das Installationsprogramm des jeweiligen PDF Betrachtungsprogramms in einer Offline-Variante vorlag, ist es entsprechend archiviert worden. Jeder in diesem Kapitel beschriebene Manipulationsversuch wurde unter jedem Betrachtungsprogramm ausgewertet und die Ergebnisse als Screenshot und in Tabellenform festgehalten.

[1] Stand: Oktober 2019

6.2 Anforderungen an PDF Betrachtungsprogramme

Im Folgenden werden die Mindestanforderungen an die PDF Betrachtungsprogramme definiert. Dies ist nötig, um die erstellten Angriffsvektoren vergleichbar evaluieren zu können. Ausschlüsse verschiedener PDF Betrachtungsprogramme werden begründet und eine Einteilung in zwei Klassen für unterschiedliches Verhalten bei Modifikationen getroffen.

Anforderungen:

1. Signierte PDF Dokumente können validiert werden.
2. Die Signaturvalidierung ist auch nach einem Incremental Update möglich.
3. Die Signaturvalidierung ist auch möglich, falls das PDF Dokument Formularfelder enthält.
4. Die Signaturvalidierung ist auch möglich, falls das PDF Dokument eine DocMDP Methode enthält, unabhängig vom gewählten Parameter.
5. Die Signaturvalidierung ist auch möglich, falls das PDF Dokument eine FieldMDP Methode enthält, unabhängig vom gewählten Parameter.

6.2.1 Ausgeschlossene PDF Betrachtungsprogramme

Während der Auswertung von Manipulationsversuchen sowie bei der Adaption von Exploits auf Certification Signaturen, wurden mehrere PDF Betrachtungsprogramme identifiziert, die nicht den definierten Anforderungen aus Abschnitt 6.2 genügen. Diese Betrachtungsprogramme sind in Tabelle 6.1 aufgeführt.

Tabelle 6.1: Ausgeschlossene PDF Betrachtungsprogramme.

Name	Version	Begründung
PDF Studio 2019 / Pro / Viewer	V2019.0.0	Anforderung 2 wird nicht erfüllt, da jede Form von Incremental Updates zu der Ausgabe „Signature UNKNOWN“ führt und somit nicht zwischen gültiger und ungültiger Signatur unterschieden werden kann.
Expert PDF Reader	V9.0.180	Anforderung 3 wird nicht erfüllt, da keine Signaturvalidierung möglich ist, wenn das signierte PDF Dokument Formularfelder enthält. Ebenfalls nicht erfüllt werden Anforderung 4 und 5, da die Signaturvalidierung nur unter DocMDP Parameter `P 1` bzw. bei FieldMDP nur ohne Lock-Parameter möglich ist.
PDF-XChange Viewer	V2.5 (Build 322.10)	Anforderung 4 und 5 werden nicht erfüllt, da die Signaturvalidierung nur unter DocMDP Parameter `P 1` bzw. bei FieldMDP nur ohne Lock-Parameter möglich ist.

Das PDF Betrachtungsprogramm Adobe Reader XI befindet sich seit 15. Oktober 2017 außerhalb des Adobe Lebenszyklus [5]. Aus diesem Grund wird das Betrachtungsprogramm in dieser Arbeit durch Adobe Acrobat Pro 2017 ersetzt, für das

offiziell ein Ende des Core-Supports am 6. Juni 2022 angegeben ist [6]. Insgesamt wurden 23 PDF Betrachtungsprogramme identifiziert, wovon 18 PDF Betrachtungsprogramme für die Analyse geeignet sind.

6.2.2 Klasseneinteilung

Einige PDF Betrachtungsprogramme reagieren auf Modifikationen mit einer Hinweismeldung. Die Signatur ist weiterhin als gültig angezeigt, aber es wird auf Änderungen nach dem Signieren hingewiesen. Der Hinweis ist allgemein gefasst und enthält keine Informationen was genau geändert wurde. Darüber hinaus wird die Hinweismeldung auch bei Änderungen ausgegeben, welche durch die Transformationsmethoden genehmigt wurden. Dies kann bspw. mehrfache Signierung oder das Ausfüllen von Formularfeldern sein. Neben der Hinweismeldung wird das Validierungssymbol der Signatur leicht angepasst, siehe hierzu Abbildung 6.12. Für diese Betrachtungsprogramme ist das maximal erreichbare Ziel, welches durch einen Angriff erreicht werden kann, eine gültige Signaturvalidierung mit entsprechender Hinweismeldung. Dadurch ergibt sich eine Einteilung der Betrachtungsprogramme in zwei Klassen:

- Klasse 1: Modifikationen sind ohne Hinweismeldung auf nachträgliche Änderungen möglich.
- Klasse 2: Modifikationen sind nur mit Hinweismeldung auf nachträgliche Änderungen möglich.

In der Tabelle 6.2 sind die PDF Betrachtungsprogramme der Klasse 2 mit entsprechender Hinweismeldung in UI-Layer 1 und 2 aufgeführt. Ausschlagend für die Einteilung ist das Verhalten der Betrachtungsprogramme auf minimalste Änderungen nach der Signierung, wie bspw. ein Leerzeichen oder Leerzeile hinter dem Dokumentende.

Tabelle 6.2: Hinweismeldungen der PDF Betrachtungsprogramme in Klasse 2.

Name	Version	Hinweismeldung (Certification)	Hinweismeldung (Approval)
Expert PDF 14	14.0.25.3456 64-bit	„Signature is Valid. The part of the document approved by this signature has not been altered; however, some other modifications have been made in the document.“	Gleiche Hinweismeldung.
LibreOffice Draw	6.2.5.2 (x64)	„The signature is OK, but the document is only partially signed.“	Gleiche Hinweismeldung.
Nitro Pro	12.16.3.574	„Certificate is valid, document hast been changed after it was certified“	„Signature is valid, document hast been changed after it was signed“
Nitro Reader	5.5.9.2	Keine Hinweismeldung, gleiches Validierungssymbol.	Keine Hinweismeldung, nur anderes Validierungssymbol.
PDF Architect 7	7.0.26.3193 64-bit	„Signature is Valid. The part of the document approved by this signature has not been altered; however, some other modifications have been made in the document.“	Gleiche Hinweismeldung.
PDF-XChange Editor	8.0 (Build 331.0)	„Signature is VALID. There have been subsequent changes to the document.“	Gleiche Hinweismeldung.
Perfect PDF Reader	V14.0.9 (29.0)	„Gültig für unterschriebene Version“	Gleiche Hinweismeldung.
Soda PDF Desktop	11.1.09.4184 64-bit	„Signature is Valid. The part of the document approved by this signature has not been altered; however, some other modifications have been made in the document.“	Gleiche Hinweismeldung.

6.3 Adaption veröffentlichter Exploits und Re-Evaluation

Die von Mladenov et al. [35] veröffentlichten Exploits[2] sind überwiegend auf Basis von Approval Signaturen erstellt worden. Dieser Abschnitt befasst sich mit den Ergebnissen der Adaption auf Certification Signaturen sowie der Re-Evaluation dieser Exploits. Aus den Ergebnissen wird somit ersichtlich, welche Schwachstellen zum Zeitpunkt der Auswertung weiterhin in den getesteten PDF Betrachtungsprogrammen vorhanden sind.
Die Tabelle 6.3 enthält die Ergebnisse zur Auswertung der bereits veröffentlichten Exploits auf Approval Signaturbasis sowie die Ergebnisse für die auf Certification Signaturen übertragenen Exploits. Es wird ersichtlich, dass viele Hersteller Sicherheitsupdates entwickelt haben, welche die Lücken geschlossen haben. So sind bspw. keine Angriffe mehr in der USF-Kategorie erfolgreich. Allerdings sind auch einige

[2] `https://pdf-insecurity.org/signature/evaluation_2018.html#download-pocs`

Negativbeispiele sichtbar, so sind bspw. in PDF Editor 6 Pro und PDFelement Pro, trotz neuer Versionsnummer, weiterhin die Angriffe in der ISA und SWA-Kategorie erfolgreich. Im Fall des Nitro Reader und Perfect PDF 10 Premium gab es keine Aktualisierungen seitens der Herstellers. Daher sind die Angriffe für beide PDF Betrachtungsprogramme weiterhin erfolgreich. Es wird weiterhin ersichtlich, dass es bei Nitro Reader und Power PDF Standard Unterschiede zwischen den Signaturtypen gibt. Diese Betrachtungsprogramme erlauben keinerlei nachträgliche Änderung, wenn eine Certification Signatur angewendet wurde.
Insgesamt sind 8 von 18 PDF Betrachtungsprogramme weiterhin für mindestens einen Angriff anfällig.

Tabelle 6.3: Ergebnisse der Re-Evaluation veröffentlichter Exploits.

		Certification			Approval		
Name	**Version**	**USF**	**ISA**	**SWA**	**USF**	**ISA**	**SWA**
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	✓	✓	✓	✓	✓
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	✓	✓	✓	✓	✓
Expert PDF 14	14.0.25.3456 64-bit	✓	✓	✓	✓	✓	✓
Foxit Reader	9.5.0.20723, 9.7.0.29455	✓	✓	✓	✓	✓	✓
Foxit PhantomPDF	9.7.0.29478	✓	✓	✓	✓	✓	✓
LibreOffice Draw	6.2.5.2 (x64)	✓	✓	✓	✓	◗	✓
Master PDF Editor	5.4.22, 64 bit	✓	●	✓	✓	●	✓
Nitro Pro	12.16.3.574	✓	✓	✓	✓	✓	✓
Nitro Reader	5.5.9.2	✓	✓	✓	✓	◗	◗
PDF Architect 7	7.0.26.3193 64-bit	✓	✓	✓	✓	✓	✓
PDF Editor 6 Pro	6.5.0.3929	✓	●	●	✓	●	●
PDFelement Pro	7.0.3.4309	✓	●	●	✓	●	●
PDF-XChange Editor	8.0 (Build 331.0)	✓	✓	✓	✓	✓	✓
Perfect PDF Reader	V14.0.9 (29.0)	✓	◗	●	✓	✓	●
Perfect PDF 8 Reader	8.0.3.5	✓	●	●	✓	●	●
Perfect PDF 10 Premium	10.0.0.1	✓	●	●	✓	●	●
Power PDF Standard	3.0	✓	✓	✓	✓	●	✓
Soda PDF Desktop	11.1.09.4184 64-bit	✓	✓	✓	✓	✓	✓
Anzahl angreifbarer PDF Betrachtungsprogramme:		0/18	6/18	5/18	0/18	8/18	6/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

6.3.1 Universal Signature Forgery

Innerhalb der USF-Exploits wurden sieben verschiedene Varianten identifiziert. Zwei der Exploits sind Betrachtungsprogrammen von Adobe zuzuordnen und fünf weitere Exploits für PDF Editor 6 Pro und PDFelement Pro.

Adobe Acrobat Reader DC und Adobe Acrobat Pro 2017

Die Manipulation am signierten PDF Dokument wurde im Signatur-Dictionary vorgenommen. Dabei wird in der ersten Variante die gesamte Zeile der `ByteRange` entfernt. Variante zwei weist der `ByteRange` das `null`-Objekt als Wert zu. Die Schwachstelle lässt sich unter den beiden PDF Betrachtungsprogrammen nicht mehr ausnutzen. Dies gilt sowohl bei der Verwendung von Approval als auch bei der Verwendung von Certification Signaturen.

PDF Editor 6 Pro und PDFelement Pro

Für die Betrachtungsprogramme PDF Editor 6 Pro und PDFelement Pro wurden insgesamt fünf USF-Exploits veröffentlicht. Diese enthalten ebenfalls Manipulationen im Signatur-Dictionary, betreffen aber die `Contents`-Zeile und lassen sich wie folgt unterscheiden:

- Gesamte `Contents`-Zeile wird entfernt.
- Dem `Contents`-Eintrag wird kein Wert hinzugefügt.
- Dem `Contents`-Eintrag wird eine leere hexadezimale Zeichenkette hinzugefügt: `<>`.
- Dem `Contents`-Eintrag wird das null-Objekt hinzugefügt.
- Dem `Contents`-Eintrag wird eine hexadezimale Zeichenkette mit Byte-Wert 0 hinzugefügt: `<00>`.

Die Manipulationen führten sowohl unter Approval als auch unter Certification signierten PDF Dokumenten nicht zu einer gültigen Signatur.

6.3.2 Incremental Saving Attacks

ISA-Exploits wurden für die PDF Betrachtungsprogramme Foxit Reader, LibreOffice Draw, Master PDF Editor, Nitro Pro, Nitro Reader, PDF Editor 6 Pro, PDFelement Pro, Perfect PDF Reader, Perfect PDF 10 Premium und Power PDF Standard veröffentlicht.

Foxit Reader

Die beiden bereitgestellten Exploits wurden bereits auf Basis einer Certification Signatur erstellt, daher ist eine Adaption nicht notwendig. Die gesamten indirekten Objekte des Incremental Update der Signatur wurden kopiert und ans Ende der Datei angehängt. Hinzu gekommen sind indirekte Objekte der Typen `Pages`, `Page` sowie ein neues Inhaltsobjekt, welches den manipulierten Text enthält. Das `Catalog`-Objekt referenziert auf das neue `Pages`-Objekt, welches in der ersten Variante einen `Kids`-Eintrag mit Referenz auf das alte und neue `Page`-Objekt enthält. In Variante zwei wird lediglich auf das neue `Page`-Objekt referenziert. Das Dokument endet mit `%%EOF` ohne Angabe einer Cross-Reference Table oder Trailer. In den getesteten Programmversionen bleibt die Attacke erfolglos.

Foxit PhantomPDF

Das Betrachtungsprogramm Foxit PhantomPDF wurde mit ausgewertet, war aber nicht Bestandteil der Evaluation durch Mladenov et al. Es wurden die gleichen Exploits, wie für Foxit Reader ausgewertet. In der getesteten Programmversion bleibt die Attacke erfolglos.

LibreOffice Draw

Das manipulierte Dokument enthält im Incremental Update ein neues `Pages`-Objekt, welches über den `Kids`-Eintrag auf ein neues `Page`-Objekt referenziert. Dieses indirekte Objekt referenziert wiederum auf ein neues Inhaltsobjekt mit manipuliertem Inhalt. Die Cross-Reference Table wird zwar mit dem Schlüsselwort `xref` eingeleitet, enthält aber keine Einträge. Der nachfolgende Trailer besteht aus den Minimalangaben und enthält eine gültige Referenzierung auf `xref`. Das Dokument schließt nicht mit `%%EOF` ab. Beim PDF Dokument mit Approval Signatur wird die Manipulation umgesetzt. Die Signatur ist gültig, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2. Für das Dokument auf Certification Signaturbasis wird seitens LibreOffice Draw hingegen keine Signatur erkannt, somit scheitert die Attacke in diesem Fall.

Master PDF Editor

Das manipulierte Dokument enthält im Incremental Update ein neues `Pages`-Objekt, welches über den `Kids`-Eintrag auf ein neues `Page`-Objekt referenziert. Dieses indirekte Objekt referenziert wiederum auf ein neues Inhaltsobjekt mit manipuliertem Inhalt. Die Cross-Reference Table wird zwar mit dem Schlüsselwort `xref` eingeleitet, enthält aber keine Einträge. Der nachfolgende Trailer enthält eine ungültige Referenzierung auf `xref`. Das Dokument schließt mit `%%EOF` ab. Die Signaturverifikation endet bei beiden Signaturvarianten mit einem gültigen Ergebnis. Master PDF Editor bleibt somit weiterhin für diese Attacke anfällig.

Nitro Pro

Das PDF Betrachtungsprogramm teilt sich die gleiche Exploitbasis, wie bereits im Abschnitt des Master PDF Editor dargestellt. Das Programm Nitro Pro ist, unabhängig vom Signaturtyp, nicht für die Attacke anfällig.

Nitro Reader[3]

Das PDF Betrachtungsprogramm teilt sich die gleiche Exploitbasis, wie bereits im Abschnitt des Master PDF Editor dargestellt. Die Signaturverifikation endet beim Nitro Reader mit einem gültigen Ergebnis. Allerdings nur unter Verwendung eines PDF Dokuments mit Approval Signatur. Das gleiche Dokument mit Certification Signatur weist der Nitro Reader als Dokument mit invalider Signatur aus.

PDF Editor 6 Pro und PDFelement Pro

Die PDF Betrachtungsprogramme teilen sich die gleiche Exploitbasis, wie bereits im Abschnitt des Master PDF Editor dargestellt. Die Signaturverifikation endet mit einem gültigen Ergebnis. Dies gilt gleichermaßen für beide Signaturtypen. PDF Editor 6 Pro und PDFelement Pro bleiben somit weiterhin für diese Attacke anfällig.

Perfect PDF Reader

Das manipulierte Dokument enthält im Incremental Update ein neues `Catalog` und `Pages`-Objekt, welches über den `Kids`-Eintrag auf ein neues `Page`-Objekt referenziert. Dieses indirekte Objekt referenziert wiederum auf ein neues Inhaltsobjekt mit manipuliertem Inhalt. Cross-Reference Table und Trailer sind vollständig enthalten. Die Signatur liegt bereits auf Certification-Basis vor, daher ist eine Adaption nicht notwendig. In der getesteten Programmversion ist die Attacke erfolgreich.

Perfect PDF 8 Reader

Das Betrachtungsprogramm Perfect PDF 8 Reader wurde mit ausgewertet, war aber nicht Bestandteil der Evaluation durch Mladenov et al. Es wurde der gleiche Exploit, wie für Perfect PDF 10 Premium ausgewertet. Die Ergebnisse entsprechen denen des Perfect PDF 10 Premium.

Perfect PDF 10 Premium[4]

Das manipulierte Dokument enthält im Incremental Update ein neues `Catalog` und

[3] Nitro Reader Version gleich zur Evaluation von Mladenov et al.
[4] Perfect PDF 10 Premium Version gleich zur Evaluation von Mladenov et al.

Pages-Objekt, welches über den Kids-Eintrag auf ein neues Page-Objekt referenziert. Dieses indirekte Objekt referenziert wiederum auf ein neues Inhaltsobjekt mit manipuliertem Inhalt. Eine Cross-Reference Table ist nicht enthalten. Der Trailer ist vollständig, referenziert aber auf einen ungültigen Byte-Wert. Die Signaturverifikation endet mit einem gültigen Ergebnis. Dies gilt gleichermaßen für beide Signaturtypen.

Power PDF Standard

Das manipulierte Dokument enthält im Incremental Update ein neues Catalog und Pages-Objekt, welches über den Kids-Eintrag auf ein neues Page-Objekt referenziert. Dieses indirekte Objekt referenziert wiederum auf ein neues Inhaltsobjekt mit manipuliertem Inhalt. Die Cross-Reference Table enthält die vollständige Liste der neuen Objekte mit korrekten Byte-Werten. Der nachfolgende Trailer enthält eine gültige Referenzierung auf xref. Das Dokument schließt mit %%EOF ab. Die Signaturverifikation endet für die Approval Signaturvariante mit einem gültigen Ergebnis. Nicht anfällig für die Attacke zeigt sich das PDF Betrachtungsprogramm bei Verwendung einer Certification Signatur.

6.3.3 Signature Wrapping Attacks

SWA-Exploits wurden für die PDF Betrachtungsprogramme Expert PDF 14, Foxit Reader, Nitro Pro, Nitro Reader, PDF Architect 7, PDF Editor 6 Pro, PDFelement Pro, PDF-XChange Editor, Perfect PDF Reader, Perfect PDF 10 Premium, Power PDF Standard und Soda PDF Desktop veröffentlicht.

Expert PDF 14

Das manipulierte Dokument enthält nach dem Contents-Eintrag eine neue ByteRange mit verändertem Wert an dritter Stelle, welcher dem neuen Byte-Wert des alten ByteRange-Objekts entspricht. Zwischen neuer und alter ByteRange sind die Objekte Catalog, Pages, Page sowie ein manipuliertes neues Inhaltsobjekt eingefügt. Darüber hinaus befindet sich nach dem Inhaltsobjekt eine Cross-Reference Table mit ungültigen Byte-Werten innerhalb der Objektreferenzierung. Das letzte Element vor der alten ByteRange bildet ein Trailer, ohne Referenz auf die neue Cross-Reference Table. Die Signatur im Dokument wird von Expert PDF 14 nicht mehr gefunden. Dies gilt für beide Signaturtypen.

Foxit Reader

Das manipulierte Dokument enthält einen gekürzten Contents-Eintrag. Die zum Auffüllen verwendeten Null-Bytes in hexadezimaler Schreibweise wurden bis auf elf Stellen entfernt. Nach der schließenden spitzen Klammer folgt eine neue ByteRange mit verändertem Wert an dritter Stelle, welcher dem neuen Byte-Wert des alten

`ByteRange`-Objekts entspricht. Zwischen neuer und alter `ByteRange` befindet sich ein manipuliertes Inhaltsobjekt, welches durch eine neue vollständige Cross-Reference Table referenziert wird. Ein Trailer ist nicht vorhanden. Die Signatur ist unter beiden Signaturtypen ungültig.

Foxit PhantomPDF

Das Betrachtungsprogramm Foxit PhantomPDF wurde mit ausgewertet, war aber nicht Bestandteil der Evaluation durch Mladenov et al. Es wurde der gleiche Exploit, wie für Foxit Reader ausgewertet. In der getesteten Programmversion bleibt die Attacke erfolglos.

Nitro Pro und Nitro Reader

Die PDF Betrachtungsprogramm teilen sich die gleiche Exploitbasis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Beide Betrachtungsprogramme sind, unabhängig vom Signaturtyp, nicht für die Attacke anfällig.

PDF Architect 7

Das PDF Betrachtungsprogramm teilt sich die gleiche Exploitbasis, wie bereits im Abschnitt des Foxit Reader dargestellt. Das Programm PDF Architect 7 ist, unabhängig vom Signaturtyp, nicht für die Attacke anfällig.

PDF Editor 6 Pro und PDFelement Pro

Für die PDF Betrachtungsprogramme wurden zwei Exploits veröffentlicht. Der erste Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Der zweite Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Foxit Reader dargestellt. Beide Exploits sind unter beiden Signaturtypen erfolgreich und führen zu einer gültigen Signaturvalidierung.

PDF-XChange Editor

Für das PDF Betrachtungsprogramm wurden zwei Exploits veröffentlicht. Der erste Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Der zweite Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Foxit Reader dargestellt. Das Programm PDF-XChange Editor ist für beide Exploits, unabhängig vom Signaturtyp, nicht anfällig.

Perfect PDF Reader

Für das PDF Betrachtungsprogramm wurden zwei Exploits veröffentlicht. Der erste Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Der zweite Exploit ist auf Certification-Basis und enthält nach dem `Contents`-Eintrag eine neue `ByteRange` mit verändertem Wert an dritter Stelle,

welcher dem neuen Byte-Wert des alten `ByteRange`-Objekts entspricht. Zwischen neuer und alter `ByteRange` sind die Objekte `SigRef`, `TransformParams` sowie ein `Pages`-Objekt, welches die Original Seite zweimal als `Kids`-Eintrag enthält. Darüber hinaus findet sich eine vollständige Cross-Reference Table mit gültiger Referenzierung sowie ein vollständiger Trailer unter den manipulierten Objekten. Direkt vor der alten `ByteRange` findet sich die Zeile `%%EOF`. Exploit 1 ist unter beiden Signaturtypen erfolgreich und führt zu einer gültigen Signaturvalidierung. Exploit 2 gibt die Hauptseite ohne Manipulation zweimal aus. Die Signatur bleibt währenddessen gültig.

Perfect PDF 8 Reader

Das Betrachtungsprogramm Perfect PDF 8 Reader wurde mit ausgewertet, war aber nicht Bestandteil der Evaluation durch Mladenov et al. Es wurde der gleiche Exploit wie für Perfect PDF 10 Premium ausgewertet. Der Exploit ist unter beiden Signaturtypen erfolgreich und führt zu einer gültigen Signaturvalidierung.

Perfect PDF 10 Premium[5]

Das PDF Betrachtungsprogramm teilt sich die gleiche Exploitbasis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Der Exploit ist unter beiden Signaturtypen erfolgreich und führt zu einer gültigen Signaturvalidierung.

Power PDF Standard

Für das PDF Betrachtungsprogramm wurden zwei Exploits veröffentlicht. Der erste Exploit teilt sich die gleiche Basis, wie bereits im Abschnitt des Expert PDF 14 dargestellt. Der zweite Exploit enthält nach dem `Contents`-Eintrag eine neue `ByteRange` mit verändertem Wert an dritter Stelle, welcher dem neuen Byte-Wert des alten `ByteRange`-Objekts entspricht. Zwischen neuer und alter `ByteRange` sind die Objekte `Pages`, `Page` sowie ein manipuliertes neues Inhaltsobjekt eingefügt. Darüber hinaus befindet sich nach dem Inhaltsobjekt eine Cross-Reference Table mit gültigen Byte-Werten innerhalb der Objektreferenzierung. Ein Trailer ist nicht vorhanden. Exploit 1 kann als Approval-Version nicht geöffnet werden. Als Certification-Version ist die Signatur ungültig. Exploit 2 führt bei beiden Signaturtypen zu einer ungültigen Signatur.

Soda PDF Desktop

Der Exploit ist auf Certification-Basis und enthält nach dem `Contents`-Eintrag eine neue `ByteRange` mit verändertem Wert an dritter Stelle, welcher dem neuen Byte-Wert des alten `ByteRange`-Objekts entspricht. Zwischen neuer und alter `ByteRange` befinden sich die Objekte `SigRef`, `TransformParams`, `Catalog` und `Pages`, welches

[5]Perfect PDF 10 Premium Version gleich zur Evaluation von Mladenov et al.

die Original Seite zweimal als `Kids`-Eintrag enthält. Dazu kommt außerdem ein `Page`-Objekt sowie ein Inhaltsobjekt. Darüber hinaus findet sich eine vollständige Cross-Reference Table mit gültiger Referenzierung sowie ein vollständiger Trailer unter den manipulierten Objekten. Direkt vor der alten `ByteRange` findet sich die Zeile `%%EOF`. Das Programm Soda PDF Desktop ist für den Exploit, unabhängig vom Signaturtyp, nicht anfällig.

6.4 Transformationsmethoden

In diesem Abschnitt wurde analysiert, wie die PDF Betrachtungsprogramme mit Restriktionen durch die Transformationsmethoden umgehen. Dabei ist der Abschnitt 6.4.1 den Certification Signaturen und der Abschnitt 6.4.2 den Approval Signaturen zuzuordnen. Die Angriffsvektoren sind eine Unterform der ISA-Angriffsklasse (siehe Abschnitt 4.2.3).
Die Tabelle 6.4 enthält die Ergebnisse zur Auswertung der Transformationsmethoden Analyse für DocMDP und FieldMDP. Insgesamt sind 15 von 18 PDF Betrachtungsprogramme für mindestens einen Angriff anfällig. Die Ergebnisse zeigen deutlich, dass die meisten Betrachtungsprogramme die Restriktionen, welche sich aus den Transformationsmethoden ergeben, nicht korrekt umsetzen. Während die Adobe Programme der eigenen Spezifikation korrekt folgen und die Signatur als ungültig werten, weist bspw. Foxit zwar auf die konkrete Änderung hin, aber wertet die Signatur als gültig.

Tabelle 6.4: Ergebnisse der Transformationsmethoden Analyse.

Name	Version	Certification DocMDP	Approval FieldMDP
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	✓
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	✓
Expert PDF 14	14.0.25.3456 64-bit	◗	◗
Foxit Reader	9.5.0.20723, 9.7.0.29455	◗	◗
Foxit PhantomPDF	9.7.0.29478	◗	◗
LibreOffice Draw	6.2.5.2 (x64)	◗	◗
Master PDF Editor	5.4.22, 64 bit	●	●
Nitro Pro	12.16.3.574	✓	◗
Nitro Reader	5.5.9.2	✓	✓
PDF Architect 7	7.0.26.3193 64-bit	◗	◗
PDF Editor 6 Pro	6.5.0.3929	●	●
PDFelement Pro	7.0.3.4309	●	●
PDF-XChange Editor	8.0 (Build 331.0)	◗	◗
Perfect PDF Reader	V14.0.9 (29.0)	◗	◗
Perfect PDF 8 Reader	8.0.3.5	●	●
Perfect PDF 10 Premium	10.0.0.1	●	●
Power PDF Standard	3.0	✓	◗
Soda PDF Desktop	11.1.09.4184 64-bit	◗*	◗
Anzahl angreifbarer PDF Betrachtungsprogramme:		13/18	15/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

*Die Angriffe auf Soda PDF Desktop setzen mindestens DocMDP Parameter `P 2` voraus.

6.4.1 DocMDP

Angriffsvektoren

Für die Analyse wurden folgende Angriffsvektoren entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit Formularfeldern erstellt und mit einer Certification Signatur versehen. Durch den DocMDP Parameter `P 1` sind keine Änderungen am Dokument erlaubt. Anschließend wurde ein Incremental Update mit korrekter Cross-Reference Table und Trailer ans Dokument angehängt, welches das zweite Formularfeld mit „Hello Attacker“ überschreibt.

- Für den Angriffsvektor 2 wurde ein PDF Dokument mit Inhalt „Hello World“ erstellt und mit einer Certification Signatur versehen. Durch den DocMDP Parameter `P 1` sind keine Änderungen am Dokument erlaubt. Anschließend wurde ein Incremental Update mit korrekter Cross-Reference Table und Trailer ans Dokument angehängt, welches eine Bilddatei mit dem Inhalt „Hello Attacker“ über den Originalinhalt legt.

Angriffsvektor 1 und 2 wurden für Perfect PDF Reader und Angriffsvektor 2 für Soda PDF Desktop modifiziert.
Insgesamt sind 13 von 18 PDF Betrachtungsprogramme für die Angriffe anfällig.

Foxit Reader und Foxit PhantomPDF

Die PDF Betrachtungsprogramme setzen das Incremental Update für beide Angriffsvektoren um. Die Signaturen sind gültig, aber es wird folgende Hinweismeldung ausgegeben: „Document certification is valid: This document has been modified since being certified, but these modifications are allowed and do not invalidate the signature.“. Des Weiteren erfolgt der Hinweis, dass ein Formularfeld ausgefüllt wurde bzw. für Angriffsvektor 2, dass eine Annotation erstellt wurde. Foxit Reader und Foxit PhantomPDF erkennen also die Änderung am Dokument, invalidieren aber nicht die Signatur, wie von der Spezifikation vorgesehen.

LibreOffice Draw

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben. Für Angriffsvektor 2 gilt die Einschränkung, dass die Bilddatei zwar angezeigt wird, diese aber nicht korrekt über dem Originalinhalt platziert wird.

Perfect PDF Reader

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Damit die Signaturen als gültig angezeigt werden, bedarf es dem folgenden zusätzlichen Eintrag in der Cross-Reference Table:

```
xx 1
0000000000 00000 n
```

`xx` muss jeweils der Nummer des Objekts vom Typ `SigRef` angepasst werden, welches dem Signatur-Reference-Dictionary entspricht. Durch Verweis auf Byte-Position 0, wird es vom Betrachtungsprogramm nicht mehr gefunden und dadurch die Restriktion umgangen. Anschließend werden die Signaturen als gültig angezeigt, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2.

Soda PDF Desktop

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Allerdings wird eine gültige Signatur nur unter DocMDP Parameter `P 2` erreicht. Der erste Angriffsvektor kann damit nicht mehr berücksichtigt werden, da das Ausfüllen von Formularfeldern unter Parameter `P 2` ohnehin erlaubt ist. Für den zweiten Angriffsvektor lässt sich unter Parameter `P 2` eine gültige Signaturvalidierung realisieren, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

6.4.2 FieldMDP

Angriffsvektoren

Für die Analyse wurden drei Angriffsvektoren erstellt. Jeder Angriffsvektor besteht aus einem PDF Dokument mit Formularfeldern und einer Approval Signatur. Hierbei wurde jeweils ein Dokument mit FieldMDP Parameter `All`, `Include` oder `Exclude` eingeschränkt. Anschließend wurde ein Incremental Update mit korrekter Cross-Reference Table und Trailer ans Dokument angehängt, welches ein vom FieldMDP Parameter geschütztes Formularfeld mit „Hello Attacker“ überschreibt.
Insgesamt sind 15 von 18 PDF Betrachtungsprogramme für die Angriffe anfällig.

Foxit Reader und Foxit PhantomPDF

Die PDF Betrachtungsprogramme setzen das Incremental Update für alle Angriffsvektoren um. Die Signaturen sind gültig, aber es wird folgende Hinweismeldung ausgegeben: „Signature is valid: The revision of the document that was covered by this signature has not been altered; however, there have been subsequent changes to the document.“. Des Weiteren erfolgt der Hinweis, dass ein Formularfeld ausgefüllt wurde. Foxit Reader und Foxit PhantomPDF erkennen also die Änderung am Dokument, invalidieren aber nicht die Signatur, wie von der Spezifikation vorgesehen.

Power PDF Standard

Das PDF Betrachtungsprogramm setzt das Incremental Update für alle Angriffsvektoren um. Die Signaturen sind gültig, aber es wird folgende Hinweismeldung ausgegeben: „Signature is valid: The revision of the document has not been altered; There have been subsequent changes to the document“.

6.5 Font-Masking

Bei den Angriffen zu Font-Masking wird dem PDF Dokument eine manipulierte Schriftart angehängt, welche bspw. Zahlen einer Kontonummer vertauschen soll.

Dadurch wird die Darstellung des PDF Dokuments verändert, während der Inhalt gleich bleibt. Da Schriftarten in der Regel nicht als Risiko durch die PDF Betrachtungsprogramme betrachtet werden, können auf diese Weise Sicherheitsüberprüfungen, welche den Inhalt des Dokuments schützen sollen, umgangen werden.
Die Tabelle 6.5 enthält die Ergebnisse zur Auswertung der Font-Masking Analyse für Certification und Approval Signaturen.

Tabelle 6.5: Ergebnisse der Font-Masking Analyse.

Name	**Version**	**Certification Font-Masking**	**Approval Font-Masking**
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	✓
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	✓
Expert PDF 14	14.0.25.3456 64-bit	◗	◗
Foxit Reader*	9.5.0.20723, 9.7.0.29455	●	●
Foxit PhantomPDF	9.7.0.29478	●	●
LibreOffice Draw**	6.2.5.2 (x64)	✓	✓
Master PDF Editor	5.4.22, 64 bit	●	●
Nitro Pro	12.16.3.574	✓	◗
Nitro Reader	5.5.9.2	✓	◗
PDF Architect 7	7.0.26.3193 64-bit	◗	◗
PDF Editor 6 Pro	6.5.0.3929	●	●
PDFelement Pro	7.0.3.4309	●	●
PDF-XChange Editor	8.0 (Build 331.0)	◗	◗
Perfect PDF Reader	V14.0.9 (29.0)	◗	◗
Perfect PDF 8 Reader	8.0.3.5	●	●
Perfect PDF 10 Premium	10.0.0.1	●	●
Power PDF Standard	3.0	✓	●
Soda PDF Desktop	11.1.09.4184 64-bit	◗***	◗
Anzahl angreifbarer PDF Betrachtungsprogramme:		12/18	15/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

*Font-Masking mit Foxit Reader ist nur unter Version 9.7.0.29455 erfolgreich.
**LibreOffice Draw nutzte für die Darstellung keine im PDF integrierte Schriftart.
***Font-Masking mit Soda PDF Desktop setzt mindestens DocMDP Parameter `P 2` voraus.

Insgesamt sind 15 von 18 PDF Betrachtungsprogramme für mindestens einen Angriff anfällig. Auch hier sind Nitro Pro, Nitro Reader und Power PDF Standard lediglich gegen die Approval Angriffsvariante anfällig. Das Incremental Update bei

der Certification Variante wird als unerlaubt eingestuft. In der Gesamtheit zeigt das Ergebnis, dass die Hersteller der PDF Betrachtungsprogramme manipulierte Schriftarten bisher kaum als Risiko bewerten.

Angriffsvektoren

Zunächst wurde mittels der Open Source Software FontForge [16] eine Times-New-Romans Schriftart manipuliert. Hierbei wurden Ziffern untereinander ausgetauscht, um eine Zahlenfolge von „9056283174“ auf „0123456789“ zu ändern. Für die Analyse wurden anschließend folgende Angriffsvektoren entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit Certification Signatur und dem Inhalt „Die Kontonummer lautet 9056283174“ erstellt. Durch den DocMDP Parameter `P 1` sind keine Änderungen am Dokument erlaubt. Anschließend wurde ein Incremental Update mit korrekter Cross-Reference Table und Trailer ans Dokument angehängt, welches die im Quelltext hinterlegte Schriftart durch die manipulierte Version ersetzt.
- Angriffsvektor 2 ist auf Approval Signaturbasis. Die restlichen Schritte entsprechen denen aus Angriffsvektor 1.

Angriffsvektor 1 wurde für Perfect PDF Reader und Soda PDF Desktop modifiziert.

Foxit Reader

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Die Signaturen sind allerdings nur in Version 9.7.0.29455 gültig, eine weitere Hinweismeldung erfolgt nicht. In Version 9.5.0.20723 führt der Angriffsvektor unter beiden Signaturtypen zu einer ungültigen Signatur.

LibreOffice Draw

Das PDF Betrachtungsprogramm zeigt eine gültige Signatur, mit der üblichen Hinweismeldung für Betrachtungsprogramme der Klasse 2, an. Allerdings wird keine der im PDF Dokument enthaltenen Schriftarten verwendet, weshalb der Angriff nicht erfolgreich ist.

Perfect PDF Reader

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Damit die Signaturen als gültig angezeigt werden, bedarf es dem folgenden zusätzlichen Eintrag in der Cross-Reference Table:

```
xx 1
0000000000 00000 n
```

`xx` muss jeweils der Nummer des Objekts vom Typ `SigRef` angepasst werden, welches dem Signatur-Reference-Dictionary entspricht. Durch Verweis auf Byte-Position 0, wird es vom Betrachtungsprogramm nicht mehr gefunden und dadurch die Restriktion umgangen. Anschließend werden beide Signaturen als gültig angezeigt, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2.

Soda PDF Desktop

Das PDF Betrachtungsprogramm setzt das Incremental Update für beide Angriffsvektoren um. Allerdings wird eine gültige Signatur für die Certification-Variante nur unter DocMDP Parameter `P 2` erreicht. Mit dieser Änderung sind beide Signaturen gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

6.6 Neue Incremental Saving Attacks

In diesem Abschnitt werden weitere Incremental Saving Attacks vorgestellt. Die Angriffe sind in Certification, Approval und der Kombination beider Signaturtypen unterteilt.
Die Tabelle 6.6 enthält die Ergebnisse zur Auswertung der Incremental Saving Attacks für Certification, Approval und der Kombination beider Signaturtypen. Insgesamt sind alle PDF Betrachtungsprogramme für mindestens einen Angriff anfällig. Die ISA-Angriffe auf Approval Signaturbasis liefern die erfolgreichste Quote innerhalb dieser Masterarbeit. Die Angriffe bei enthaltenen Certification Signatur entsprechen den Ergebnissen der DocMDP Auswertung. So sind auch hier die gleichen PDF Betrachtungsprogramme nicht angreifbar. Deutlich weniger erfolgreich zeigen sich die Angriffe auf PDF Dokumente, welche sowohl eine Certification als auch eine Approval Signatur beinhalteten. Dies kann daran liegen, dass der Angriff gleich zwei Signaturprüfungen überstehen muss.

Tabelle 6.6: Ergebnisse der neuen Incremental Saving Attacks.

Name	Version	Certification ISA	Approval ISA	Kombination* ISA
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	●	✓
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	●	✓
Expert PDF 14	14.0.25.3456 64-bit	◗	◗	◗
Foxit Reader	9.5.0.20723, 9.7.0.29455	●*	●	●
Foxit PhantomPDF	9.7.0.29478	●*	●	●
LibreOffice Draw	6.2.5.2 (x64)	◗	◗	✓
Master PDF Editor	5.4.22, 64 bit	●	●	✓
Nitro Pro	12.16.3.574	✓	◗	✓
Nitro Reader	5.5.9.2	✓	◗	✓
PDF Architect 7	7.0.26.3193 64-bit	◗	◗	◗
PDF Editor 6 Pro	6.5.0.3929	●	●	✓
PDFelement Pro	7.0.3.4309	●	●	✓
PDF-XChange Editor	8.0 (Build 331.0)	◗	◗	◗
Perfect PDF Reader	V14.0.9 (29.0)	◗	◗	✓
Perfect PDF 8 Reader	8.0.3.5	●	●	●
Perfect PDF 10 Premium	10.0.0.1	●	●	●
Power PDF Standard	3.0	✓	●	✓
Soda PDF Desktop	11.1.09.4184 64-bit	◗*	◗	✓
Anzahl angreifbarer PDF Betrachtungsprogramme:		13/18	18/18	7/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

*Angriffe nur unter Verwendung von DocMDP Parameter `P 2` oder `P 3` möglich.

6.6.1 Certification Signatur

Angriffsvektoren

Für die Analyse wurden folgende Angriffsvektoren entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit einer Certification Signatur erstellt. Durch den DocMDP Parameter `P 3`, darf nach dem Signieren ein Formular ausgefüllt, Signaturen angehängt und Annotations (bspw. ein Kommentar) hinzugefügt werden. Dem Dokument wurde per Incremental Update mit korrekter Cross-Reference Table und Trailer ein neues Inhaltsobjekt übergeben, das den Originaltext von „Hello World“ zu „Hello Attacker“

ändert.
- Angriffsvektor 2 entspricht dem ersten Angriffsvektor mit Änderung des DocMDP Parameter auf `P 1`. Dadurch sind keine Änderungen am Dokument erlaubt.
- Für den Angriffsvektor 3 wurde ein PDF Dokument mit Formularfeldern und einer Certification Signatur versehen. Durch den DocMDP Parameter `P 1`, sind keine Änderungen am Dokument erlaubt. Dem Dokument wurde per Incremental Update mit korrekter Cross-Reference Table und Trailer ein neues Inhaltsobjekt übergeben, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert. Darüber hinaus enthält das Update eine Inhaltsänderung am zweiten Formularfeld.
- Für den Angriffsvektor 4 wurde ein PDF Dokument mit einer Certification Signatur erstellt. Durch den DocMDP Parameter `P 1`, sind keine Änderungen am Dokument erlaubt. Dem Dokument wurden per Incremental Update mit korrekter Cross-Reference Table und Trailer folgende Objekte übergeben: `Pages` mit Referenz auf ein neues `Page`-Objekt, welches auf ein neues Inhaltsobjekt verweist, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert. Darüber hinaus wurde das Signatur-Field-Dictionary ins Update integriert und die Referenz aufs neue `Page`-Objekt umgelegt. Auf diese Weise wird ein komplett neuer Objektpfad integriert, während der alte Objektpfad ausgeblendet wird.
- Angriffsvektor 5 entspricht dem vierten Angriffsvektor, allerdings wurde die Reihenfolge wie in ISA-Variante 5 (siehe Abbildung 4.4) angepasst. Somit enthält das Incremental Update erst die Cross-Reference Table, danach den Trailer und schließt mit `%%EOF` das Dokument ab. Darauf folgend werden die in Angriffsvektor 4 beschriebenen Objekte angehängt. Da die Objekte erst nach `%%EOF` folgen, soll das Betrachtungsprogramm diese nicht als Update wahrnehmen, den Verarbeitungsprozess aber durch die korrekte Referenzierung in der Cross-Reference Table trotzdem erfolgreich durchführen.

Angriffsvektor 1-3 wurde für Perfect PDF Reader und Angriffsvektor 5 für Foxit Reader, Foxit PhantomPDF sowie Soda PDF Desktop modifiziert.
Insgesamt sind 13 von 18 PDF Betrachtungsprogramme für die Angriffe anfällig.

Foxit Reader und Foxit PhantomPDF

Für die PDF Betrachtungsprogramme ist nur der Angriff mit Angriffsvektor 5 erfolgreich. Die Signatur ist gültig, eine weitere Hinweismeldung erfolgt nicht. Der Angriffsvektor ist nur mit DocMDP Parameter `P 2` oder `P 3` möglich. Der gleiche Angriffsvektor unter Parameter `P 1` führt zu einer ungültigen Signatur.

LibreOffice Draw

Mit die PDF Betrachtungsprogramme lassen sich die Angriffsvektoren 1-3 erfolg-

reich validieren. Die Signaturen sind gültig, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2.

Master PDF Editor, PDF Editor 6 Pro, PDFelement Pro, Perfect PDF 8 Reader und Perfect PDF 10 Premium

Mit die PDF Betrachtungsprogramme lassen sich die Angriffsvektoren 1-3 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht

Perfect PDF Reader

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 1-3 erfolgreich validieren. Damit die Signaturen als gültig angezeigt werden, bedarf es dem folgenden zusätzlichen Eintrag in der Cross-Reference Table:

```
xx 1
0000000000 00000 n
```

`xx` muss jeweils der Nummer des Objekts vom Typ `SigRef` angepasst werden, welches dem Signatur-Reference-Dictionary entspricht. Durch Verweis auf Byte-Position 0, wird es vom Betrachtungsprogramm nicht mehr gefunden und dadurch die Restriktion umgangen. Anschließend werden die Signaturen als gültig angezeigt, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2.

Soda PDF Desktop

Für das PDF Betrachtungsprogramm ist nur der Angriff mit Angriffsvektor 5 erfolgreich. Die Signatur ist gültig, aber mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2. Der Angriffsvektor ist nur mit DocMDP Parameter `P 2` oder `P 3` möglich. Der gleiche Angriffsvektor unter Parameter `P 1` führt zu einer ungültigen Signatur.

6.6.2 Approval Signatur

Angriffsvektoren

Für die Analyse wurden folgende Angriffsvektoren entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit Formularfeldern und einer Approval Signatur versehen. Der gesetzte FieldMDP Parameter `All` erlaubt keine Änderungen an Formularfeldern. Dem Dokument wurde per Incremental Update mit korrekter Cross-Reference Table und Trailer ein neues Inhaltsobjekt übergeben, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert.

- Für den Angriffsvektor 2 wurde ein PDF Dokument mit Formularfeldern und einer Approval Signatur versehen. Der gesetzte FieldMDP Parameter `All` erlaubt keine Änderungen an Formularfeldern. Dem Dokument wurde per Incremental Update mit korrekter Cross-Reference Table und Trailer ein neues Inhaltsobjekt übergeben, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert. Darüber hinaus enthält das Update eine Textänderung am ersten Formularfeld.
- Für den Angriffsvektor 3 wurde ein PDF Dokument mit einer Approval Signatur erstellt. Dem Dokument wurden per Incremental Update mit korrekter Cross-Reference Table und Trailer folgende Objekte übergeben: `Pages` mit Referenz auf ein neues `Page`-Objekt, welches auf ein neues Inhaltsobjekt verweist, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert. Darüber hinaus wurde das Signatur-Field-Dictionary ins Update integriert und die Referenz aufs neue `Page`-Objekt umgelegt. Auf diese Weise wird ein komplett neuer Objektpfad integriert, während der alte Objektpfad ausgeblendet wird.
- Angriffsvektor 4 entspricht dem dritten Angriffsvektor, allerdings wurde zusätzlich ein leeres ungültiges Objekt mit der Objektnummer des alten Inhaltsobjekts hinzugefügt und referenziert. Darüber hinaus enthält das `Pages`-Objekt zwei `Kids`-Einträge für neues und altes `Page`-Objekt, während der Seitenzähler auf den Wert `1` gesetzt ist. Diese Änderungen sollen dazu führen, dass die Signaturvalidierung über den Original-Objektpfad durchgeführt wird, während der Verarbeitungsprozess lediglich den neuen Objektpfad verarbeitet und anzeigt. In Listing 6.7 ist das gesamte Incremental Update abgebildet. Das neue `Pages`-Objekt sowie das ungültige Objekt sind farblich markiert.
- Angriffsvektor 5 entspricht dem vierten Angriffsvektor, allerdings enthält das Dokument nun Formularfelder, welche durch den gesetzten FieldMDP Parameter `All` vor Änderungen geschützt werden sollen.
- Angriffsvektor 6 entspricht dem fünften Angriffsvektor, allerdings enthält das vormals leere ungültige Objekt nun einen öffnenden, aber nicht schließenden Stream.
- Angriffsvektor 7 entspricht dem dritten Angriffsvektor, allerdings wurde die Reihenfolge wie in ISA-Variante 5 (siehe Abbildung 4.4) angepasst. Somit enthält das Incremental Update erst die Cross-Reference Table, danach den Trailer und schließt mit `%%EOF` das Dokument ab. Darauf folgend werden die in Angriffsvektor 3 beschriebenen Objekte angehängt. Da die Objekte erst nach `%%EOF` folgen, soll das Betrachtungsprogramm diese nicht als Update wahrnehmen, den Verarbeitungsprozess aber durch die korrekte Referenzierung in der Cross-Reference Table trotzdem erfolgreich durchführen.

Es konnten für alle getesteten PDF Betrachtungsprogramme Schwachstellen ausgemacht werden.

```
4 0 obj
<</Count 1/Type/Pages/Kids[25 0 R 10 0 R]>>
endobj
12 0 obj
<</Rect[105.878 569.453 277.172 664.878]/Subtype/Widget/F 132/P 25 0 R/T(Signature2)/V 22 0 R
    /DA(/MyriadPro-Regular 0 Tf 0 Tz 0 g)/FT/Sig/Type/Annot/MK<<>>/AP<</N 13 0 R>>>>
endobj
21 0 obj endobj
24 0 obj
<<
/Length 48
>>
stream
BT
/F1 12 Tf
100 700 Td
(Hello Attacker!) Tj
ET

endstream
endobj
25 0 obj
<</CropBox[0.0 0.0 612.0 792.0]/Annots 11 0 R/Parent 4 0 R/Contents 24 0 R/Rotate 0/MediaBox
    [0.0 0.0 612.0 792.0]/Resources<</Font<</T1_0 20 0 R>>/ProcSet[/PDF/Text]>>/Type/Page>>
endobj
xref
0 1
0000000000 65535 f
4 1
0000068121 00000 n
12 1
0000068180 00000 n
21 1
0000068370 00000 n
24 1
0000068386 00000 n
25 1
0000068486 00000 n
trailer
<</Size 26/Root 8 0 R/Info 4 0 R/Prev 116>>
startxref
68682
%%EOF
```

Listing 6.7: Incremental Update des ISA-Angriffsvektor 4 für Approval Signaturen.

Adobe Acrobat Reader DC und Adobe Acrobat Pro 2017

Mit den PDF Betrachtungsprogrammen lassen sich die Angriffsvektoren 4-6 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht. Allerdings besteht die Einschränkung, dass ein Klick auf den sichtbaren Teil der Signatur zu einer Fehlermeldung führt (siehe Abbildung 6.8).

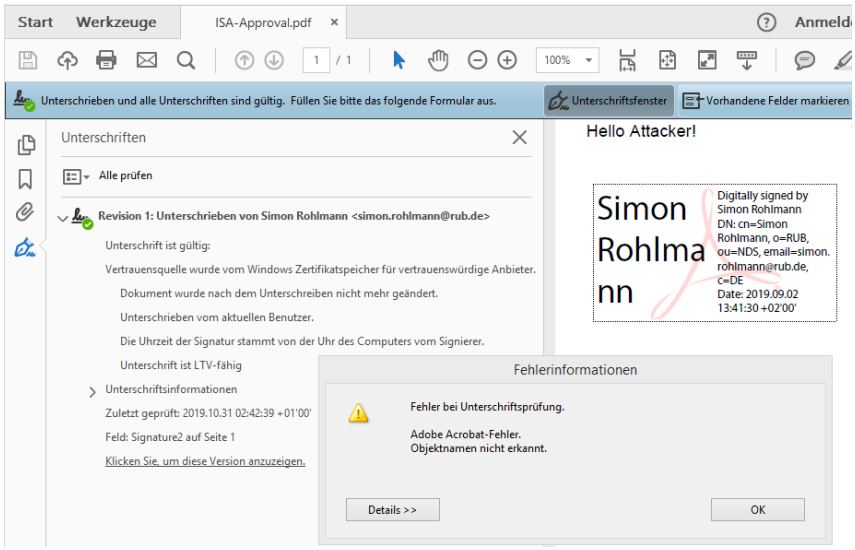

Abbildung 6.8: ISA-Angriffsvektor 4: Fehlermeldung unter Adobe Acrobat Reader DC und Adobe Acrobat Pro 2017.

Foxit Reader und Foxit PhantomPDF

Mit den PDF Betrachtungsprogrammen ist nur der Angriff mit Angriffsvektor 7 erfolgreich. Die Signatur ist gültig, eine weitere Hinweismeldung erfolgt nicht.

LibreOffice Draw

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 1 und 2 erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

Master PDF Editor, PDF Editor 6 Pro, PDFelement Pro und Perfect PDF Reader

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 1 und 2 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht.

Nitro Pro

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 1-4 und 7 erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

Nitro Reader

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 3, 4 und 7

erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

Perfect PDF 8 Reader und Perfect PDF 10 Premium

Mit den PDF Betrachtungsprogrammen lassen sich die Angriffsvektoren 1, 2 und 7 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht.

Power PDF Standard

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 3 und 7 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht. Eine ebenfalls gültige Signatur liefern die Angriffsvektoren 4-6, allerdings wird für diese Angriffsvektoren die Hinweismeldung: „Signature is valid: The revision of the document has not been altered; There have been subsequent changes to the document“ ausgegeben.

Soda PDF Desktop

Für das PDF Betrachtungsprogramm ist nur der Angriff mit Angriffsvektor 7 erfolgreich. Die Signatur ist gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

6.6.3 Kombination beider Signaturtypen

Angriffsvektoren

Für die Analyse wurden folgende Angriffsvektoren entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit Formularfeldern, einem leerem Signaturfeld und einer Certification Signatur versehen. Durch den DocMDP Parameter `P 2`, bleibt das Ausfüllen von Formularen und das Hinzufügen weiterer Signaturen erlaubt. Im nächsten Schritt wurde dem Dokument eine Approval Signatur, mit FieldMDP Parameter `All` zum Schutz aller Felder, hinzugefügt. Im anschließenden Incremental Update, mit korrekter Cross-Reference Table und Trailer, wurden folgende Objekte übergeben: `Pages`, `Page`, leeres ungültiges Objekt, Signatur-Field-Dictionary, Inhaltsobjekt. `Pages` enthält als `Kids`-Eintrag das neue und alte `Page`-Objekt, aber mit dem Seitenzählerwert `1`. Das Signatur-Field-Dictionary referenziert nun auf das neue `Page`-Objekt. Das neue Inhaltsobjekt enthält die Zeichenkette „Hello Attacker“. Die Objektnummer des leeren ungültigen Objekts entspricht der Nummer des alten Inhaltsobjekts mit der Zeichenkette „Hello World“. Diese Änderungen sollen dazu führen, dass die Signaturvalidierung über den Original-Objektpfad durchgeführt wird, während der Verarbeitungsprozess lediglich den neuen Objektpfad verarbeitet und anzeigt.

- Für den Angriffsvektor 2 wurde ein PDF Dokument mit einer Certification Signatur und leerem Signaturfeld erstellt. Durch den DocMDP Parameter `P 2`, bleibt das Ausfüllen von Formularen und das Hinzufügen weiterer Signaturen erlaubt. Im nächsten Schritt wurde dem Dokument eine Approval Signatur, mit FieldMDP Parameter `Include` zum Schutz des Signaturfelds, hinzugefügt. Im anschließenden Incremental Update, mit korrekter Cross-Reference Table und Trailer, wurden folgende Objekte übergeben: `Pages` mit Referenz auf das neue `Page`-Objekt, welches auf ein neues Inhaltsobjekt verweist, das den Originaltext von „Hello World“ zu „Hello Attacker“ ändert. Darüber hinaus wurde das Signatur-Field-Dictionary ins Update integriert und die Referenz aufs neue `Page`-Objekt umgelegt. Auf diese Weise wird ein komplett neuer Objektpfad integriert, während der alte Objektpfad ausgeblendet wird. Allerdings wurde die Reihenfolge wie in ISA-Variante 5 (siehe Abbildung 4.4) angepasst. Somit enthält das Incremental Update erst die Cross-Reference Table, danach den Trailer und schließt mit `%%EOF` das Dokument ab. Darauf folgend werden die beschriebenen Objekte angehängt. Da die Objekte erst nach `%%EOF` folgen, soll das Betrachtungsprogramm diese nicht als Update wahrnehmen, den Verarbeitungsprozess aber durch die korrekte Referenzierung in der Cross-Reference Table trotzdem erfolgreich durchführen.
- Angriffsvektor 3 entspricht dem zweiten Angriffsvektor, allerdings enthält das Dokument nun zusätzliche Formularfelder, welche durch den FieldMDP Parameter `All` geschützt werden.

Insgesamt sind 7 von 18 PDF Betrachtungsprogramme für die Angriffe anfällig.

Foxit Reader und Foxit PhantomPDF

Die PDF Betrachtungsprogramme setzen das Incremental Update nur für Angriffsvektor 2 erfolgreich um. Die Signaturen sind gültig. Für die Certification Signatur wird folgender Hinweis ausgegeben: „Document certification is valid: This document has been modified since being certified, but these modifications are allowed and do not invalidate the signature.“. Dieser Hinweis wird allerdings auch bei einem Dokument mit zwei angewendeten Signaturen ausgegeben, ohne dass eine Manipulation stattgefunden. Eine Hinweismeldung für die Approval Signatur folgt nicht.

6.7 Shadow Attacks

In diesem Abschnitt werden die Angriffsvektoren auf Basis der in Abschnitt 4.4 eingeführten Shadow Attack Klasse vorgestellt. Die Angriffsvektoren wurden von Vladislav Mladenov und Christian Mainka auf Basis von Approval Signaturen entwickelt und im Rahmen dieser Masterarbeit auf PDF Dokumente mit Certification Signatur übertragen.

Die Tabelle 6.9 enthält die Ergebnisse zur Auswertung der Shadow Attacks für Certification und Approval Signaturen. Insgesamt sind 17 von 18 PDF Betrachtungsprogramme für mindestens einen Angriff anfällig. Auch bei den Shadow Attacks zeigt sich eine Verschiebung der Erfolgsquote zwischen Certification und Approval Signaturen. Wobei es sich bis auf den Perfect PDF Reader, um die bereits identifizierten Betrachtungsprogramme handelt, welche Modifikationen bei Certification Signaturen generell verbieten.

Tabelle 6.9: Ergebnisse der Shadow Attacks.

Name	**Version**	**Certification Shadow Attack**	**Approval Shadow Attack**
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	●
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	●
Expert PDF 14	14.0.25.3456 64-bit	◗	◗
Foxit Reader	9.5.0.20723, 9.7.0.29455	●	●
Foxit PhantomPDF	9.7.0.29478	●	●
LibreOffice Draw	6.2.5.2 (x64)	◗	◗
Master PDF Editor	5.4.22, 64 bit	●	●
Nitro Pro	12.16.3.574	✓	◗
Nitro Reader	5.5.9.2	✓	◗
PDF Architect 7	7.0.26.3193 64-bit	◗	◗
PDF Editor 6 Pro	6.5.0.3929	●	●
PDFelement Pro	7.0.3.4309	●	●
PDF-XChange Editor	8.0 (Build 331.0)	◗	◗
Perfect PDF Reader	V14.0.9 (29.0)	✓	◗
Perfect PDF 8 Reader	8.0.3.5	●	●
Perfect PDF 10 Premium	10.0.0.1	●	●
Power PDF Standard	3.0	✓	✓
Soda PDF Desktop	11.1.09.4184 64-bit	◗	◗
Anzahl angreifbarer PDF Betrachtungsprogramme:		12/18	17/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

Angriffsvektoren

Für die Analyse wurden folgende Angriffsvektoren auf Approval und Certification Signaturbasis entwickelt:

- Für den Angriffsvektor 1 wurde ein PDF Dokument mit Inhalt `X` im Objektpfad

`1` erstellt. Darüber hinaus wurde ein weiterer Objektpfad `2` mit Inhalt `Y` hinzugefügt. Der Objektpfad `2` beinhaltet ein `Catalog`, ein `Pages`, ein `Page`-Objekt sowie ein Inhaltsobjekt. Die Objektnummer des `Catalog` und `Pages`-Objekt entsprechen denen des Objektpfad `1`, sind aber über die Cross-Reference Table mit unterschiedlichen und nicht genutzten Objektnummern referenziert. `Pages` referenziert auf das `Page`-Objekt, welches wiederum auf das Inhaltsobjekt referenziert. Außerdem wird das `Page`-Objekt und Inhaltsobjekt korrekt über die Cross-Reference Table referenziert. D.h. in dieser Konstellation wird über das PDF Betrachtungsprogramm der Objektpfad `1` mit Inhalt `X` ausgegeben. Nach der Erstellung wurde das Dokument digital signiert. Anschließend wird über ein Update der Cross-Reference Table das `Pages`-Objekt aus Objektpfad `1` mit dem `Pages`-Objekt aus Objektpfad `2` ersetzt. Dieses Update bewirkt, dass das PDF Betrachtungsprogramm nun den Objektpfad `2` mit Inhalt `Y` ausgibt und Objektpfad `1` ausblendet.

- Angriffsvektor 2 entspricht dem ersten Angriffsvektor, allerdings wurde die Dokumentenbasis getauscht und die Signierung durch Acrobat Web Capture 15.0 / Adobe Acrobat 17.0, statt durch den *PDFSigner*, vorgenommen.
- Angriffsvektor 3 entspricht dem zweiten Angriffsvektor, allerdings ohne sichtbaren Signaturteil im Dokument.
- Für Angriffsvektor 4 wurde ein PDF Dokument mit Inhalt `Y` erstellt. Darüber hinaus wurde über den sichtbaren Teil des Inhalts `Y` eine Bilddatei vom `SubType Image` gelegt, welche den Inhalt `X` besitzt. D.h. in dieser Konstellation ist über das PDF Betrachtungsprogramm nur der Inhalt `X` sichtbar. Nach der Erstellung wurde das Dokument digital signiert. Anschließend wird über ein Update der Cross-Reference Table das `Image`-Objekt mit folgendem Eintrag ersetzt: `21 0 obj «/Subtype/XML/Type/Metadata» endobj`. Dies führt dazu, dass der Inhalt `X` ausgeblendet wird und somit der Inhalt `Y` wieder sichtbar ist.

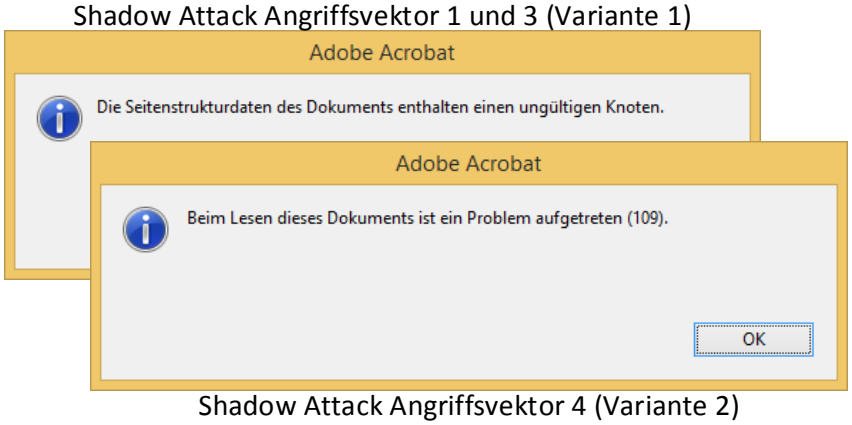

Abbildung 6.10: Shadow Attack Angriffsvektoren: Fehlermeldung unter Adobe Acrobat Reader DC und Adobe Acrobat Pro 2017.

Adobe Acrobat Reader DC und Adobe Acrobat Pro 2017

Mit den PDF Betrachtungsprogrammen lassen sich die Approval-Angriffsvektoren 1, 3 und 4 erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht. Angriffsvektor 2 ist nur in UI-Layer 1 gültig. UI-Layer 2 bleibt leer. Beim Ausführen der manuellen Signaturvalidierung treten die in Abbildung 6.10 dargestellten Fehlermeldungen auf. Die Certification-Angriffsvektoren führen zu einer ungültigen Signatur.

Foxit Reader und Foxit PhantomPDF

Mit den PDF Betrachtungsprogrammen lassen sich die Angriffsvektoren 1 und 2 beider Signaturtypen erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht.

LibreOffice Draw

Mit dem PDF Betrachtungsprogramm lässt sich der Angriffsvektor 1 beider Signaturtypen erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben. Approval-Angriffsvektoren 2 und 3, sowie Certification-Angriffsvektor 4 sind ebenfalls erfolgreich und geben eine gültige Signatur, mit dem üblichen Hinweis für PDF Betrachtungsprogramme der Klasse 2, aus.

Master PDF Editor

Mit dem PDF Betrachtungsprogramm lassen sich die Angriffsvektoren 1-3 beider Signaturtypen erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht. Ebenfalls erfolgreich ist der Certification-Angriffsvektor 4, welcher auch zu einer gültigen Signatur führt.

Nitro Pro und Nitro Reader

Mit den PDF Betrachtungsprogrammen lassen sich die Angriffsvektoren 1-3 unter der Approval Signatur erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben. Die Certification-Angriffsvektoren führen zu einer ungültigen Signatur.

PDF Editor 6 Pro und PDFelement Pro

Mit den PDF Betrachtungsprogrammen lassen sich die Angriffsvektoren 1-4 unter der Approval Signatur sowie 1 und 4 unter der Certification Signatur erfolgreich validieren. Die Signaturen sind gültig, eine weitere Hinweismeldung erfolgt nicht.

Perfect PDF Reader

Mit dem PDF Betrachtungsprogramm lässt sich nur der Angriffsvektor 1 unter Approval Signaturbasis erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

Soda PDF Desktop

Mit dem PDF Betrachtungsprogramm lässt sich nur der Angriffsvektor 2 unter beiden Signaturtypen erfolgreich validieren. Die Signaturen sind gültig, aber es wird der übliche Hinweis für PDF Betrachtungsprogramme der Klasse 2 ausgegeben.

6.8 Zusammenfassung der Ergebnisse

Die Tabelle 6.11 enthält die Zusammenfassung der Ergebnisse aus den Abschnitten 6.4, 6.5, 6.6 und 6.7. Die verschiedenen Signaturtypen sind jeweils in einer Spalte zusammengefasst.
In den Ergebnissen lassen sich für die PDF Betrachtungsprogramme Adobe Acrobat Reader DC, Adobe Acrobat Pro 2017, Nitro Pro, Nitro Reader und Power PDF Standard deutliche Unterschiede bei den verschiedenen Signaturtypen feststellen. Während für diese Betrachtungsprogramme Angriffe auf Approval Signaturen jeweils in mindestens zwei Fällen möglich waren, blieben sämtliche Angriffe auf Certification Signaturbasis erfolglos. Dies ließ sich auch nicht, wie bei den Betrachtungsprogrammen Foxit Reader, Foxit PhantomPDF und Soda PDF Desktop, durch einen angepassten DocMDP Parameter ausgleichen. Diese Einschränkung spiegelt sich mit 13 von 18 angreifbaren PDF Betrachtungsprogrammen in den Ergebnissen wieder.
Insgesamt zeigt sich bei den Ergebnissen zu Angriffen auf Transformationsmethoden, dass die Umsetzung der Restriktionen kaum oder gar nicht erfolgt. Obwohl der Foxit Reader und Foxit PhantomPDF die Modifikationen im Detail erkennen, bleibt die Signatur im Gegensatz zu den Adobe Produkten gültig. Andere Betrachtungsprogramme, wie bspw. Expert PDF 14, PDF Architect 7, PDF-XChange Editor oder Soda PDF Desktop, liefern hingegen nur eine allgemeine Hinweismeldung auf Modifikationen nach dem Signieren. Was genau manipuliert bzw. geändert wurde, erfährt der Betrachter des PDF Dokuments nicht.
Im Font-Masking Bereich sticht vor allem der Foxit Reader heraus. In Version 9.5.0.20723 gibt das Betrachtungsprogramm die Signatur als ungültig aus. In der neueren Version 9.7.0.29455 wird die manipulierte Schriftart ohne jede Modifikationsmeldung akzeptiert und die Signatur als gültig validiert. Neben den Adobe Programme ist auch LibreOffice Draw nicht für den Angriff anfällig. LibreOffice Draw verwendet keine der beiden im PDF Dokument integrierten Schriftarten und ist daher immun gegen diese Art von Manipulation.

Die ISA-Angriffe unter Verwendung einer Approval Signatur verliefen bei allen Betrachtungsprogrammen erfolgreich und boten neben den Shadow Attacks eine von zwei erfolgreichen Angriffen gegen die Adobe Betrachtungsprogramme. Die Shadow Attacks lieferten, bis auf den Power PDF Standard, die gleichen Ergebnisse hinsichtlich des Manipulationserfolgs, wie bei den ISA-Angriffen.
Am Stärksten anfällig für sämtliche Angriffe zeigen sich vor allem die PDF Betrachtungsprogramme Master PDF Editor, PDF Editor 6 Pro, PDFelement Pro, Perfect PDF 8 Reader und Perfect PDF 10 Premium. Die aufgezählten Programme gaben für die Angriffe eine gültige Signatur aus und wiesen gleichzeitig in keinem dieser Fälle auf irgendeine Form von Manipulation hin.

Tabelle 6.11: Zusammenfassung der Evaluationsergebnisse.

Name	**Version**	**Doc-MDP**	**Field-MDP**	**Font-Masking**	**ISA**	**Shadow Attack**
Adobe Acrobat Reader DC	2019.012.20035, 2019.021.20049	✓	✓	✓	●	●
Adobe Acrobat Pro 2017	2017.011.30143, 2017.011.30152	✓	✓	✓	●	●
Expert PDF 14	14.0.25.3456 64-bit	◗	◗	◗	◗	◗
Foxit Reader	9.5.0.20723, 9.7.0.29455	◗	◗	●*	●	●
Foxit PhantomPDF	9.7.0.29478	◗	◗	●	●	●
LibreOffice Draw	6.2.5.2 (x64)	◗	◗	✓	◗	◗
Master PDF Editor	5.4.22, 64 bit	●	●	●	●	●
Nitro Pro	12.16.3.574	✓	◗	◗	◗	◗
Nitro Reader	5.5.9.2	✓	✓	◗	◗	◗
PDF Architect 7	7.0.26.3193 64-bit	◗	◗	◗	◗	◗
PDF Editor 6 Pro	6.5.0.3929	●	●	●	●	●
PDFelement Pro	7.0.3.4309	●	●	●	●	●
PDF-XChange Editor	8.0 (Build 331.0)	◗	◗	◗	◗	◗
Perfect PDF Reader	V14.0.9 (29.0)	◗	◗	◗	◗	◗
Perfect PDF 8 Reader	8.0.3.5	●	●	●	●	●
Perfect PDF 10 Premium	10.0.0.1	●	●	●	●	●
Power PDF Standard	3.0	✓	◗	●	●	✓
Soda PDF Desktop	11.1.09.4184 64-bit	◗	◗	◗	◗	◗
Anzahl angreifbarer PDF Betrachtungsprogramme:		13/18	15/18	15/18	18/18	17/18

✓ Das Programm ist für diese Attacke nicht anfällig.
◗ Das Programm ist für diese Attacke anfällig, aber es wird eine Hinweismeldung ausgegeben.
● Das Programm ist ohne Einschränkung für diese Attacke anfällig.

*Font-Masking Angriff nur unter Version 9.7.0.29455 erfolgreich.

6.9 Weitere Erkenntnisse

Während des Entstehungsprozesses der Masterarbeit, ergaben sich weitere Ergebnisse und Erkenntnisse im Zusammenhang mit digitalen Signaturen in PDF Dokumenten, welche im Folgenden dargestellt werden.

6.9.1 ISA per Zwischenperson

Ein von den Angreifermodellen in Abschnitt 3.1 nicht berücksichtigter Fall tritt auf, falls ein Angreifer die Manipulation zwischen zwei Signaturen durchführt. Erhält der Angreifer ein PDF Dokument mit Certification Signatur zur Signierung vorgelegt, kann er eine Manipulation auf ISA-Basis durchführen und das Dokument anschließend per Approval Signatur unterschreiben. Bei Klasse 2 Betrachtungsprogrammen lässt sich die nachträgliche Änderung auf diese Weise verschleiern, da für den Certification Teil ohnehin eine Hinweismeldung durch die zweite Signatur erfolgt. Der Approval Teil bleibt frei von einer Hinweismeldung und die Darstellung ist nicht von einem Dokument ohne ISA zu unterscheiden. Ein Beispiel für die Darstellung innerhalb eines PDF Betrachtungsprogramms der Klasse 2 findet sich in Abbildung 6.12.

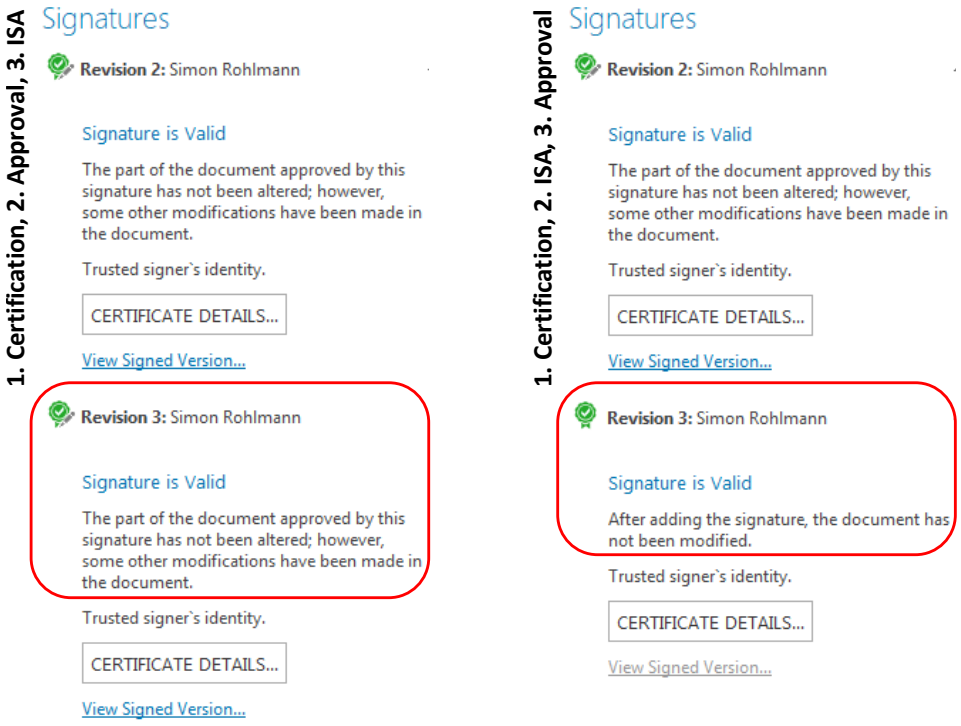

Abbildung 6.12: ISA per Zwischenperson im PDF Betrachtungsprogramm der Klasse 2.

6.9.2 Certification Signatur entfernen

Für PDF Dokumente, die sowohl eine Certification als auch eine Approval Signatur enthalten, wurde unter den Adobe Programmen keine Möglichkeit gefunden eine Manipulation auf ISA-Basis durchzuführen, ohne die Certification Signatur zu invalidieren. Allerdings besteht eine Möglichkeit die Certification Signatur auszublenden, ohne die Approval Signatur zu invalidieren. Die Basis bildet der Angriffsvektor 4 aus Abschnitt 6.6.2 mit dem Unterschied, dass das Dokument zusätzlich eine Certification Signatur enthält. Der `Annots`-Eintrag innerhalb des neuen `Page`-Objekts wird auf ein neues Objekt referenziert, welches alle Annotations des Dokuments, mit Ausnahme der Certification Annotation, enthält. Dadurch wird der sichtbare Teil der Certification Signatur ausgeblendet. Im UI-Layer 2 erscheint, neben der Approval Signatur, ein Hinweis auf eine gelöschte Signatur. Es wurde keine Möglichkeit gefunden diesen Eintrag zu entfernen, allerdings ist es möglich den UI-Layer 2 auszublenden. Der UI-Layer 1 bleibt dabei nach wie vor aktiv. Um den UI-Layer 2 auszublenden, muss ein neues Objekt erzeugt werden, welches für die `AcroForms`-Objekte innerhalb des Dokuments zuständig ist. In diesem Objekt wird der Wert des `SigFlags`-Eintrags von `3` auf `0` geändert. Im UI-Layer 2 werden anschließend keine Signaturen mehr aufgeführt, siehe hierzu Abbildung 6.13.

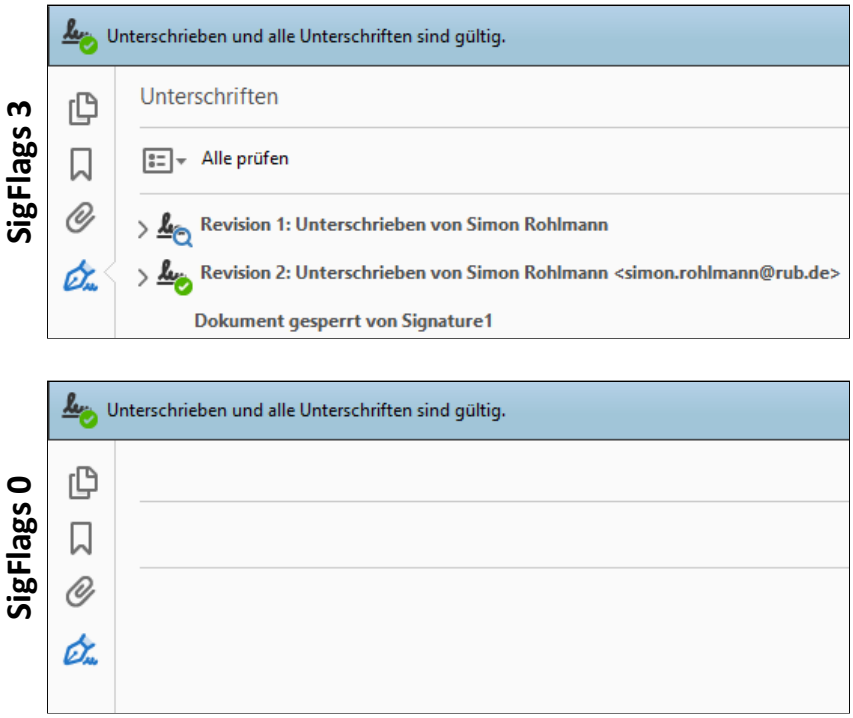

Abbildung 6.13: Adobe UI-Layer 2 nachdem die Certification Signatur entfernt wurde.

7 Fazit und Ausblick

In diesem Kapitel werden die entstandenen Ergebnisse reflektiert und kritisch bewertet. Darüber hinaus wird ein Ausblick zu weiteren Arbeiten dargestellt.

7.1 Fazit

Signierte PDF Dokumente sind weltweit stark verbreitet und finden in verschiedenen Bereichen der Wirtschaft und Verwaltung Anwendung. Aus diesem Grund sollte die Sicherheit der digitalen Signaturen in PDF Dokumenten einen hohen Stellenwert einnehmen.
Wie die Re-Evaluation der bereits veröffentlichten Exploits zeigt, haben viele Hersteller die Schwachstellen in ihren Produkten behoben. Allerdings ist auch ersichtlich, dass bei einigen Herstellern die gleichen Schwachstellen, trotz neuer Programmversionen, weiterhin ausnutzbar sind.
Die Evaluation der Exploits zu Transformationsmethoden zeigt darüber hinaus deutlich auf, dass die PDF Spezifikation nicht vollumfänglich durch die verschiedenen PDF Betrachtungsprogramme umgesetzt wird. So werden Einschränkungen durch die Transformationsparameter lediglich bei 3 von 18 Betrachtungsprogrammen durchgängig korrekt umgesetzt.
Für die Exploits der Font-Masking Kategorie zeigt sich ein ähnliches Bild, wie für die Transformationsmethoden. So bewerten die PDF Betrachtungsprogramme nachträgliche Schriftarten kaum als Risiko. Dies führt dazu, dass 15 der 18 Betrachtungsprogramme manipulierte Schriftarten erfolgreich verarbeiten.
Als besonders bedenklich sind die Ergebnisse der Kategorie Incremental Saving Attacks einzustufen. Erlangt ein Angreifer Zugriff auf ein signiertes PDF Dokument kann er es nach belieben verändern. Dies ist vergleichbar mit einem unterschriebenen Blankoscheck, welcher sich immer wieder neu ausstellen lässt. Dass alle getesteten PDF Betrachtungsprogramme von mindestens einem ISA-Exploit direkt betroffen sind, zeigt aber auch die Schwierigkeit des Validierungsprozesses. So müssen nicht nur die kryptographischen Eigenschaften einer digitalen Signatur geprüft werden, sondern auch verschiedene Konstellationen von erlaubten nachträglichen Änderungen berücksichtigt werden. Durch die komplexere Validierung, steigt auch die Anzahl der möglichen Fehlerquellen.
Die neu eingeführte Shadow Attack Klasse zeigt außerdem, dass eine kritische Betrachtung nicht nur bei bereits signierten PDF Dokumenten von Bedeutung ist. So

sind empfindliche Manipulationen auch vor dem Signierungsprozess möglich und machen 17 von 18 getesteten PDF Betrachtungsprogrammen angreifbar.
Schlussendlich zeigen die Ergebnisse dieser Arbeit, dass digitale Signaturen in PDF Dokumenten nach wie vor keinen ausreichenden Schutz bieten, der bei Anbetracht von Verbreitung und Anwendung angemessen wäre.

7.2 Ausblick

Für die PDF Betrachtungsprogramme von Adobe wurden zwei ungültige Objekte identifiziert, welche die Überprüfung der angefügten Incremental Updates durch die Validierungslogik unterbrochen haben und somit Angriffe auf ISA-Basis ermöglichten. Allerdings betreffen die genannten Objekte nur die Sicherheit von Approval signaturbasierten PDF Dokumenten. In einer späteren Arbeit könnten ähnliche Methoden erforscht werden, um Incremental Saving Attacks auf PDF Dokumente mit Certification Signatur auszuweiten, welche ebenfalls die von Adobe integrierte Überprüfung umgehen.
In dieser Arbeit wurden Methoden vorgestellt, um mittels manipulierter Schriftarten den Dokumentinhalt visuell zu verändern. Ein ähnliches Themengebiet bilden Manipulation an den UI-Layern, welche bereits von Mladenov et al. [35, S. 12] thematisiert wurden. Die Manipulationen sollten direkt durch ein modifiziertes PDF Dokument ausgelöst werden. Auf diese Weise könnten dem Opfer Signaturen beliebiger Personen vorgetäuscht werden.
Im Paper „Practical Decryption exFiltration: Breaking PDF Encryption“ präsentieren Müller et al. [19] Möglichkeiten, um an den Inhalt von verschlüsselten PDF Dokumenten zu gelangen. Ein weiteres Forschungsfeld ergibt sich aus der Kombination von verschlüsselten und gleichzeitig signierten PDF Dokumenten. So könnten Möglichkeit geprüft werden, die es erlauben, bspw. über Incremental Saving Attacks, neuen Inhalt ins Dokument einzuschleusen, ohne dabei die Signatur zu invalidieren.

A Anhang

Im Anhang findet sich der Quelltext für Beispiele von digital signierten PDF Dokumenten. Jeweils ein Listing für Dokumente mit Certification bzw. Approval Signatur. Die charakteristischen Ausprägungen der enthaltenen Transformationsmethoden DocMDP und FieldMDP sind farblich markiert. Darüber hinaus finden sich in diesem Kapitel Abbildungen der Benutzeroberfläche zum entwickelten *PDF Tester* Programm aus Kapitel 5.

A.1 Beispieldokument mit Certification Signatur

```
%PDF-1.7
%
1 0 obj
<< /Type /Catalog
/Pages 2 0 R
>>
endobj
2 0 obj
<< /Type /Pages
/Kids [3 0 R]
/Count 1
/MediaBox [0 0 612 792]
>>
endobj
3 0 obj
<<  /Type /Page
/Parent 2 0 R
/Resources
<< /Font
<< /F1
<< /Type /Font
/Subtype /Type1
/BaseFont /Times-Roman
>>
>>
>>
/Contents 4 0 R
>>
endobj
4 0 obj
<<
/Length 47
>>
stream
BT
```

```
/F1 12 Tf
100 700 Td
(Hello World!!!) Tj
ET
endstream
endobj
xref
0 5
0000000000 65535 f
0000000019 00000 n
0000000077 00000 n
0000000177 00000 n
0000000455 00000 n
trailer
<<  /Root 1 0 R
/Size 5
>>
startxref
553
%%EOF

1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
/AcroForm <<
/Fields [5 0 R]
/DA (/Helv 0 Tf 0 g )
/DR 6 0 R
/SigFlags 3
>>
/Perms 7 0 R
>>
endobj
5 0 obj
<<
/FT /Sig
/Type /Annot
/Subtype /Widget
/F 132
/T (Signature1)
/TU (Digital Signature)
/P 3 0 R
/Rect [150.0 400.0 450.0 480.0]
/V 8 0 R
/AP <<
/N 9 0 R
>>
>>
endobj
6 0 obj
<<
/Font 10 0 R
>>
endobj
7 0 obj
<<
/DocMDP 8 0 R
>>
endobj
3 0 obj
<<
```

```
/Type /Page
/Parent 2 0 R
/Resources <<
/Font <<
/F1 <<
/Type /Font
/Subtype /Type1
/BaseFont /Times-Roman
>>
>>
>>
/Contents 4 0 R
/Annots [5 0 R]
>>
endobj
8 0 obj
<<
/Type /Sig
/Reference [11 0 R]
/Filter /Adobe.PPKLite
/SubFilter /adbe.pkcs7.detached
/Name (PDF Insecurity Team)
/Location (Bochum)
/Reason (PDF Security Testing)
/M (D:20191025180839+02'00')
/Contents <...>
/ByteRange [0 1495 20441 35027]
>>
endobj
9 0 obj
<<
/Length 55
/Type /XObject
/Subtype /Form
/Resources <<
/XObject <<
/Im1 12 0 R
>>
>>
/FormType 1
/BBox [0.0 0.0 300.0 80.0]
>>
stream
0.25 0 0 0.25 0 0 cm
q
1079 0 0 150 0 0 cm
/Im1 Do
Q
Q

endstream
endobj
10 0 obj
<<
/Helv 13 0 R
/ZaDb 14 0 R
>>
endobj
11 0 obj
<<
/Type /SigRef
/DigestMethod /SHA1
```

```
/TransformParams 15 0 R
/TransformMethod /DocMDP
>>
endobj
12 0 obj
<<
/Length 33239
/Type /XObject
/Subtype /Image
/Filter /FlateDecode
/BitsPerComponent 8
/Width 1079
/Height 150
/ColorSpace /DeviceRGB
/DecodeParms 16 0 R
/SMask 17 0 R
>>
stream
...
endstream
endobj
13 0 obj
<<
/Type /Font
/Subtype /Type1
/BaseFont /Helvetica
/Encoding /WinAnsiEncoding
>>
endobj
14 0 obj
<<
/Type /Font
/Subtype /Type1
/BaseFont /ZapfDingbats
>>
endobj
15 0 obj
<<
/Type /TransformParams
/V /1.2
/P 1
>>
endobj
16 0 obj
<<
/BitsPerComponent 8
/Predictor 15
/Columns 1079
/Colors 3
>>
endobj
17 0 obj
<<
/Length 180
/Type /XObject
/Subtype /Image
/Filter /FlateDecode
/BitsPerComponent 8
/Width 1079
/Height 150
/ColorSpace /DeviceGray
>>
```

```
stream
...
endstream
endobj
xref
0 2
0000000000 65535 f
0000000729 00000 n
3 1
0000001112 00000 n
5 13
0000000867 00000 n
0000001043 00000 n
0000001077 00000 n
0000001281 00000 n
0000020500 00000 n
0000020720 00000 n
0000020768 00000 n
0000020875 00000 n
0000054323 00000 n
0000054421 00000 n
0000054495 00000 n
0000054553 00000 n
0000054633 00000 n
trailer
<<
/Root 1 0 R
/Size 18
/ID [<24BFBA9B0694AAC0B79AC7CB34630B25> <24BFBA9B0694AAC0B79AC7CB34630B25>]
/Prev 553
>>
startxref
54987
%%EOF
```

Listing A.1: Beispieldokument mit Certification Signatur und DocMDP *P 1* Parameter (Zeile 123, 178 und 223 gekürzt).

A.2 Beispieldokument mit Approval Signatur

```
%PDF-1.7
%
1 0 obj
<< /Type /Catalog
/Pages 2 0 R
>>
endobj
2 0 obj
<< /Type /Pages
/Kids [3 0 R]
/Count 1
/MediaBox [0 0 612 792]
>>
endobj
3 0 obj
<< /Type /Page
```

```
/Parent 2 0 R
/Resources
<< /Font
<< /F1
<< /Type /Font
/Subtype /Type1
/BaseFont /Times-Roman
>>
>>
>>
/Contents 4 0 R
>>
endobj
4 0 obj
<<
/Length 47
>>
stream
BT
/F1 12 Tf
100 700 Td
(Hello World!!!) Tj
ET
endstream
endobj
xref
0 5
0000000000 65535 f
0000000019 00000 n
0000000077 00000 n
0000000177 00000 n
0000000455 00000 n
trailer
<<  /Root 1 0 R
/Size 5
>>
startxref
553
%%EOF

1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
/AcroForm <<
/Fields [5 0 R]
/DA (/Helv 0 Tf 0 g )
/DR 6 0 R
/SigFlags 3
>>
>>
endobj
5 0 obj
<<
/FT /Sig
/Type /Annot
/Subtype /Widget
/F 132
/T (Signature1)
/TU (Digital Signature)
/P 3 0 R
/Rect [150.0 400.0 450.0 480.0]
```

```
/Lock 7 0 R
/V 8 0 R
/AP <<
/N 9 0 R
>>
>>
endobj
6 0 obj
<<
/Font 10 0 R
>>
endobj
3 0 obj
<<
/Type /Page
/Parent 2 0 R
/Resources <<
/Font <<
/F1 <<
/Type /Font
/Subtype /Type1
/BaseFont /Times-Roman
>>
>>
>>
/Contents 4 0 R
/Annots [5 0 R]
>>
endobj
7 0 obj
<<
/Action /All
/Type /SigFieldLock
/P 1
>>
endobj
8 0 obj
<<
/Type /Sig
/Reference [11 0 R]
/All 7 0 R
/Filter /Adobe.PPKLite
/SubFilter /adbe.pkcs7.detached
/Name (PDF Insecurity Team)
/Location (Bochum)
/Reason (PDF Security Testing)
/M (D:20191025184447+02'00')
/Contents <...>
/ByteRange [0 1529 20475 35042]
>>
endobj
9 0 obj
<<
/Length 55
/Type /XObject
/Subtype /Form
/Resources <<
/XObject <<
/Im1 12 0 R
>>
>>
/FormType 1
```

```
/BBox [0.0 0.0 300.0 80.0]
>>
stream
0.25 0 0 0.25 0 0 cm
q
1079 0 0 150 0 0 cm
/Im1 Do
Q
Q

endstream
endobj
10 0 obj
<<
/Helv 13 0 R
/ZaDb 14 0 R
>>
endobj
11 0 obj
<<
/Type /SigRef
/DigestMethod /SHA1
/TransformParams 15 0 R
/TransformMethod /FieldMDP
/Data 1 0 R
>>
endobj
12 0 obj
<<
/Length 33239
/Type /XObject
/Subtype /Image
/Filter /FlateDecode
/BitsPerComponent 8
/Width 1079
/Height 150
/ColorSpace /DeviceRGB
/DecodeParms 16 0 R
/SMask 17 0 R
>>
stream
...
endstream
endobj
13 0 obj
<<
/Type /Font
/Subtype /Type1
/BaseFont /Helvetica
/Encoding /WinAnsiEncoding
>>
endobj
14 0 obj
<<
/Type /Font
/Subtype /Type1
/BaseFont /ZapfDingbats
>>
endobj
15 0 obj
<<
/Action /All
```

```
/Type /TransformParams
/P 1
/V /1.2
>>
endobj
16 0 obj
<<
/BitsPerComponent 8
/Predictor 15
/Columns 1079
/Colors 3
>>
endobj
17 0 obj
<<
/Length 180
/Type /XObject
/Subtype /Image
/Filter /FlateDecode
/BitsPerComponent 8
/Width 1079
/Height 150
/ColorSpace /DeviceGray
>>
stream
...
endstream
endobj
xref
0 2
0000000000 65535 f
0000000729 00000 n
3 1
0000001076 00000 n
5 13
0000000854 00000 n
0000001042 00000 n
0000001245 00000 n
0000001304 00000 n
0000020534 00000 n
0000020754 00000 n
0000020802 00000 n
0000020911 00000 n
0000054359 00000 n
0000054457 00000 n
0000054531 00000 n
0000054602 00000 n
0000054682 00000 n
trailer
<<
/Root 1 0 R
/Size 18
/ID [<6BE8661B8D4DCBD657D02C93257F569F> <6BE8661B8D4DCBD657D02C93257F569F>]
/Prev 553
>>
startxref
55036
%%EOF
```

Listing A.2: Beispieldokument mit Approval Signatur und FieldMDP *All* Parameter (Zeile 126, 182 und 228 gekürzt).

A.3 Grafische Oberfläche PDF Tester

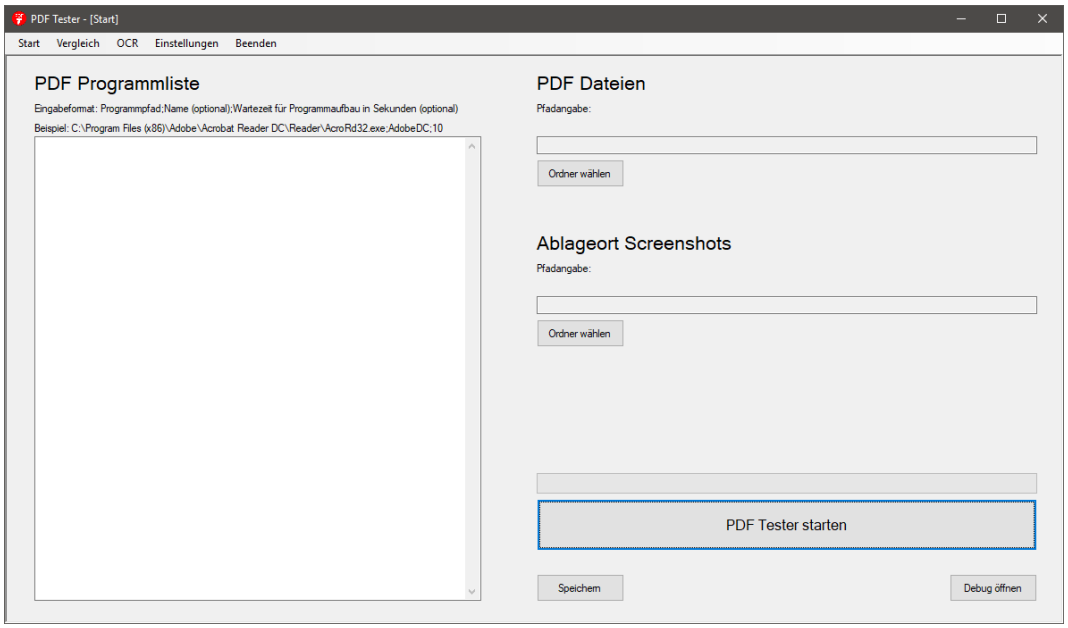

Abbildung A.3: Startseite *PDF Tester*

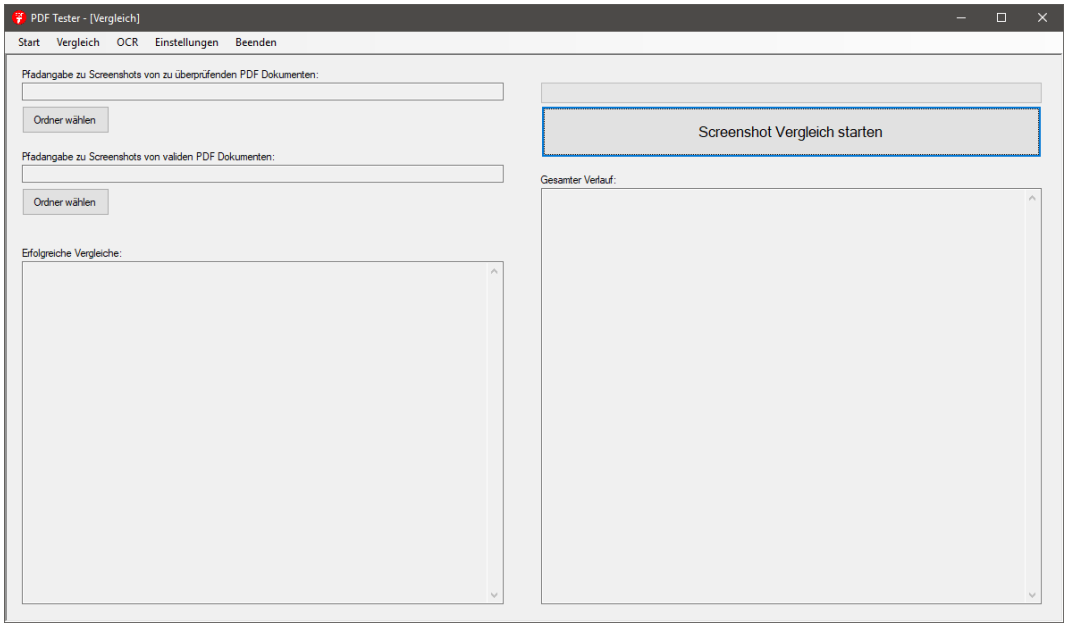

Abbildung A.4: Screenshot-Vergleich *PDF Tester*

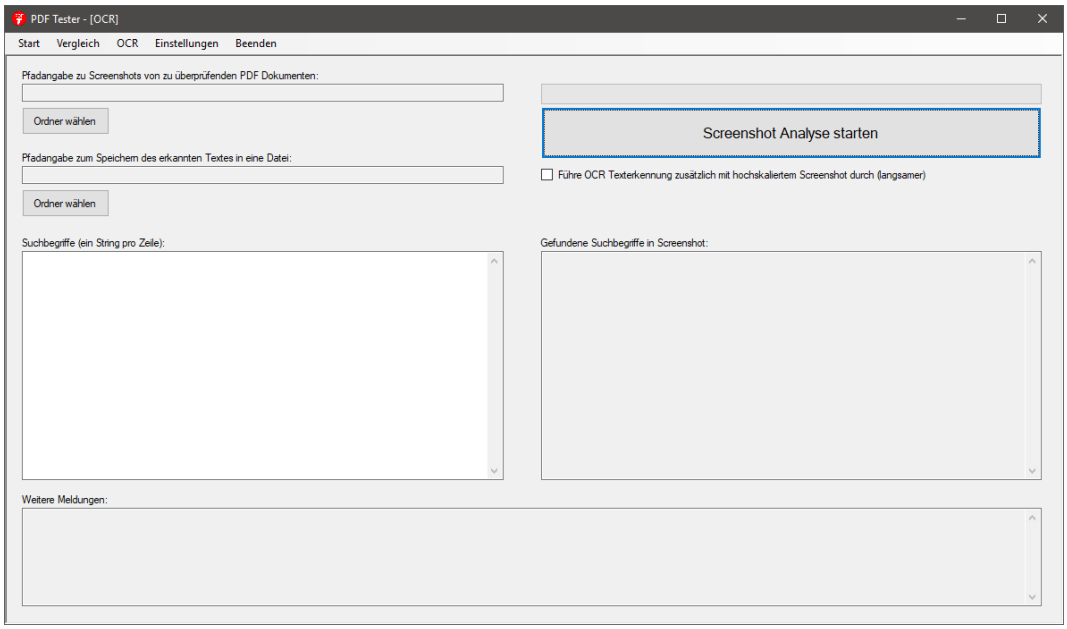

Abbildung A.5: Texterkennung *PDF Tester*

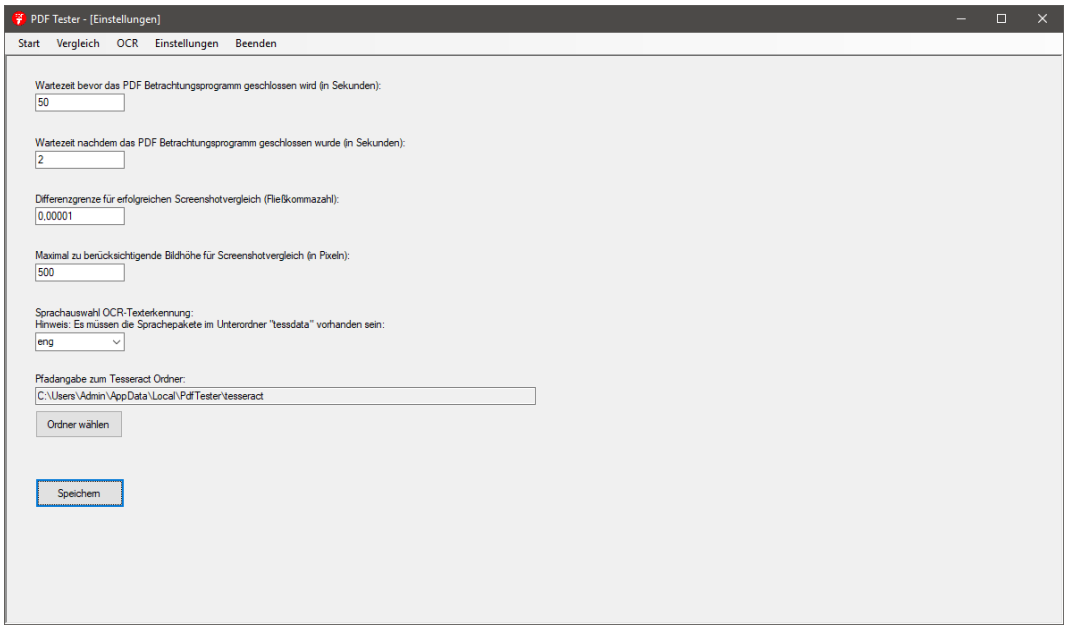

Abbildung A.6: Einstellungen *PDF Tester*

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

CBC Cipher Block Chaining.

DER Distinguished Encoding Rules.

DSA Digital Signature Algorithm.

EU Europäischen Union.

ID Identifikation.

ISA Incremental Saving Attack.

MDI Multiple Document Interface.

MDP Modification Detection and Prevention.

OCR Optical Character Recognition.

PDF Portable Document Format.

PIN Persönliche Identifikationsnummer.

PKCS Public-Key Cryptography Standards.

PKI Public Key Infrastructure.

RFC Request for Comments.

RSA Rivest–Shamir–Adleman.

SWA Signature Wrapping Attack.

UI User Interface.

USF Universal Signature Forgery.

VDG Vertrauensdienstegesetz.

Literaturverzeichnis

[1] Adobe Inc.: *Adobe Fast Facts*. https://www.adobe.com/content/dam/cc/en/fast-facts/pdfs/fast-facts.pdf, abgerufen am 1. Oktober 2019.

[2] Adobe Inc.: *Create and distribute PDF forms: Enable Reader users to save form data*. https://helpx.adobe.com/acrobat/using/creating-distributing-pdf-forms.html#enable_reader_users_to_save_form_data, abgerufen am 23. Oktober 2019.

[3] Adobe Systems Incorporated: *Adobe Portable Document Format Version 1.3*. Addison-Wesley, 1999.

[4] Adobe Systems Incorporated: *PDF 32000-1:2008*. In: *Document management - Portable document format - Part 1: PDF 1.7*, 2008.

[5] Adobe Systems Software Ireland Limited: *Ende der Unterstützung für Adobe Acrobat XI und Reader XI*. https://helpx.adobe.com/de/acrobat/kb/end-of-support-acrobat-xi-reader-xi.html, abgerufen am 15. Oktober 2019.

[6] Adobe Systems Software Ireland Limited: *Produkte und Zeiträume für technischen Support*. https://helpx.adobe.com/de/support/programs/eol-matrix.html, abgerufen am 15. Oktober 2019.

[7] Adobe Systems Software Ireland Limited: *Zertifikatbasierte Signaturen*. https://helpx.adobe.com/de/acrobat/using/certificate-based-signatures.html, abgerufen am 26. Oktober 2019.

[8] Ben Rogers (Adobe Systems Incorporated): *Digital Signatures in a PDF*. https://www.adobe.com/devnet-docs/etk_deprecated/tools/DigSig/Acrobat_DigitalSignatures_in_PDF.pdf, abgerufen am 1. Oktober 2019.

[9] Ben Rogers (Adobe Systems Incorporated): *Digital Signatures Workflow Guide*. https://www.adobe.com/devnet-docs/etk_deprecated/tools/DigSig/Acrobat_DigSig_WorkflowGuide.pdf, abgerufen am 27. Oktober 2019.

[10] Bundesrepublik Deutschland: *Vertrauensdienstegesetz (VDG)*. https://www.gesetze-im-internet.de/vdg/BJNR274510017.html, abgerufen am 1. Oktober 2019.

[11] Christof Paar und Jan Pelzl: *Digitale Signaturen*. In: *Kryptografie verständlich*. Springer Vieweg, 2016.

[12] Dan-Sabin Popescu: *Hiding Malicious Content in PDF Documents*. Journal of Mobile, Embedded and Distributed Systems - JMEDS, 2011. http://www.jmeds.eu/index.php/jmeds/article/view/Hiding-Malicious-Content-in-PDF-Documents/pdf.

[13] Das Europäische Parlament und der Rat der Europäischen Union: *Verordnung (EU) Nr. 910/2014 des Europäischen Parlaments und des Rates*. https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32014R0910, abgerufen am 1. Oktober 2019.

[14] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley und Tim Polk: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280, RFC Editor, Mai 2008. https://tools.ietf.org/html/rfc5280.

[15] Freistaat Sachsen: Landesamt für Steuern und Finanzen: *Qualifizierte Elektronische Signatur*. http://www.finanzamt.sachsen.de/eSignatur.html, abgerufen am 1. Oktober 2019.

[16] George Williams (FontForge Project): *FontForge Open Source Font Editor*. https://fontforge.github.io/en-US/, abgerufen am 30. Oktober 2019.

[17] Google Ireland Limited: *Tesseract OCR*. https://opensource.google/projects/tesseract, abgerufen am 17. Oktober 2019.

[18] Ian Markwood, Dakun Shen, Yao Liu und Zhuo Lu: *PDF Mirage: Content Masking Attack Against Information-Based Online Services*. 26th USENIX Security Symposium (USENIX Security 17), 2017. https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-markwood.pdf.

[19] Jens Müller, Fabian Ising, Vladislav Mladenov, Christian Mainka, Sebastian Schinzel und Jörg Schwenk: *Practical Decryption exFiltration: Breaking PDF Encryption*. 26th ACM Conference on Computer and Communications Security (CCS), 2019. https://pdf-insecurity.org/download/paper-pdf_encryption-ccs2019.pdf.

[20] Jörg Schwenk: *Kryptographie und das Internet*. In: *Sicherheit und Kryptographie im Internet*. Springer Vieweg, 2014.

[21] Karsten Meyer zu Selhausen: *Security of PDF Signatures*. Ruhr-Univeristät Bochum, 2018. https://pdf-insecurity.org/download/DIGITALVERSION_KMeyerZuSelhausen_SecurityOfPDFSignatures_2018-11-25.pdf.

[22] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini und Yarik Markov: *The first collision for full SHA-1*. CRYPTO 2017: 37th Annual International Cryptology Conference, 2017. https://www.researchgate.net/publication/318750485_The_First_Collision_for_Full_SHA-1.

[23] Microsoft Corporation: *BackgroundWorker Class*. https://docs.microsoft.com/de-de/dotnet/api/system.componentmodel.backgroundworker?view=netframework-4.7, abgerufen am 11. Oktober 2019.

[24] Microsoft Corporation: *MDI-Anwendungen (Multiple Document Interface)*. https://docs.microsoft.com/de-de/dotnet/framework/winforms/advanced/multiple-document-interface-mdi-applications, abgerufen am 10. Oktober 2019.

[25] Microsoft Corporation: *System.Windows.Forms Namespace*. https://docs.microsoft.com/de-de/dotnet/api/system.windows.forms?view=netframework-4.7, abgerufen am 10. Oktober 2019.

[26] Philip Doxakis: *How to use Tesseract OCR 4.0 with C#*. https://github.com/doxakis/How-to-use-tesseract-ocr-4.0-with-csharp/blob/master/Demo/Program.cs, abgerufen am 21. Oktober 2019.

[27] Ray Smith et al.: *Tesseract Open Source OCR Engine (main repository)*. https://github.com/tesseract-ocr/tesseract, abgerufen am 17. Oktober 2019.

[28] Rosetta Code: *Percentage difference between images*. https://rosettacode.org/wiki/Percentage_difference_between_images, abgerufen am 14. Oktober 2019.

[29] Russell Housley: *Cryptographic Message Syntax (CMS)*. RFC 3852, RFC Editor, Juli 2004. https://tools.ietf.org/html/rfc3852.

[30] StatCounter: *Desktop Operating System Market Share Worldwide - October 2019*. https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201909-201910, abgerufen am 1. November 2019.

[31] The Apache Software Foundation: *Apache PDFBox® - A Java PDF Library*. https://pdfbox.apache.org/, abgerufen am 25. Oktober 2019.

[32] Tim Bienz und Richard Cohn. In: *Portable Document Format Reference Manual*. Addison-Wesley, 1993.

[33] Tomáš Stefan: *Digital Signature Verification in PDF*. Czech Technical University in Prague, 2017. https://dspace.cvut.cz/bitstream/handle/10467/76810/F8-BP-2018-Stefan-Tomas-thesis.pdf?sequence=-1.

[34] Universitätsbibliothek Mannheim: *Tesseract at UB Mannheim*. https://github.com/UB-Mannheim/tesseract/wiki, abgerufen am 21. Oktober 2019.

[35] Vladislav Mladenov, Christian Mainka, Karsten Meyer zu Selhausen, Martin Grothe und Jörg Schwenk: *1 Trillion Dollar Refund - How To Spoof PDF Signatures*. 26th ACM Conference on Computer and Communications Security (CCS), 2019. https://pdf-insecurity.org/download/paper-pdf-signatures-ccs2019.pdf.