TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

TUT Centre for Digital Forensics and Cyber Security

ITC70LT

Rando Kulla 143650IVCM

MIGRATING PDF SIGNING TO NEW KSI FORMAT

Master thesis

Jaan Priisalu

MSc

Junior Researcher, TUT

Ahto Truu

MSc

Software Integration Architect, Guardtime AS

Tallinn 2016

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Rando Kulla

Abstract

Guardtime has developed Keyless Signature Infrastructure (KSI), which ensures data integrity and proves time of existence. This technology can also be used to sign and verify PDF files. Recently, Guardtime has introduced a new format for KSI signatures, which meets the requirements of KSI better. Currently only the old format can be used to sign and verify PDF files. Migration to the new format would make it possible to use the improved version of KSI signatures with PDF files.

The aim of this thesis is to provide solutions for migrating PDF signing to the new KSI format. An overview of relevant standards is given and how Guardtime and users of their services are affected by each provided solution. It is described for each solution what user needs to do and what Guardtime needs to develop to conduct the migration. The output of this thesis serves as an input to Guardtime for making decisions on the migration process.

This thesis is written in English and is 52 pages long, including 5 chapters, 10 figures and 6 tables.

Annotatsioon

PDF’i allkirjastamise üleminek uuele KSI vormingule

Keyless Signature Infrastructure (KSI) on teenus, mida pakub tehnoloogiaettevõte Guardtime. KSI võimaldab luua võtmevabu allkirju andmete terviklikkuse tõestamiseks. KSI võtmevabu allkirju võib kasutada ka PDF dokumentide allkirjastamiseks. Selleks pakub Guardtime Adobe pistikprogrammi ning programmeerimiskeeles Java kasutatavat rakendusliidest. Pistikprogrammiga saab võtmevabu allkirju verifitseerida ning rakendusliides võimaldab võtmevabu allkirju luua ja verifitseerida.

Guardtime on kasutusele võtnud uue võtmevaba allkirja vormingu. Hetkel on võimalik PDF dokumente allkirjastada ainult vana vormingut kasutades. Lõputöö eesmärgiks on kirjeldada probleeme ja võimalikke lahendusi, mis on vajalikud selleks, et kasutada uut vormingut PDF dokumentide allkirjastamiseks. Lahendusi välja pakkudes tuleb silmas pidada seda, et uue vormingu kasutuselevõtt ei tohi mõjutada vana vorminguga allkirjade verifitseerimist.

Esmalt antakse ülevaade sellest, kuidas KSI töötab. Seejärel käsitletakse allkirja salvestamise võimalust PDF failides, kasutades selleks PDF vormingu standardit ISO 32000-1. Lisaks kirjeldatakse tegevusi, mida peab kooskõlastama Adobe’ga, et kasutusele võtta uus võtmevaba allkirja vorming ja registreerida uus pistikprogramm. Lõputöö pakub analüüsidel põhinevaid lahendusi, et võtta kasutusele võtmevaba allkirja uus vorming PDF dokumentides ja kuidas see mõjutab Guardtime’i ja nende kliente.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 52 leheküljel, 5 peatükki, 10 joonist, 6 tabelit.

Table of abbreviations and terms

API	Application Programming Interface
ETSI	European Telecommunications Standards Institute
ISO	International Organization for Standardization
KSI	Keyless Signature Infrastructure
PDF	Portable Document Format
PKI	Public Key Infrastructure
SDK	Software Development Kit
TLV	Type-Length-Value
TS	Technical Specification

Table of contents

List of figures

List of tables

Introduction

Guardtime is a technology company focusing on data-centric security. One of the services they provide is Keyless Signature Infrastructure (KSI). KSI is a hash-linking-based time-stamping service. A time-stamp token used with KSI is called a keyless signature, which can be used to verify data integrity and prove time of existence.

There are two signature formats developed for a keyless signature. The old format is based on the RFC 3161 specification. As the KSI service has evolved, a new signature format specification has been developed to better support the new features of the service.

In addition to KSI service, Guardtime has developed a PDF toolkit and an Adobe plug-in to support the usage of a keyless signature with PDF files. The PDF toolkit can be used to create and verify keyless signatures, the Adobe plug-in is meant only for verification. Currently only the old keyless signature format can be used with PDF files.

The goal of this thesis is to provide solutions for migrating to the new format of keyless signatures for PDF files. Each solution analyses the migration plan, backwards compatibility, user impact and possible security vulnerabilities. As a result, the thesis gives a recommendation on which solution should be used in which circumstance.

1. KSI

This chapter gives an overview of Keyless Signature Infrastructure (KSI) technology to understand its main principles. The signature formats and communication protocols are described as well.

1.1. Description of KSI technology

KSI uses a keyless signature to ensure data integrity and prove time of existence. Guardtime has defined two signature formats for keyless signatures. These two signature formats are used by two different KSI services, which use their own infrastructure. This means there are two different protocols for creating and verifying keyless signatures. These formats and protocols are described in sections 1.2 and 1.3.

KSI is a hash-linking-based time-stamping service. The implementation of KSI does not rely on the secrecy of a private key.

Data signature in KSI comes from the usage of a Merkle tree, what is used to verify the integrity of data [1, pp. 125-127]. Hash values provided for building the Merkle tree are received from users who want to time-stamp data. After the tree is built from received requests, the hash values are published using the Merkle tree (Figure 1).

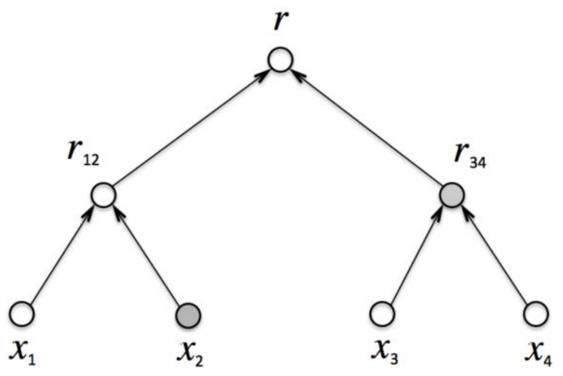

Figure 1. Merkle hash tree. [2, p. 4]

The root hash value of received requests is added to a special kind of hash tree which is called a hash calendar (Figure 2). The hash calendar is updated every second by publishing new hash value [2, p. 5].

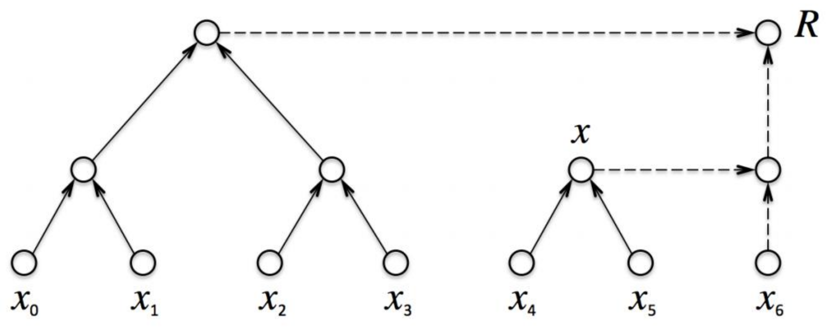

Figure 2. Computation of publication from hash calendar. [2, p. 5]

The status of the hash calendar (root hash of the hash calendar) is fixed by publishing it once a month electronically in a publication file and in newspapers. Guardtime signs the electronic publication files using public key infrastructure (PKI) to provide a reliable source of information to clients until the next newspaper publication becomes available. Publishing the hash calendar's root hash makes it possible to widely witness its value, making it impossible to be manipulated by anyone [3, p. 30]. The publication can be used to verify a data signature. This leads to one important point of KSI - there is no need to trust a third party to verify the signature.

1.1.1. How KSI works

Aggregation and extension are two services of KSI. Aggregation is the most important step to get a datum time-stamped. During aggregation the user sends the hash of the datum to the aggregation service, which links together all the received hash values by building a hash tree. After the aggregation is done, the user receives a time-stamp token called a keyless signature. This keyless signature connects the user's hash value with the root hash value of the aggregated hash tree. The root hash value of all the aggregations is published in a hash calendar, which is maintained by Guardtime

The extension service is used to connect a keyless signature to a publication [4]. Figure 3 describes the sequence of creating and verifying a keyless signature.

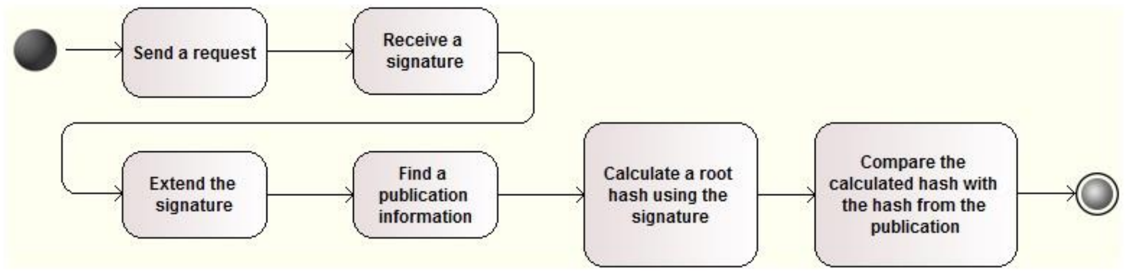
Send a request
Receive a signature
Extend the signature
Find a publication information
Calculate a root hash using the signature
Compare the calculated hash with the hash from the publication

Figure 3. Simplified description of the signing and verification process from the user's perspective.

1.2. General overview of the old format

The specification of the old format is not publicly available, so it cannot be described in full detail.

There is a dedicated infrastructure providing the KSI service with the old signature format and the old protocol. The old KSI service provides aggregation and extension according to ISO 18014-3 [5] standard. RFC 3161 [6] based communication protocol is used to provide access to the KSI service. All keyless signatures created by the old KSI service are using the old signature format, which is based on RFC 3161.

Signing process is initiated by the user who sends a time-stamping request to the aggregation service. KSI service returns a keyless signature to the user when the aggregation is completed.

As described in Section 1.1.1 the extension service can be used to verify a keyless signature. The extension service for signatures in the old format is provided by the old KSI service.

1.3. New format

As RFC 3161 based communication protocol and format needed many modifications to meet new KSI-specific requirements, it was decided to introduce a new format and a new protocol. The new keyless signature format and protocol are supported by the new KSI service. The new protocol used for aggregation and extension is not backwards compatible with the old protocol. All the requests and responses have a different structure from the RFC 3161. Type-Length-Value (TLV) structured Protocol Data Units (PDU) are used for the communication [7]. There still remains a possibility to convert signatures from the old format to the new format. The support for format compatibility is described in Section 1.3.1.

1.3.1. Signature format

New signatures are encoded using TLV encoding [7]. A uni-signature is a data unit providing information about a keyless signature. Values which can be in a uni-signature are aggregation hash chain components, calendar hash chain components, reference to the media containing publication and RFC 3161 compatibility record [8].

As described before, one of the components in a uni-signature can be a record for RFC 3161 compatible signatures. In the old format the signed value was not only the hash of a datum but also the hash value of different structures. Figure 4 describes a datum signed with the old KSI service. The new signature format has a helper data structure for old signatures to provide a possibility to convert a signature in the old format to the new format. However, the conversion from the new format to the old format is not possible. In general, the new signature format holds more information than the old format, but not all the information needed for signatures in the old format.

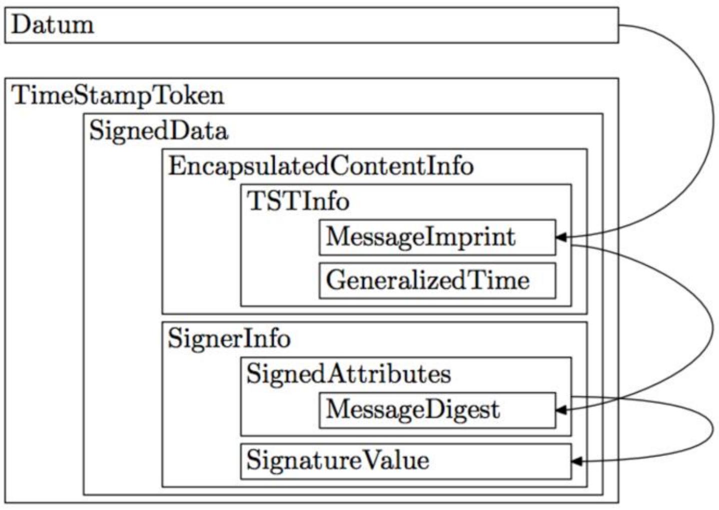

Figure 4. Computation of signature value with RFC 3161. [9, p. 23]

2. PDF signature

This chapter describes how Portable Document Format (PDF) supports the storage of digital signatures and how this is used by Guardtime to store a keyless signature inside PDF files. Standards which define how signatures must be stored inside a PDF file are also described.

Guardtime provides a PDF toolkit, which is described in Section 2.2.1, for creating and verifying PDF files with keyless signatures. Guardtime has also developed an Adobe plug-in for verifying a keyless signature using Adobe Reader or Adobe Acrobat. Requirements for developing an Adobe plug-in are described in Section 2.2.2.

2.1. Standards

Keyless signatures inside PDF file should be stored in a standardized way. This enables use of third party applications for adding support for the keyless signature format. Without following standards there is no possibility to use Adobe Reader/Acrobat for signature verification. Adobe Reader/Acrobat would not be able to recognize signature inside a PDF file and would not activate the plug-in.

2.1.1. ISO 32000-1

In general, ISO 32000-1 specifies the representation of Portable Document Format (PDF). Together with supporting software, PDF provides a possibility for digital signing [10, p. vii]. ISO 32000-1 defines two activities for digital signatures: adding a signature into a file and validating a signature [10, p. 466]. It also describes how a signature needs to be stored inside a PDF file (Figure 5).

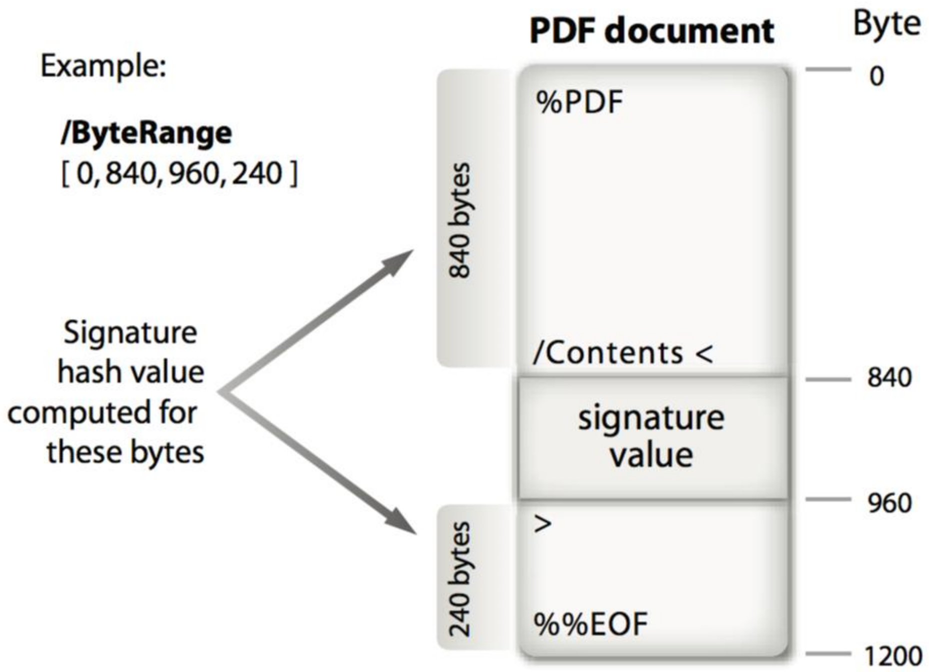

Figure 5. Structure of a signed PDF file. [11, p. 5]

Before signing a PDF file, it is needed to allocate space for the signature. The standard defines that signature must fit precisely into space allocated for signature. If the signature size cannot be predicted, it must be padded with zeroes to fill the allocated space [10, p. 476].

The standard defines a signature dictionary, which is an object for storing signature related information inside a PDF file [10, p. 466]. Figure 6 describes the signature dictionary. The value of the signature is stored in the Contents field in the signature dictionary as a hexadecimal string enclosed in angle brackets [10, p. 16]. In case of public-key signatures the value must be DER-encoded PKCS#1 or PKCS#7 binary data object [10, p. 468]. Currently a keyless signature is an RFC 3161 signature object. The ByteRange field of the signature dictionary indicates the range of bytes used for computing the hash for signing – showing what part of the file is signed [10, pp. 466, 468]. The Filter field defines the preferred signature handler for validating the signature [10, p. 467]. Guardtime has defined “GTTS.TimeStamp” as a signature type for keyless signatures, where “GTTS” is a developer specific prefix. The SubFilter field specifies the encoding of the signature value [10, p. 474]. Guardtime uses the SubFilter value defined by European Telecommunications Standards Institute (ETSI) which is “ETSI.RFC 3161”.

Also Type value is defined as “DocTimeStamp” according to ETSI. Section 2.1.2 shortly describes the ETSI standard.

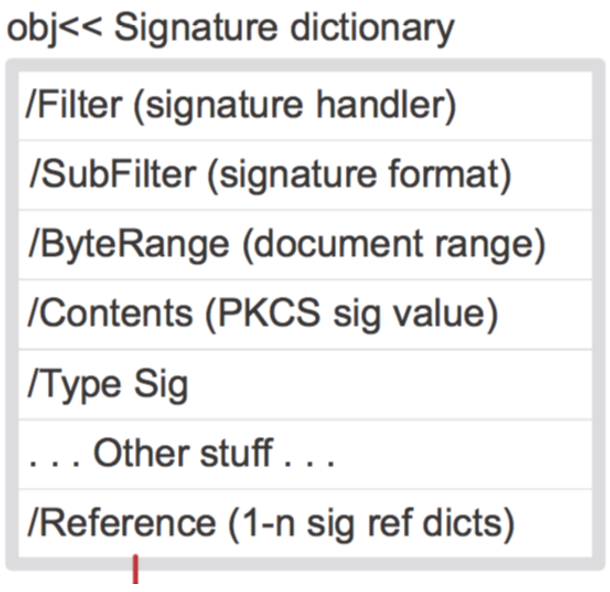
obj<< Signature dictionary
/Filter (signature handler)
/SubFilter (signature format)
/ByteRange (document range)
/Contents (PKCS sig value)
/Type Sig
. . . Other stuff . . .
/Reference (1-n sig ref dicts)

Figure 6. Structure of a signature dictionary inside a PDF file. [11, p. 2]

Several signatures can be added to a PDF file without corrupting previous signatures. It can be achieved due to the incremental update functionality of the Portable Document Format. A PDF file can be modified by adding changes to the end of the file without changing previously signed bytes [10, p. 44]. The second signature covers the updated part and the previous version of the file, including the previous signature. It means that if there is one signature inside a PDF file, it is possible to change the signature’s value. If a second signature is added, it is not possible to change the value of the first signature without breaking the second one. Only the most recent signature can be changed. There might be a need to change the signature’s value during signature conversion to continue supporting the verification of signatures in the old format (see Section 3.3.3 for more details).

2.1.2. ETSI TS 102 778 part 4

ETSI has specified requirements for Long Term Validation of PDF signatures in TS 102 778 part 4. One part of this paper specifies how to use time-stamps to provide possibility to verify PDF signatures long after signing [12, p. 5].

Guardtime follows this specification for storing keyless signatures inside PDF files. A keyless signature is stored according to RFC 3161 format as it is described in ETSI TS

102 778 part 4. Guardtime uses the SubFilter value defined by ETSI which is “ETSI.RFC 3161”. Also the Type value is defined as “DocTimeStamp” according to ETSI [12, p. 15].

2.2. Software

Guardtime provides two software components for the user to use keyless signatures with PDF files: a PDF toolkit and an Adobe plug-in. The PDF toolkit is meant for development purposes – it allows to integrate PDF file signing and verification into third-party software. The PDF toolkit allows to create, sign and verify PDF files. The Adobe plug-in is meant for verification using Adobe Reader or Adobe Acrobat.

2.2.1. PDF toolkit

The PDF toolkit is a product developed in the Java programming language, providing a possibility to handle keyless signatures with PDF files. PDF toolkit allows creating PDF files and then signing them with keyless signatures. Existing keyless signatures can also be verified with PDF toolkit.

Source code of the PDF toolkit is provided to customers. This gives a possibility to conduct code review for the source code before integrating the functionality of keyless signatures.

2.2.2. Adobe plug-in

Adobe plug-in is developed to be used with Adobe Acrobat and Adobe Reader. It can be used to verify keyless signatures inside PDF files. Adobe Reader and Adobe Acrobat are supported on OS X and Windows, making the distribution of the verification tool easier [13] [14].

According ISO 32000-1 the Adobe plug-in developed by Guardtime is a signature handler. Adobe has a registry of developers and registered formats to avoid name conflicts. For registration Adobe provides an online form to apply for a format registration [10, pp. 466, 673]. Guardtime has registered “GTTS.TimeStamp” as a signature handler type for the keyless signatures. There is a license fee for each signature type, which needs to be paid annually.

Before being able to develop a plug-in which would be accepted by Adobe Reader, there is a need to get registered as a developer with Adobe. The registration process may take a few weeks [15]. During the registration process a developer will create a public and private key pair for signing the plug-in. During the registration process Adobe will sign a certificate, which includes the developer’s public key. After that, the developer is able to sign the plug-in. The signed version of the plug-in will be accepted by Adobe Reader or Adobe Acrobat [16]. The signed version of the plug-in can be distributed to users directly by the developer.

Before signing the plug-in, there is a need to define, which signature type can be handled by this plug-in. It is needed for connecting signed files with signature handler. Adobe Reader/Acrobat examines if there are any plug-ins, which can process the found signature type when a PDF file with a signature is opened. If a corresponding plug-in is found, it will be activated [17].

It is also possible to register the developed plug-in with Adobe. During the registration process, there is a need to define which signature type can be handled by the plug-in. The plug-in registration helps in a situation where the user wants to verify a PDF file with the corresponding signature type, but do not have the plug-in installed for verification. If the plug-in is registered, Adobe Reader and Adobe Acrobat are able to redirect the user to the link where the plug-in can be downloaded [18].

3. New format adoption

This chapter is based on the analysis made in Chapter 1 and Chapter 2. The aim of this chapter is to provide options for migrating to new keyless signature format.

Firstly, there is an overview of the aspects, which need to be kept in mind during the new signature format adoption. After that, a description of possible migration options is given. The last two sections will describe what Guardtime needs to do for each migration method and how end users would be affected by the migration process.

3.1. Overview

The next section points out what are the requirements for the migration and what makes the migration difficult.

3.1.1. Facts to be considered

The old signature format cannot be used with the new KSI service. The new KSI service does not support RFC 3161 based communication protocols and cannot be made backwards compatible. Creating signatures in the old format with the new KSI service is not possible, but there is a way to add support for the verification of signatures in the old format. Section 3.3.4 describes how this can be achieved.

The new protocol may get updates or modifications during its lifetime. Changes made to the new protocol can influence the support of signatures in the old format. This must be considered when describing adoption options. Every provided solution should describe which part of the solution might be affected by future changes made to the new protocol or the new format.

3.1.2. Requirements

One of the requirements is to maintain the possibility of verification of old signatures after the new signature format is deployed. It would not be acceptable if old signatures could not be verified anymore. The old KSI service has been used for several years and a large amount of signatures would be affected. Problems may rise if the old KSI service is not provided anymore and signatures in the old format cannot be verified with the new KSI service. This aspect must be kept in mind as eventually the old KSI service will have to be retired. Without the old KSI service it is possible to shut down the old infrastructure which eliminates the need to maintain two infrastructures at the same time.

Every provided solution needs to point out the kind of resources needed on the user side and what Guardtime needs to do to conduct the migration. All the software components which need to be developed or updated must be described.

3.2. Solutions

This section provides possible solutions for starting to use the new format of keyless signatures with PDF files. A general overview of possible solutions is given, followed by detailed description of all the options.

Although it is possible to develop a web service where the user can upload a PDF file for the verification, it is not a suitable option, because the user would need to disclose the file's content. This is not acceptable from the confidentiality point of view.

To minimize the impact on customers, it would be beneficial to continue using the PDF toolkit and the Adobe plug-in. All the provided options continue using those software components.

There are two important aspects to be considered: registrating the signature type with Adobe and keeping the old format in use. Supporting only the new format means there will be no opportunity to either sign or extend with the old KSI service. Without the old KSI service there is no way to use the old versions of the Adobe plug-in and the PDF toolkit. Another option would be to continue supporting the old format and add support for the new signature format. There is an option to start using the new signature type by registering it with Adobe (see Section 2.2.2 why registration is needed). The new signature type registration results in a new contract with Adobe. As a result, two signature types would be supported by Guardtime. It is possible to use the new signature format without the registration of the new signature type. The new signature format can be used with the same signature type registered with Adobe as it is used with the old signature format.

Taking all the previous aspects into account, Table 1 sums up the possible options for the new format migration.

Table 1. Potential solutions.

Supported format / New signature type	New and old format	New format
Registered with Adobe	"Signature type registration method"	Not suitable
Not registered with Adobe	"One type, two formats method"	"Signature conversion method"

The option where the new signature type is registered with Adobe and only the new format is supported does not support the verification of signatures with the old format. Due to the new signature type there is no possibility to convert old signatures to the new format because signature type is signed and cannot be changed. This solution is not acceptable from the user perspective and will not be discussed further.

3.2.1. Signature type registration method

With the "Signature type registration method, the new signature type is registered with Adobe and the old format is kept in use in addition to the new format.

After the new signature type is registered with Adobe, a new plug-in must be provided for the verification of signatures in the new format. The old version of Adobe plug-in cannot be used to verify signatures in the new format. Signatures in the new format would use a different signature type and they have different encoding from the old format. It would be desirable to have one Adobe plug-in which could verify signatures in both signature formats, but it is not possible as there must be one plug-in per signature type (see Section 2.2.2 for more details). Signatures in the old format cannot be verified with the new plug-in as old signatures' type cannot be changed because signature type is a signed value inside the signature dictionary.

Due to the fact that the new signature type is taken into use, there is no need to convert signatures in the old format to the new format. The usage of the new signature type makes it impossible to convert signatures in the old format into the new format if it might be needed in the future. The conversion would need to change the signature type which is part of signed datum. The change of signature type would invalidate the signature.

It is needed to develop a new Adobe plug-in and update the PDF toolkit. Sections 3.3.1 and 3.3.2 describe what needs to be covered during the development of the PDF toolkit and the Adobe plug-in.

There is no possibility to sign PDF files with the old signature format if Guardtime stops providing the old KSI service. There still remains an opportunity to verify signatures in the old format using the new KSI service. It is possible by developing an external proxy (see Section 3.3.4 for more details) which converts an old protocol verification request to the new format and responds using the old format. Signatures in the old format can be verified using the PDF toolkit and the Adobe plug-in if they are configured to use the external proxy for the verification. The other option would be to embed the proxy's functionality inside the PDF toolkit and the Adobe plug-in. The embedded proxy method assumes that the user updates the PDF toolkit and the Adobe plug-in. It must be taken into account that the new KSI service is used by the proxy and changes made to the new protocol can affect the proxy's functionality. Major changes in the new protocol may result in a situation in which signatures in the old format cannot be verified with the new KSI service. The proxy should be used only if the support of the old KSI service is shut down.

Figure 7 gives an overview of activities needed to migrate the new signature format using this solution. The order of actions which influence the support of signature formats are pointed out.

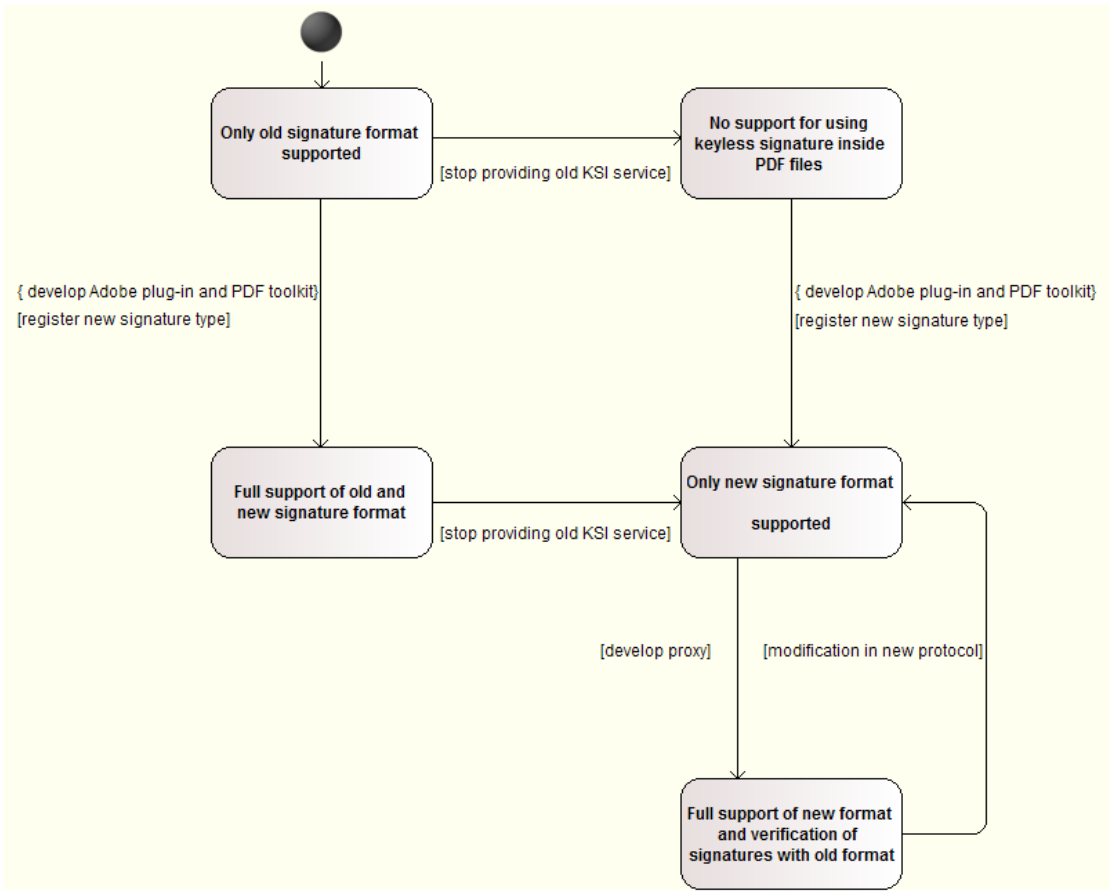
Only old signature format supported
[stop providing old KSI service]
No support for using keyless signature inside PDF files
{ develop Adobe plug-in and PDF toolkit}
[register new signature type]
{ develop Adobe plug-in and PDF toolkit}
[register new signature type]
Full support of old and new signature format
[stop providing old KSI service]
Only new signature format supported
[develop proxy]
[modification in new protocol]
Full support of new format and verification of signatures with old format

Figure 7. State diagram of "Signature type registration" method.

To sum up, a new Adobe plug-in supporting the new signature format should be developed. The old plug-in must be maintained as well. The PDF toolkit must be updated so it would support the new signature format. Before starting to use the new signature format the new signature type must be registered with Adobe. The registration of the new Adobe plug-in would increase user experience. All the registration processes with Adobe take time. The old KSI service must be provided as the old format remains in the use. If there is a need to shut down the old KSI service, then the proxy makes it possible to verify signatures in the old format. Read Section 3.4 for activities required from the user. Table 2 summarizes actions connected to this solution.

Table 2. Summary of “Signature type registration” method activities.

	Conversion	Old KSI service	Proxy	Software development	Adobe
Description	Cannot be done	Required	Required if old KSI service is not provided	New plug-in + update PDF toolkit + proxy (if required)	Register new signature type and new plug-in

3.2.2. One type, two formats method

The new signature format can be used in addition to the old format without the registration of the new signature type. This means that the new and the old signature formats are using the same signature type registered with Adobe.

The first step is to update the PDF toolkit and the Adobe plug-in so that the new signature format is supported. Sections 3.3.1 and 3.3.2 describe the update of those software components. As the old and the new formats are using the same signature type, there is no need to develop an additional Adobe plug-in. There is also no need to go through the registration process of the new plug-in with Adobe (as it would be with “Signature type registration method). The updated version of the PDF toolkit and the Adobe plug-in are able to handle both signature formats.

Using the same signature type keeps an opportunity to convert old signatures to the new format. This solution itself does not require the conversion.

With this solution, the old KSI service is required for signing and verifying signatures in the old format. When the old KSI service is retired, the Signature conversion method could be used as a fallback to handle the situation. Another option would be to develop the proxy (see Section 3.3.4) for verifying signatures in the old format, as it was with “Signature type registration method. With the external proxy solution, the user needs to configure the PDF toolkit and the Adobe plug-in to use the proxy for verification. When the embedded proxy solution is chosen, the user must update the PDF toolkit and the Adobe plug-in to be able to verify signatures in the old format.

To sum up, this solution does not need the registration of the new signature type with Adobe. The old Adobe plug-in must be updated in a way that signatures in the new format can be verified. The support for the new signature format must also be added to the PDF toolkit. If there is a need to shut down the old KSI service, then the proxy enables verification of signatures in the old format. Section 3.4 describes activities required from the user. Table 3 summarizes the actions required with this solution. Figure 8 shows how the order of actions influence the support of the signature formats.

Table 3. Summary of "One type, two formats" method activities.

	Conversion	Old KSI service	Proxy	Software development	Adobe
Description	Not required (possibility remains)	Required	Required if old KSI service is not provided	PDF toolkit + Adobe plug-in + proxy (if required)	No contact required

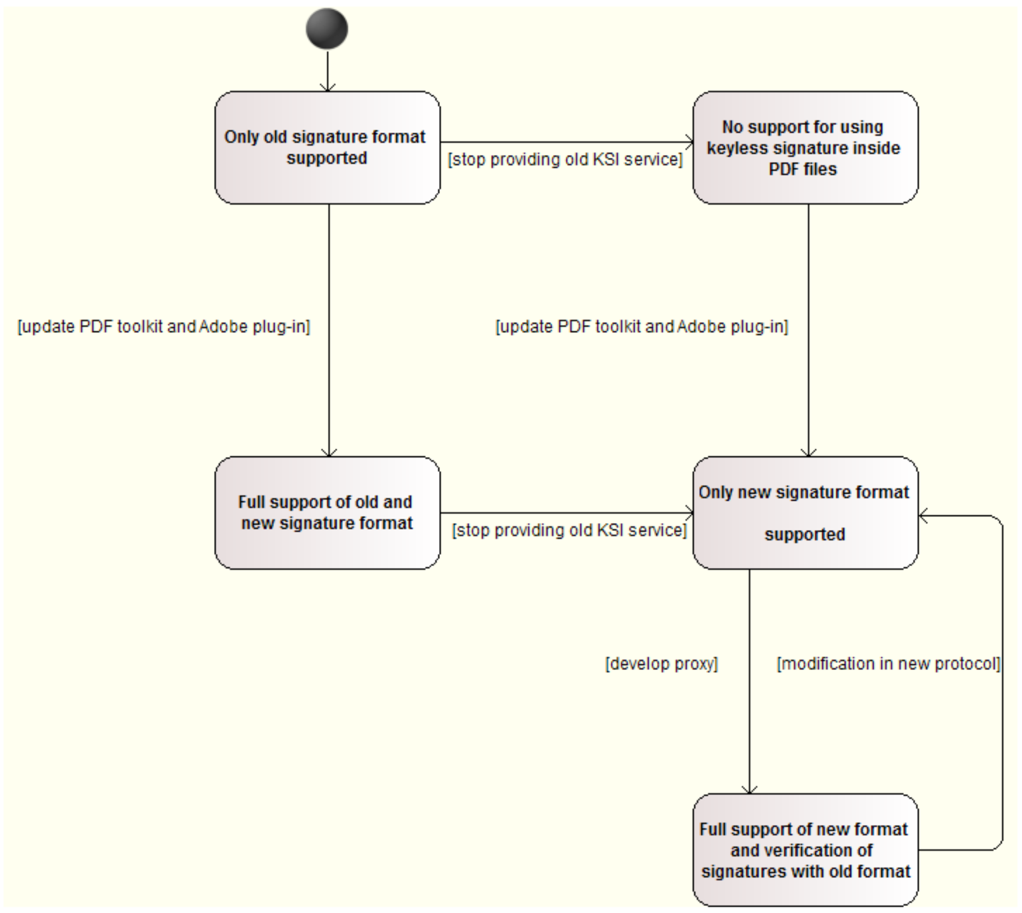
Only old signature format supported
[stop providing old KSI service]
No support for using keyless signature inside PDF files
[update PDF toolkit and Adobe plug-in]
[update PDF toolkit and Adobe plug-in]
Full support of old and new signature format
[stop providing old KSI service]
Only new signature format supported
[develop proxy]
[modification in new protocol]
Full support of new format and verification of signatures with old format

Figure 8. State diagram of "One type, two formats" method.

3.2.3. Signature conversion method

Rather than using the new signature format in addition to the old format, there is a possibility to use the new format instead of the old format. This means there is no need to register the new signature type with Adobe as it is with “Signature type registration method. It is also not required to continue supporting the old KSI service as it is with the “Signature type registration method and the “One type, two formats method.

The first step is to update the PDF toolkit and the Adobe plug-in, so the new signature format is supported. Sections 3.3.1 and 3.3.2 instruct how these software components must be updated. The new PDF toolkit will not support creating signatures in the old signature format, which is not possible because the old KSI service is not provided with this solution. This removes a necessity to maintain two infrastructures at the same time, while still being able to verify signatures in the old format. The support of the old signature format depends on which signature conversion method is chosen.

As new signatures are using the same signature type as old signatures, it is possible to verify both signature formats with one Adobe plug-in. Signatures in the old format must be converted to the new format to verify them with the new KSI service. There are two options for converting old signatures to the new format. One option is to provide the converter, which will convert signatures to the new format by replacing the old signature inside a PDF file with a converted signature. The other option is to convert signatures during the verification process inside the PDF toolkit and the Adobe plug-in without modifying the PDF file. Section 3.3.3 provides more information about the signature conversion methods.

The new PDF toolkit will support the creation and the verification of signatures in the new signature format. Depending on which conversion method is chosen, it might be that the PDF toolkit is able to verify signatures in the old format.

To sum up, this solution does not need the registration of the new signature type with Adobe. The old Adobe plug-in must be updated in a way that signatures in the new format can be verified. The updated version of the PDF toolkit does not provide an option to create signatures in the old format. Supporting the verification of signatures in the old format depends on the chosen conversion method, this applies to the PDF toolkit and the Adobe plug-in. Section 3.4 describes activities required from the user. Table 4 summarizes the actions required with this solution. Figure 9 shows how the order of actions influences the support of the signature formats.

Table 4. Summary of "Signature conversion" method activities.

	Conversion	Old KSI service	Proxy	Software development	Adobe
Description	Required	Not required	Not required	PDF toolkit + Adobe plug-in + converter	No contact required

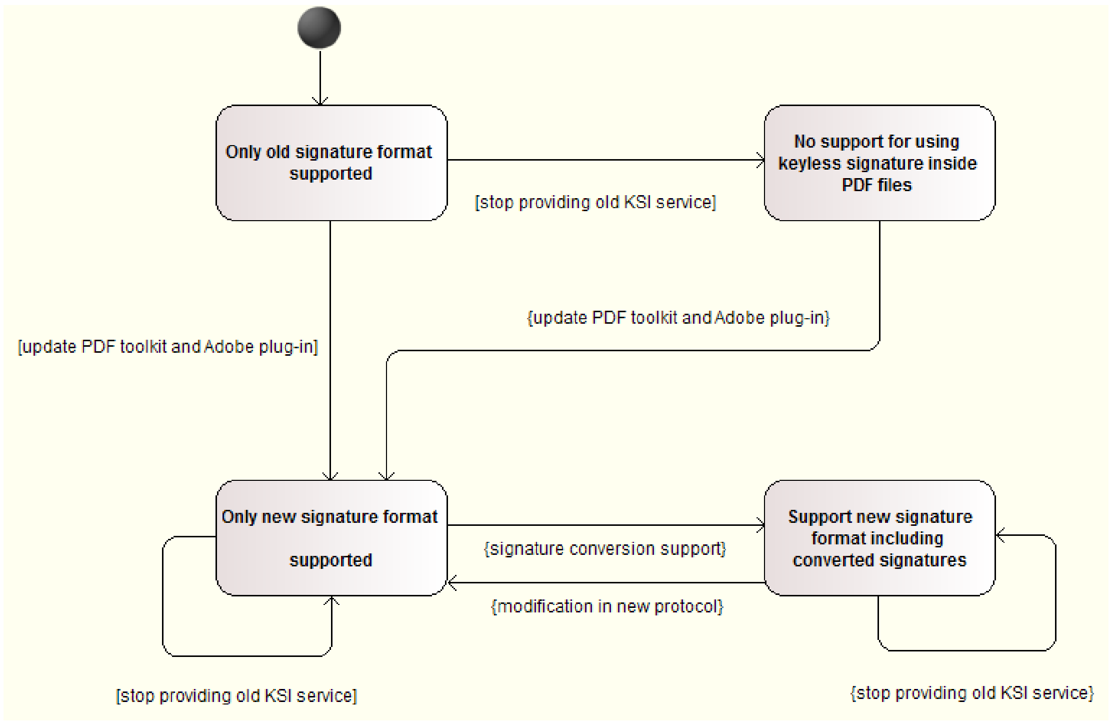
Only old signature format supported
No support for using keyless signature inside PDF files
[stop providing old KSI service]
{update PDF toolkit and Adobe plug-in}
[update PDF toolkit and Adobe plug-in]
Only new signature format supported
Support new signature format including converted signatures
{signature conversion support}
{modification in new protocol}
[stop providing old KSI service]
{stop providing old KSI service}

Figure 9. State diagram of "Signature conversion" method.

3.3. Development

The new format migration cannot be conducted without updating related software components. This chapter describes what needs to be developed to support previously provided solutions. Descriptions are separated by software modules.

In addition to the new signature type registration with Adobe, there is a need to define SubFilter value for the new keyless signature format. At the moment the value "ETSI.RFC 3161" is used. The old value cannot be used because the new signature format does not follow RFC 3161 structure. According to ISO 32000-1, there is no need to register the value of SubFilter with Adobe but the used value for SubFilter should start with developer-specific prefix to avoid conflicts in SubFilter values [10, p. 673]. According to the PDF specification, the value of SubFilter must describe the encoding of a signature [10, p. 467]. As first four characters are known, the second part of the name must be created. As SubFilter must specify encoding, the possible value for SubFilter can be “GTTS.ksig.tlv”. This name would refer to a keyless signature which is encoded with the TLV scheme.

When the old and the new signature formats use the same signature type, as it is with the One type, two formats method and the Signature conversion method, the PDF toolkit and the Adobe plug-in must be able to tell the difference between the old and the new signature format. In case of the One type, two formats method, only SubFilter is required to tell the difference between the two formats. Chosen signature conversion method (see Section 3.3.3) influences the verification process with Signature conversion method. If signatures in the old format are converted and stored inside PDF files, then software components parsing the signature cannot use the value of SubFilter to check which signature format is used. SubFilter is a signed value and cannot be changed. As a result, converted signatures are in the new format but SubFilter refers to the old format. In this case, the parser must try to decode the signature value assuming it uses one of the formats. If the result is not meaningful, the other format structure must be used. The value of SubFilter can be used to determine the format if conversion is done by the PDF toolkit or the Adobe plug-in and the signature inside a PDF file remains unchanged.

3.3.1. PDF toolkit

Guardtime needs to provide the updated version of the PDF toolkit to use the new signature format for signing PDF files. The new version of KSI Java SDK must be used to add support for the new signature format. At the moment the PDF toolkit is using the old version of KSI Java SDK. There are major changes in the SDK but the PDF toolkit can be updated without changing the user interface.

With “Signature type registration method and the “One type, two formats method, there is a need to continue supporting the old signature format. If both signature formats are supported, the user interface may get changed to provide a possibility to handle both signature formats. It must be made clear and easy for a user to distinguish which signature format is used.

The updated version of the PDF toolkit with the Signature conversion method would only support the new signature format. Depending on the chosen conversion method (see Section 3.3.3), there might be a need to support the verification of the old signature format. This is required if the chosen conversion method does not modify a PDF file by converting the signature inside it. It means that the PDF toolkit converts the signature only for the time of verification and leaves a PDF file unchanged. With the second conversion method where PDF files are modified, there is a need to provide the signature converter to the user. Signature conversion functionality can be part of the PDF toolkit, making it easier for the user to manage software. Section 3.4.1 describes how users are affected by the PDF toolkit update.

3.3.2. Adobe plug-in

The Adobe plug-in must be updated to verify signatures in the new format. The new version of KSI C API, which supports the new signature format and uses the new KSI service, can be used to do the update.

When the new signature type is registered with Adobe, as it is with the "Signature type registration method, a new plug-in should be provided to verify signatures in the new format. A separate plug-in must be developed because one plug-in can be used for verifying signatures with one specific signature type. The new plug-in must be registered with Adobe. It is needed for declaring that the new plug-in is the appropriate signature handler for the new signature type. Section 2.2.2 describes how the registration can be done.

With the One type, two formats method, the Adobe plug-in must be updated in a way that it can tell the difference between the old and the new signature format. The same applies if the Signature conversion method is chosen and the plug-in needs to convert signature before verification (see Section 3.3.3). With the Signature conversion method, the updated version of the plug-in has to support only the new signature format because signatures stored inside a PDF file are converted. Section 3.4.2 points out how users are affected by the plug-in update.

3.3.3. Signature conversion

Signature conversion is one part of a migration process with Signature conversion method. When using the new KSI service, signature conversion is required to verify signatures in the old format. There are two possible ways of handling signature conversion. One option is to develop a standalone converter, which replaces the old signature inside a PDF file with the converted one. Another option is to integrate the conversion functionality to every software component which needs to verify signatures in the old format using the new KSI service.

The purpose of the standalone signature converter is to read a keyless signature from an existing PDF file, convert the signature to the new format and replace the old signature with converted value. Signatures in the old format can be converted to the new format due to the record defined for RFC 3161 compatible signature in the new format (see Section 1.3.1).

The converter must ensure there are no values other than “0” following the converted signature in the signature field after having replaced the signature. It is possible that the converted signature is smaller than the old signature and some bytes from the old signature would follow the converted signature. Padding of “0” must be added before delimiter “>” [10, p. 476].

Depending on the time of signing, the size of a signature in the old format is ~3400 bytes, converted to a hexadecimal representation (which is stored inside a PDF file) is ~6800 bytes. The size of a converted signature (in the new signature format) is ~2800 bytes and value converted to a hexadecimal representation is ~5900 bytes. Fitting a converted signature into the Contents field is not problematic because converted signatures are smaller than signatures in the old format. In most cases, there are ~16 000 bytes allocated for a signature inside a PDF file during the signing process.

The drawback of the signature converter is that it can convert only the last signature in a PDF file. This can be a problem if the file contains several keyless signatures or another signature covers a keyless signature. Only the last signature can be converted because every following signature covers all the previous data, including any previous signatures. This is due to the fact that changes to a PDF file are done incrementally [10, p. 44], meaning only the last signature can be converted without changing any signed part of a PDF file.

Another aspect to be considered is that PDF files with converted signatures continue using the old SubFilter value. This means there are signatures in the new format but the value of SubFilter refers to RFC 3161 encoding. The software component which verifies the converted signature has to tell the difference between the old and the new format without using SubFilter. It is possible by trying to decode the signature to one of the formats and check if the result is meaningful, if not, then the other encoding must be used. If there is a need to support the verification of more than one keyless signature in one PDF file or it is not acceptable that old SubFilter's value is used for converted signatures, the integrated conversion method must be used.

The integrated conversion converts a keyless signature before verifying it and does not modify the signature inside a PDF file. This means the PDF toolkit and the Adobe plug-in must be able to do the conversion themselves. Any new software component supporting the keyless signature inside a PDF file must be able to do the conversion. This functionality must remain until it is possible to verify signatures in the old format with the new KSI service.

With the integrated conversion method there are no difficulties with the usage of SubFilter's value. The old signature format is referenced by the old SubFilter value and signatures in the new format are using the value of SubFilter which refers to a TLV encoded keyless signature. The integrated conversion allows to verify all the signatures inside a PDF file if there are more than one keyless signatures or a keyless signature is covered by any other signature type.

Neither of the conversion methods would work if there were any updates to the specification of the new format which would make the conversion impossible.

Table 5 compares previously provided conversion methods. Section 3.4.3 describes how users are influenced by signature conversion.

Table 5. Comparison of conversion methods.

Conversion method / **Aspect**	Integrated conversion	Standalone converter
Limitations with verification	All signatures can be verified	Only the last signature can be verified
Signature stays unmodified	Yes	No
Correct usage of SubFilter	Yes	No
Who should support the method	All software components with PDF integration	Only the PDF toolkit

3.3.4. Proxy for extension

There might be a need to add new functionality to the new KSI service in order to provide extension service with the old protocol. The “Signature type registration method and the One type, two formats method require it if the old KSI service is no longer provided. The proxy for extension is needed to continue supporting the verification of signatures in the old format. The proxy would be able to convert old extension requests but would not be able to convert old signing requests.

The proxy needs to convert the old extension request to the new protocol. The converted request is used with the new KSI service. The received response from the new KSI service must be converted to the old protocol before sending the response to the requester.

There are two options how the proxy functionality can be provided. One option is to develop an external proxy, which is part of the new KSI service. Another option is to embed the proxy functionality inside the PDF toolkit and the Adobe plug-in.

The external proxy solution does not assume any software changes made on the user side. Guardtime needs to develop the proxy and maintain it together with the new KSI service. For the users it seems like the old KSI service is used. The user only needs to change the configuration of their tools to use the proxy instead of the old KSI service. This configuration change must be done for the PDF toolkit and the Adobe plug-in.

It is possible to embed the proxy functionality inside the PDF toolkit and the Adobe plug-in. They would use the new KSI service for extension to verify signatures in the old format. This way there is no need to maintain an external proxy. The user must update the PDF toolkit and the Adobe plug-in to use proxy's functionality without external proxy.

The proxy would stop working if there are changes made to the new protocol specification which would make it impossible to convert extension requests or responses. In this case the Signature conversion method must be chosen as it does not need a proxy.

3.3.5. Summary of development

All three solutions need software updates for the PDF toolkit and the Adobe plug-in to start supporting the new signature format with PDF files. Additionally, there might be a necessity to develop a signature converter and a proxy for extension. Signature conversion functionality can be part of the PDF toolkit. The proxy must be developed to verify signatures in the old format after the old KSI service is shut down. It is possible to integrate the functionality of the proxy inside the PDF toolkit and the Adobe plug-in.

3.4. User activities

Some of the provided solutions assume activities done by the user. This chapter describes what is required from the user with each migration option and how the user is affected.

3.4.1. PDF toolkit

With the "Signature type registration method and the One type, two formats method, there is no need to update the PDF toolkit on the user side if only the old signature format is needed. Without the software update, the user is able to sign and verify using the old KSI service.

When the user tries to verify a signature in the new format using the old PDF toolkit, the toolkit will notify about incorrect signature format and fails to verify it - this is with the One type, two formats method and the Signature conversion method. With the "Signature type registration method, the old PDF toolkit is not able to find the keyless signature in the new signature format from a PDF file.

If the old KSI service is shut down while the “Signature type registration method or the One type, two formats method is applied, the PDF toolkit must be configured to use proxy for verification (see Section 3.3.4). If external proxy method is chosen, then there is no need to do software changes by the user. If embedded proxy method is chosen, then the PDF toolkit must be updated to use it for verifying signatures in the old format.

If there is a need to use signing service after the old KSI service is shut down, the new signature format must be taken into use. To start using the new signature format, the PDF toolkit should be updated.

There is no way to verify signatures in the new format without updating the PDF toolkit. The updated version of the PDF toolkit with the Signature type registration method and the One type, two formats method is able to handle both keyless signature formats. With the Signature conversion method the user must update the PDF toolkit because the old signature format will not be supported anymore. The user cannot create signatures in the old format because the old KSI service is not provided with the Signature conversion method. After the software update the user can sign and verify PDF files using the new KSI service. With the Signature conversion method the user is able to verify signatures in the old format if the integrated conversion method is chosen (see Section 3.3.3).

3.4.2. Adobe plug-in

With the Signature type registration method and the One type, two formats method, there is no need to update the Adobe plug-in on the user side if only the old signature format is used. The Adobe plug-in must be updated to verify signatures in the new format.

With the Signature type registration method, the user has to install the new version of the plug-in to be able to verify signatures in the new format. This means that two Adobe plug-ins must be used at the same time to verify signatures in the new and the old format, because both formats have their own signature type. Section 3.3.2 points out why two plug-ins need to be developed. User experience for signature verification is not affected by using two plug-ins at the same. However, the user has to be aware that there are two different plug-ins. It might be necessary when an update for one of the plug-ins is provided. The user must understand that updating one plug-in does not affect the other one.

If the new signature format is deployed using the Signature type registration method or the One type, two formats method and the old KSI service is shut down then the Adobe plug-in must be configured to use proxy for verification (see Section 3.3.4). If an external proxy is made available, then the user has to update the plug-in configuration. If the embedded proxy solution is chosen, then the user has to update the plug-in because the plug-in has to be able to use the new KSI service for extending signatures in the old format.

With the Signature conversion method the user must update the Adobe plug-in because the old KSI service will be shut down. Depending on a chosen conversion method (see Section 3.3.3), the user might need to convert signatures in the old format before being able to verify them with the updated version of the plug-in.

When the Signature type registration method is applied and the user opens a PDF file with signature encoded according to the old format and having only the new plug-in installed, then the message “At least one signature is invalid” will be shown. The same will happen if only the old plug-in is installed and a PDF file containing a signature in the new format is opened. In both cases, the user will not be able to verify the signature because the signature handler is not found. Older versions of Adobe Reader provided an option by default to select an alternative plug-in for verification if a corresponding plug-in was not found. This option is not available by default for newer versions of Adobe Reader (Adobe Reader DC). Adobe Reader notifies that used signature type is not supported. To choose the plug-in for an unknown signature type, the user needs to change verification behavior settings in Adobe Reader. With the changed settings, the user will be redirected to download the corresponding plug-in. Redirection would work if Guardtime has registered the plug-in with Adobe (Section 2.2.2 describes plug-in registration process).

In case of the One type, two formats method, the behavior is different. When the user opens a signature in the new format using the old version of plug-in, then the plug-in is activated but it fails to decode the signature. As a result, the user fails to verify signatures in the new format. The updated version of the plug-in is able to verify both signature types.

If signatures must be converted by the user with the Signature conversion method, then the updated version of the plug-in will only be able to verify signatures in the new format.

If the user opens a signature in the new format using the old version of the plug-in, then the plug-in fails to decode the signature as it was with the One type, two formats method. The same happens if the user does not convert old signatures and tries to verify them with the updated plug-in. If the embedded conversion method is chosen, then the updated version of the plug-in is able to verify signatures in both formats without a need to convert signatures.

3.4.3. Signatures in old format

If the Signature conversion method with a standalone converter is chosen, signatures in the old format have to be converted to the new format by the user before they can be verified (see Section 3.3.3). This can be difficult if signed files are stored in different locations. Once a signature is converted, there is no need to do it again. It must be kept in mind that once a signature is converted to the new format, there is no way to convert it back to the old format (see Section 1.3.1 for more details).

With the Signature type registration method and the One type, two formats method, no action from the user is required to verify signatures in the old format.

3.4.4. Summary of user activities

With the Signature type registration method and the One type, two formats method, the user does not need to update any software components to continue using only the old signature format. If the old KSI service is no longer provided, then the chosen proxy solution determines how the user should act to be able to verify signatures in the old format. If the old KSI service is shut down, then the user is not able to create signatures in the old format. Verification of signatures in the old format still remains possible.

To use the new signature format, the user must update the PDF toolkit and the Adobe plug-in. The update process is done as any other software update process. With updated software components, the user can create and verify signatures in the new format, signatures in the old format can only be verified. This is in common for all the three methods.

With the Signature conversion method, the user must go through software update process because the old KSI service is not provided with this solution. This means that the old signature format cannot be used to create signatures. There remains a possibility to verify signatures in the old format. With the standalone signature conversion method, the user must convert signatures in the old format before being able to verify them using the PDF toolkit or the Adobe plug-in. With the integrated signature conversion method the user does not have to convert signatures in the old format to be able to verify them when Signature conversion method is used.

4. Security

This chapter analyses the new signature format migration options from a security perspective. Possible vulnerabilities in the integration of the new format of KSI signature, which may be exposed during format migration procedures are analyzed. Ways of mitigating them are also provided. Assessment of security of the KSI service itself is out of the scope of this thesis.

To understand the possible vulnerabilities in the system, it is required understand how the whole system is built. Figure 10 gives an overview of all the system parts which are used to sign and verify PDF files and make the migration possible. The PDF toolkit and the Adobe plug-in are used for verification, which means that by manipulating them it might be possible to show incorrect information about the signature. The PDF toolkit is also used for signing PDF files, leaving a possibility to create incorrect signatures. During the standalone conversion method signatures inside PDF files are modified. This means that it is important to make sure that the unmodified version of the converter is used. An attacker may try to modify the converter so that keyless signatures are converted incorrectly.

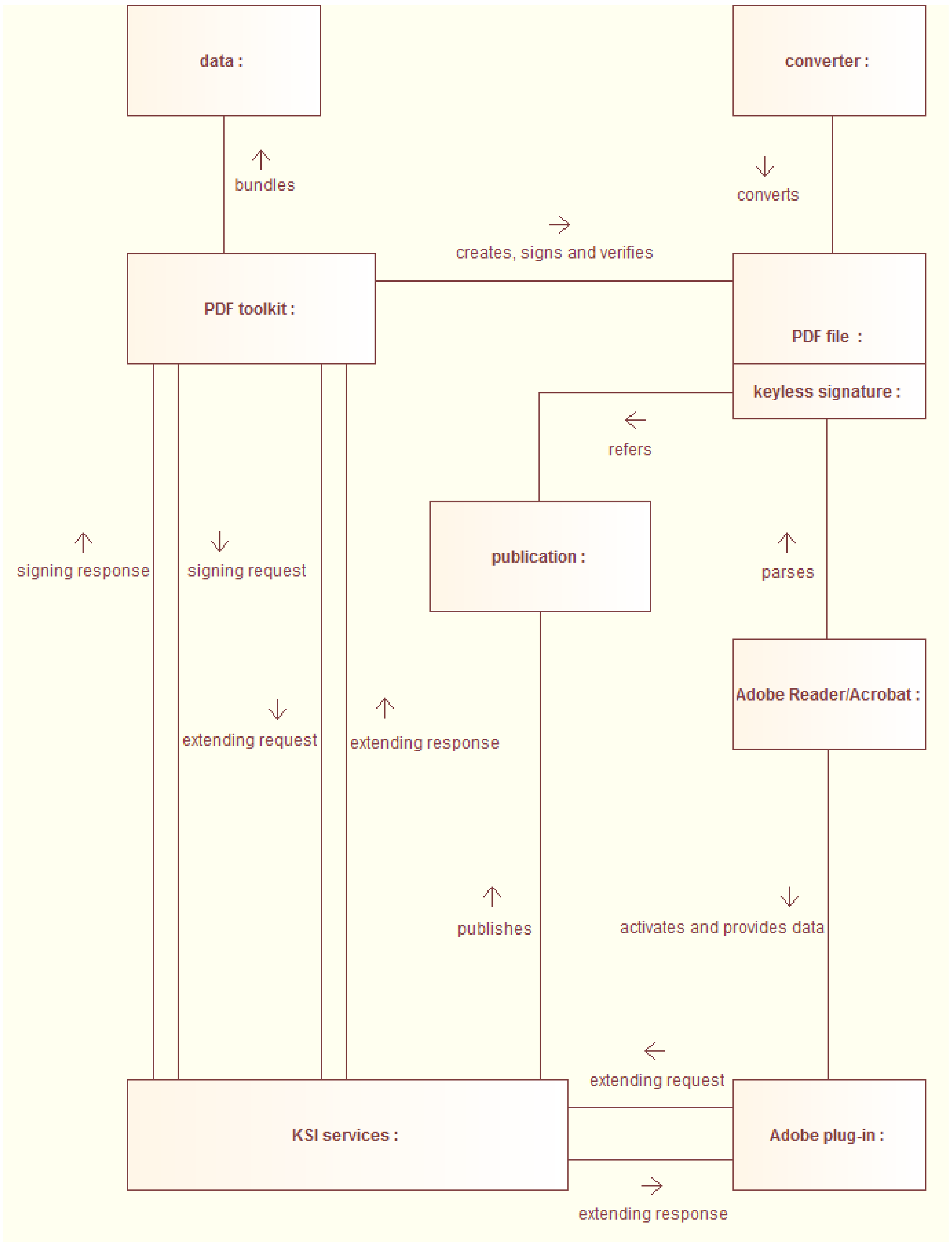

Figure 10. Communication diagram of system parts.

The following two sections introduce means of detecting any modifications made to a keyless signature and what the possible attack vectors are. It is also described what can be done to avoid these attacks.

Any modification made to a keyless signature which is not based on the specification of KSI will be detected during the verification process. To display incorrect information about a keyless signature, the attacker has to modify the signature and also alter the verification process.

4.1. Attack vectors and mitigation

The focus of this section is on how the migration of the new signature format would create possibilities to alter signatures inside PDF files or display incorrect information about a keyless signature to the user.

Activities which may expose vulnerabilities during the new format migration are related to updating software and converting keyless signatures. These are the activities which might be used to influence the verification process and alter keyless signatures.

Even if wrong information about a keyless signature is displayed to the user, there is always a possibility to validate the signature without using the PDF toolkit or the Adobe plug-in. If there are any doubts about verification process, then the user can extract the signature from the PDF file and validate it using public KSI APIs provided by Guardtime via GitHub [19]. Note that these APIs are meant for signatures in the new format. APIs for the old format must be requested directly from Guardtime. Before using the KSI API for signature verification, the user must extract the signature from the PDF file and decode it (see Section 2.1.1). After that the user can use the KSI API for verification. Instructions for signature verification are provided with APIs.

4.1.1. PDF toolkit update

During the migration of the new format there is a need to update the PDF toolkit. Guardtime provides the new version of the PDF toolkit to the user as source code. The user can review the source code to make sure that the received software does what it is intended to do. This means that outsiders should not able to substitute altered versions of the PDF toolkit.

There is a risk that an insider tries to modify the software in a way that the verification is done incorrectly with the updated version of the PDF toolkit. The verification process can be altered so that an incorrect signing time is shown or the user is not notified about modifications made to a signed file. The PDF toolkit is able to create and modify signatures, meaning that the malicious version of the PDF toolkit may alter or delete a keyless signature during the verification process.

The signing process can be altered so that incorrect signatures are created or that signatures are not created at all.

If an insider manages to modify a signature with the PDF toolkit during the verification or the signing process, then modifications to a keyless signature are detected during the verification process. To be able to restore old signatures it is needed to use backups of PDF files. When the signing process is modified so that signatures are not created then there is no way to restore them.

If the PDF toolkit is not trusted, then there is a possibility to verify signatures without the PDF toolkit. This verification process is described in the introduction part of Section 4.1. Also the Adobe plug-in can be used to verify signatures.

4.1.2. Adobe plug-in update

The Adobe plug-in must be updated in order to verify signatures in the new format. This means that the user needs to go through the installation process of the Adobe plug-in. This might create a possibility for an attacker to provide a modified version of the Adobe plug-in to the user.

The user cannot verify the author of the Adobe plug-in. Adobe Reader and Adobe Acrobat only check if the plug-in is signed by a registered developer (see Section 2.2.2 for more details). It is possible that some other developer, who is registered with Adobe, can create an Adobe plug-in which claims that it is the signature handler for Guardtime's keyless signatures. Once installed, it would be activated to verify keyless signatures.

The Adobe plug-in must be received from a trusted entity to avoid the installation of modified version of the Adobe plug-in. The Adobe plug-in received from a third party should not be trusted because it may provide incorrect information about a keyless signature inside a PDF file. It is possible to develop an Adobe plug-in with the same appearance as Guardtime's Adobe plug-in but with different functionality on the background. If the user has not used Guardtime's plug-in before, then the user is not aware of the appearance of the plug-in. The incorrect version of the Adobe plug-in might not notify about changed file or displays wrong signing time.

Since the Adobe plug-in is signed by Guardtime (using Adobe developer's keys), it is provided to user as binary executable. This makes it impossible to verify the source code of the plug-in. The plug-in must be distributed in a trusted way and the user must trust the entity who provided the plug-in. One option is to publish a keyless signature of the plug-in on Guardtime's webpage. The user should not trust any keyless signatures other than the one provided by Guardtime. This way the user can detect if the plug-in has been changed. This assumes that the user is aware of the existence of the published keyless signature.

To sum up, the user should not install Adobe plug-in received from an untrusted source. If there are any doubts that the plug-in is modified, then a keyless signature inside a PDF file can be independently verified using the method described in the introduction part of Section 4.1. Also the PDF toolkit can be used to verify keyless signatures. The keyless signature of the Adobe plug-in should be published to make it possible for the user to detect changes in the Adobe plug-in before installing it.

4.1.3. Signature conversion

With Signature conversion method, there is a need to convert signatures in the old format. One option provided in Section 3.3.3 is to develop a standalone conversion method which modifies signatures inside a PDF file. This opens an opportunity for an attacker to modify signatures during the signature conversion. This attacker must be an insider because signatures are converted on the user side and someone from the outside cannot manipulate the conversion process. Any modifications to a signature will be detected during the verification of a signature.

One option for an attacker is to try to alter signatures during the conversion in a way that a wrong signing time is shown. As told before, it will be detected during the verification process. The other option would be to replace signatures inside a PDF file with empty signatures. After signature deletion, there is no way to make sure when the file was time-stamped or if the file has changed since it was signed. This can be a potential attack vector if an attacker wants to destroy evidence. The attacker could also just destroy needed PDF files. With signature deletion, the PDF file will not be deleted. The attacker might hope that signature deletion will be detected later than a PDF file deletion. Backups can be used to restore deleted signatures.

Source code of the converter should be given to the user. This way the user could review the source code to make sure that converter does what it claims to do.

4.1.4. Summary of attack vectors

The user should review the source code of the PDF toolkit and the standalone converter before using them. The Adobe plug-in should only be installed if the binary executable is received from a trusted source or it is verified that the plug-in has not been changed.

With the Signature conversion method, an insider may try altering already created signatures. The same can happen if a malicious version of the PDF toolkit is used, meaning that signature conversion does not create any additional risks. From a security perspective, none of the adoption options differ from any software update process.

5. Recommendation for choosing migration method

This chapter gives a recommendation for choosing a migration method to start using the new signature format with PDF files. The recommendation is based on the description of each method provided in Chapter 3 and security aspects pointed out in Chapter 4. Aspects to consider with each solution are summarized in Table 6.

The Signature type registration method should be used to clearly distinguish between the two signature formats. The registration of the new signature type takes additional resources and requires the development of a new Adobe plug-in. The new signature type registration allows to use the old and the new format at the same time without the need to convert signatures. When the old KSI service is shut down, the development of a proxy is required to provide the verification of signatures in the old format.

The functionality of the proxy needs to be provided with the Signature type registration method and the One type, two formats method when the old KSI service is shut down. The chosen proxy solution influences user experience and development tasks for Guardtime. With the external proxy method, the user does not need to update software, but has to configure software to use the proxy for extending. The external proxy solution assumes that Guardtime develops an independent proxy and maintains this service as long as it is possible to convert extending requests from the old format to the new format. Maintaining such a service takes resources. The other option is to use the embedded version of a proxy where the functionality of the proxy is implemented inside the PDF toolkit and the Adobe plug-in. With this option, the user needs to update the PDF toolkit and the Adobe plug-in when the old KSI service is shut down and there is a need to verify signatures in the old format. The functionality of the embedded proxy must be integrated into every new software which is going to handle keyless signatures inside PDF files.

From the user experience perspective, it is desirable to not force users to update their software. This is only possible if the user does not want to use the new signature format. The Signature type registration method and the One type, two formats method support this approach. The choice between these two solutions depends on the need to register the new signature type with Adobe. In both cases, the user only needs to update software when the new signature format is needed or when Guardtime stops providing the old KSI service.

With the Signature conversion method the user must update software immediately because the old KSI service is not provided. Signature conversion has two possible options to verify signatures in the old format. These options differ in user experience and functionality of the PDF toolkit and the Adobe plug-in. For the user, it would be more convenient to use the integrated conversion method. With the integrated conversion method, the user only needs to update the PDF toolkit and the Adobe plug-in. The updated software would be able to do the conversion and verify signatures in the old format using the new KSI service. This means that every new software which is going to handle keyless signatures with PDF files must be able to convert signatures in the old format. With this conversion method, the SubFilter value inside the signature dictionary refers to the correct encoding. With the standalone conversion method, where the converted signature is stored inside a PDF file, the value of SubFilter contradicts with the real encoding. This is because converted signatures are TLV encoded, but SubFilter's value cannot be changed. With this approach, signature conversion needs to be done by the user. The standalone conversion method is preferable from the code maintenance perspective. The updated version of the PDF toolkit and the Adobe plug-in or any new software do not need to support the old signature format. The standalone conversion method makes it easy to stop supporting the old signature format.

The most time consuming solution is the Signature type registration method. This comes from the time required to register the new signature format and the plug-in with Adobe. Providing the new Adobe plug-in for the verification of the new signature format may take weeks (see Section 2.2.2). With the One type, two formats method and the Signature conversion method, the time would be spent for the development of the PDF toolkit and the Adobe plug-in.

From the security point of view, all three solutions do not differ from any software update. With all the solutions, there is a possibility that an insider tries to manipulate the signing and verification process or alter keyless signatures inside PDF files. In case of any doubts, it is possible to verify keyless signatures without the PDF toolkit and the Adobe plug-in. Any modifications made to a keyless signature will be detected by the KSI technology.

To sum up, one of the main questions derives from the decision if there is a need to convert signatures to the new format or convert extension requests to the new format. With the Signature type registration method and the One type, two formats method, there is a need to choose a proxy solution, but signatures do not need to be converted. With the Signature conversion method, signature conversion is required, but a proxy is not needed. All three solutions enable shutting down of the old KSI service and provide a possibility to verify signatures in the old format after the new format is taken into use. The Signature type registration method just needs time for signature type registration with Adobe.

Table 6. Summary of adaptation methods.

Solution / **Aspect**	"Signature type registration method"	"One type, two formats method"	"Signature conversion method"
What is required to continue using signatures in the old format	No actions required. Full support – signing and verification.	No actions required. Full support– signing and verification.	Update software + convert signatures[1]. Only verification supported.
What is required for using signatures in the new format	Registration process with Adobe + Update software.	Update software.	Update software.
Adobe	Required to contact for a new signature type and a new plug-in registration.	No contact required.	No contact required.
Security concerns	Same as any other software update.	Same as any other software update.	Same as any other software update.

[1] Depending on a chosen conversion method (described in Section 3.3.3) there might not be a need to convert signatures.

What is required to shut down the old KSI service	Proxy.	Proxy.	Signature conversion.
Time required to start supporting new signature format	Development of Adobe plug-in and PDF toolkit + new signature type and plug-in registration with Adobe.	Development of PDF toolkit and Adobe plug-in.	Development of PDF toolkit and Adobe plug-in.

Summary

The goal of this paper was to analyze possible options for migrating to the new format of keyless signature with PDF files. The outcome is three possible solutions.

Portable Document Format specification was analyzed to be able to provide possible options for the migration. All the possible solutions were discussed from the user experience, Guardtime's commitment to development and security perspective.

The first solution proposes to start to using the new signature format by going through signature type registration with Adobe. When the old KSI service will be shut down, then the usage of a proxy is needed. The new signature format cannot be taken into use right away because the registration process with Adobe takes time. This solution makes it possible to clearly distinguish between the two signature types.

With the second solution, the old and the new signature formats are using the same registered signature type. This means that there is no need to go through the registration process with Adobe. It is not required to use a proxy until the old KSI service is available.

The third solution describes using the new signature format instead of the old format. Signatures in the new format would use the same signature type as the old format. Signatures in the old format have to be converted to verify them when this solution is applied.

All three solutions support the shutdown of the old KSI service. They all provide the possibility to verify signatures in the old format using the new KSI service.

References

[1] R. C. Merkle, "Protocols for Public Key Cryptosystems," in *IEEE Symposium on Security and Privacy*, 1980, pp. 122-134.

[2] A. Buldas, R. Laanoja and A. Truu, ""Efficient Quantum-Immune Keyless Signature with Identity"," *Cryptology ePrint Archive http://eprint.iacr.org/2014/321.pdf,* vol. 321, 2014.

[3] S. Haber and W. S. Stornetta, "Secure Names for Bit-Strings," in *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, ACM, 1997, pp. 28-35.

[4] Guardtime, "ksi-sdk-samples/ExtendingSamples," [Online]. Available: https://github.com/GuardTime/ksi-sdk-samples/blob/master/java-sdk/src/test/java/com/guardtime/ksi/samples/ExtendingSamples.java. [Accessed 22 4 2016].

[5] ISO/IEC, "Information technology -- Security techniques -- Time-stamping services -- Part 3: Mechanisms producing linked tokens," 2004.

[6] C. Adams, P. Cain, D. Pinkas and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)," 2001.

[7] Guardtime, "TLVElement," [Online]. Available: http://guardtime.github.io/ksi-java-sdk/com/guardtime/ksi/tlv/TLVElement.html. [Accessed 22 4 2016].

[8] Guardtime, "KSISignature," [Online]. Available: http://guardtime.github.io/ksi-java-sdk/com/guardtime/ksi/unisignature/KSISignature.html. [Accessed 22 4 2016].

[9] A. Truu, "Standards for hash-linking based time-stamping schemes," 2010.

[10] Adobe Systems Incorporated, "Document management - Portable document format - Part 1: PDF 1.7," 1 7 2008. [Online]. Available:

http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf. [Accessed 14 4 2016].

[11] Adobe, "Acrobat_DigitalSignatures_in_PDF," [Online]. Available: https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSig/Acrobat_DigitalSignatures_in_PDF.pdf. [Accessed 14 04 2016].

[12] ETSI, "Electronic Signatures and Infrastructures (ESI);PDF Advanced Electronic Signature Profiles;Part 4: PAdES Long Term - PAdES-LTV Profiles," 12 2009. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102700_102799/10277804/01.01.02_60/ts_10277804v010102p.pdf. [Accessed 14 4 2016].

[13] Adobe, "System requirements | Adobe Acrobat Reader DC," [Online]. Available: https://helpx.adobe.com/reader/system-requirements.html. [Accessed 18 04 2016].

[14] Adobe, "System requirements | Acrobat family of products," [Online]. Available: https://helpx.adobe.com/acrobat/kb/system-requirements-acrobat-family-products.html. [Accessed 18 04 2016].

[15] Adobe, "Developing and enabling an Acrobat Reader plug-in," [Online]. Available: http://help.adobe.com/en_US/acrobat/acrobat_dc_sdk/2015/HTMLHelp/#t=Acro12_MasterBook%2FPlugins_ReaderPlug%2FDeveloping_and_enabling_an_Acrobat_Reader_plug-in.htm. [Accessed 14 4 2016].

[16] Adobe, "Enabling the plug-in for Acrobat Reader," [Online]. Available: http://help.adobe.com/en_US/acrobat/acrobat_dc_sdk/2015/HTMLHelp/#t=Acro12_MasterBook%2FPlugins_ReaderPlug%2FEnabling_the_plug-in_for_Acrobat_Reader.htm. [Accessed 14 4 2016].

[17] Adobe, "How Acrobat uses the plug-in list," [Online]. Available: http://help.adobe.com/en_US/acrobat/acrobat_dc_sdk/2015/HTMLHelp/#t=Acro12_MasterBook%2FPlugins_Introduction%2FHow_Acrobat_uses_the_plug-in_list.htm. [Accessed 14 4 2016].

[18] Adobe, "Submitting a request to register your plug-in," [Online]. Available: http://help.adobe.com/en_US/acrobat/acrobat_dc_sdk/2015/HTMLHelp/#t=Acro12_MasterBook%2FPlugins_Introduction%2FSubmitting_a_request_to_register_your_plug-in.htm. [Accessed 14 4 2016].

[19] "About Guardtime," [Online]. Available: https://github.com/GuardTime/. [Accessed 26 4 2016].