iText 7

# Blockchain for PDF Documents

A solution to secured invoices and digital signatures in PDF documents.

# Blockchain for Documents

## What do blockchain and PDF have in common? Security

iText Software

This book is for sale at http://leanpub.com/blockchainfordocuments

This version was published on 2018-09-26

# Contents

We all know blockchain because it's the technology used by Bitcoin in the context of cryptocurrency. "Virtual money" or "Digital currency" is only one application of Distributed Ledger Technology (DLT). At iText, we've developed a series of patents that describe mechanisms:

- to automate document workflow and version management,
- to ensure document integrity, authentication, and non-repudiation,
- to provide long-term validation, and
- to manage document identification and retrieval.

You may already have read our RefCard on DZone, entitled Blockchain and Distributed Ledger Technology for Documents. Or you've seen the article on Forbes explaining the rationale of our Blockchain efforts: Can Blockchain Solve Your Document And Digital Signature Headaches? This Forbes article was inspired by an OracleOne talk by Joris Schellekens. The patents were filed on December 22, 2016. They were already granted in Belgium, and they are now pending internationally (ETA: June 2019). We've also created an iText add-on pdfChain that is released on github.

We've been evangelizing these mechanisms at events such as:

- the ETDA workshop for e-Tax Invoice (Thailand),
- the PDF Days Washington DC (USA),
- OracleCode Los Angeles (USA),
- the Great Indian Developer Summit (India),
- the PDF Days Europe (Germany),
- the Document Strategy Forum, Boston (USA),
- and so on.

In the next handful of articles, we'll take a closer look at specific topics related to Blockchain and documents.

# Why iText and Blockchain?

When iText was first released in the year 2000, its author, Bruno Lowagie, wrote down the following mission statement:

> We want to enable the paperless world,
> pushing the limits of digital document interactivity!

With iText, we wanted to replace paper-based workflows with a workflow involving digital documents. We wanted to accomplish this by emphasizing the added value of digital over paper. Today, this mission seems almost accomplished:

- Digital invoices are legally binding in many countries,
- On January 1, 2023, the National Archives and Records Administration (NARA) will stop accepting paper documents: all transferred permanent or temporary records to the National Archives and Records Administration must be in electronic format.
- Printer use is dropping significantly, as shown in figure 1.

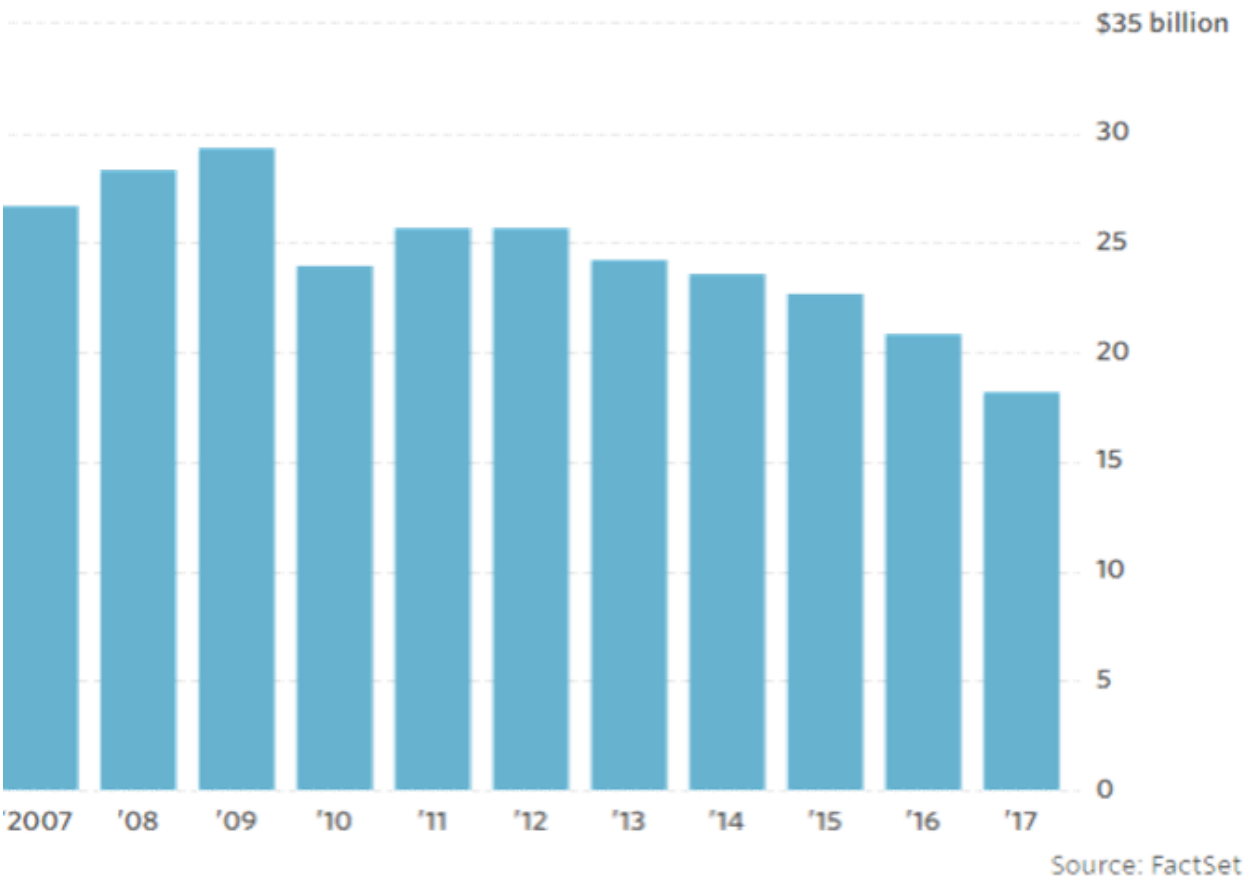


**Figure 1: HP printing revenue**

# The Dematerialization of the Document

At the same time, we see another evolution. We thought that PDF was here to stay, but when boarding a plane, we stopped using paper boarding passes, many of us are using a two-dimensional bar code in an app. When we make a purchase using an app, we often pay without receiving an invoice as a document; often it's merely data presented to us in an app. The mentality towards documents is changing:

- In the past, users didn't trust a digital document that didn't look like a paper document,
- The mobile web changed this mentality: content can look differently, but still have the same validity.

While we're still working on the further dematerialization of paper, we notice the beginning of the dematerialization of the document. As shown in figure 2, we're evolving from paper documents to digital document, and from digital documents to responsive content.

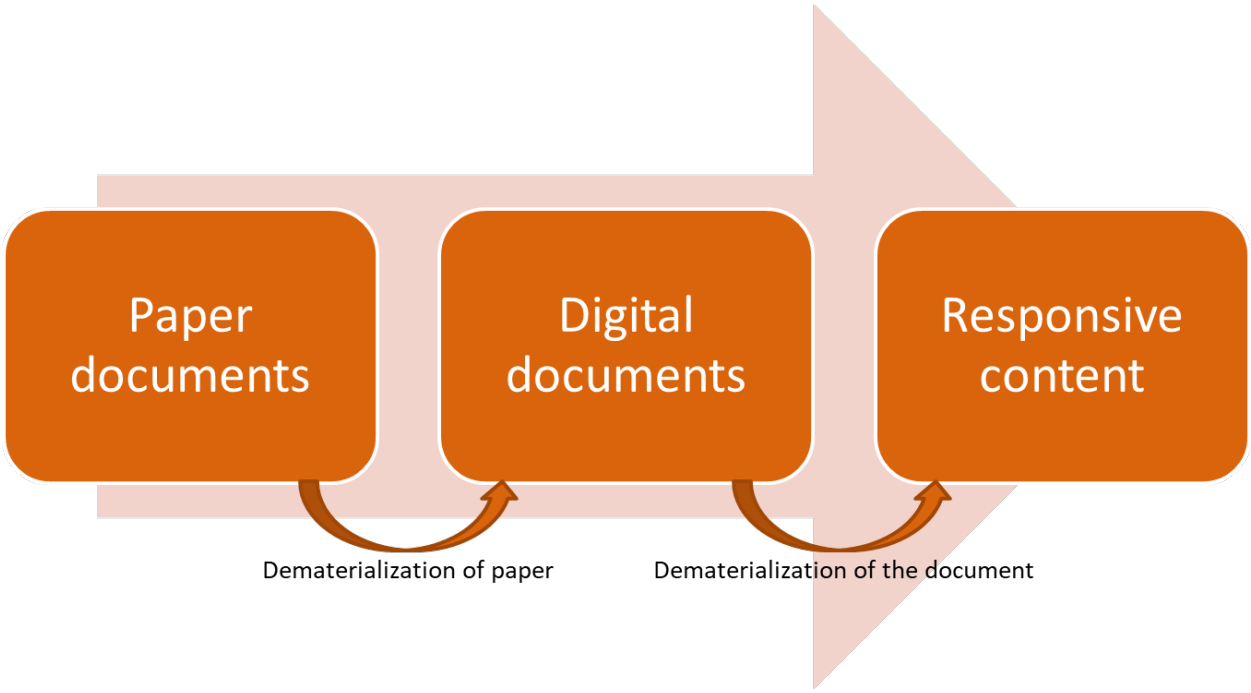


**Figure 2: dematerialization of paper / dematerialization of documents**

As we are in the document business, the dematerialization of the document could pose a threat for our industry. Fortunately, we have some ideas to turn this threat into an opportunity.

## How to turn a threat into an opportunity?

First of all, there are some known flaws with respect to data served to apps:

- **Reliability**: was the data presented correctly?
- **Immutability**: what if the data in the database changes (deliberately or undeliberately)?
- **Security**: who has access to the data?

We also aren't blind to some known flaws in PDF:

- It's difficult to unlock data from a PDF that isn't created in an accessible way. We can solve this using tools, but it's not always trivial to extract data from a PDF in a reliable way.
- Digital signatures are a pain: signatures need to be applied sequentially, you need a Certificate Authority (CA) and a Timestamp Authority (TSA), not all viewers support signatures (Preview, mobile viewers,...), and so on.

By working on solutions for these flaws, we can adapt to the new situation, and create new business by investing in new technologies:

## Next-Generation PDF

We aren't the only company in the document industry who has noticed the evolution of the dematerialization of the document. As a member of the PDF Association, we've joined forces with several other companies to develop a new specification that is currently codenamed Next-Generation PDF. Next-Generation PDF will marry core PDF capabilities to the flexibility of web technologies. Next-Generation PDF should also emphasize the value of PDF as a data container.

## Distributed Ledger Technology and Blockchain

Blockchain is a distributed database that serves as an irreversible and incorruptible repository for permanent records. While studying the concepts of distributed ledger technology, we discovered that Blockchain could solve some of the issues that are inherent to digital signatures in PDF. Based on our research, we filed a series of patents describing mechanisms that we are now developing together with different partners.

# What's the market?

Moody's Investors Service (MIS) released a report â€œCredit Strategy – Blockchain Technology: Robust, Cost-effective Applications Key to Unlocking Blockchain's Potential Credit Benefitsâ€ identifying 25 top blockchain use cases, from a list of 120.

| Financial institutions | Corporates | Governments | Cross-Industry |
|---|---|---|---|
| International payments | Supply chain management | Record management | Financial management & accounting |
| Capital markets | Healthcare | Identity management | Shareholders' voting |
| Trade finance | Real estate | voting | Record managment |
| Regulatory compliance & audit | Media | Taxes | Cybersecurity |
| Anti-money laundering & know your customer | Energy | Government & non-profit transparency | Big data |
| Insurance | | Legislation, compliance & regulatory oversight | Data storage |
| Peer-to-peer transactions | | | Internet of things |

**Figure 3: Selected potential Blockchain Use Cases (source: Moody's Investors Service)**

By tapping into that market, we can turn a threat for our business into an opportunity.

# Basic Concepts (Glossary)

If you are new at Blockchain and PDF, you'll find this short glossary interesting.

## Concepts related to Blockchain:

**Distributed database**: we talk about a distributed database when the storage devices aren't attached to one central processing unit, but are spread across a network. Some examples include:

- NoSQL databases, with well-known implementations such as MongoDB and CouchDB,
- Hadoop, which is an open source framework for storing data and running applications on clusters of hardware devices,
- Distributed Ledger Technology,
- …

**Node**: a node is a connection point in a distributed network that can receive, create, store or send data from and to other nodes in that network.

**Ledger**: a ledger is a collection of permanent, final, definitive records of transactions.

**Ledger record**: a ledger record is an entry in the ledger containing information about one or more transactions.

**Distributed ledger technology (DLT)**: DLT is a type of distributed database technology with the following characteristics:

- The records can be replicated over multiple nodes in a network (decentralized environment),
- New records can be added by each node, upon consensus reached by other nodes (ranging from one specific authoritative node to potentially every node),
- Existing records can be validated for integrity, authenticity, and non-repudiation,
- Existing records can't be removed, nor can their order be changed,
- The different nodes can act as independent participants that don't necessarily need to trust each other.

Combined, these characteristics make DLT a great way to keep a ledger of records in a trustless environment.

**Blockchain**: blockchain is a type of DLT in which records are organized in blocks that are appended to a single chain using cryptography and distributed consensus. Each block contains a timestamp and a link to a previous block. This ensures that data in any given block can't be altered retroactively

without the alteration of all subsequent blocks. This approach makes blockchain technology a good choice for the recording of events, records management, provenance tracking, and document lifecycle management.

**Centralized, decentralized, or distributed storage**: there are blockchain systems where a single instance of the ledger is stored on a central server that acts as the broker of the data. Usually, the data lives on different nodes. In the case of decentralized ledger storage, a copy of the ledger is stored on specific "super-nodes". In a distributed architecture, the ledger is replicated on every node.

**Permissionless or permissioned blockchain**: a permissionless blockchain is a DLT system where no authorization or authentication is needed, and nodes and users are unknown. In a permissioned blockchain, nodes must have a member identity; authorization and authentication is mandatory.

**Public or private blockchain**: in a public blockchain, any node can join to read blocks and records, append records, and to participate in the consensus mechanism. In a private blockchain only nodes that have been granted authority have that access.

**Centralized, decentralized, or distributed ledger control**: in case of centralized control, one authority, e.g. a central server, decides on the validation of a new block of records. With decentralized control, a central authority delegates the validation of new blocks to a limited number of nodes. In a distributed architecture, all the nodes work together using a consensus mechanism to validate a new block.

**Consensus mechanism**: a consensus mechanism is an agreement among all the nodes regarding the validity and consistency of the records and blocks that are being added to the blockchain. The consensus mechanism also guarantees the order of the records in a distributed ledger. A consensus mechanism can be implemented in many different ways (e.g. in the context of Bitcoin, a proof-of-work is needed), but that would go beyond the scope of this ref card.

Although blockchain isn't a synonym for DLT, the industry started using blockchain as the common name for all kinds of distributed ledger technologies, probably because blockchain sounds easier than DLT, and is a more catchy word to market.

You'll read articles criticizing Blockchain (or DLT) arguing that it's slow. Authors of such articles often refer to Bitcoin, which is a public, decentralized system that uses a proof-of-work. They overlook that our examples are completely agnostic of the type of Blockchain that is used. A permissioned Blockchain with centralized storage and centralized ledger control without a consensus mechanism is much faster.

Blockchain shouldn't be seen as a competitor to traditional database systems. Traditional databases will probably be faster and easier to use than distributed ledger technology. Moreover, Blockchain shouldn't be used to store data that you don't want to expose to the outside world. The most important advantage of using Blockchain when compared with other database systems, is the immutability of the records. Those records should be used to store metadata, such as the digest message of the bytes of a PDF document.

# Concepts related to PDF

**Document**: a document is a piece of written, printed, or electronic matter that provides information or evidence or that serves as an official record.

**Portable Document Format**: the Portable Document Format (PDF) is a file format for capturing and sending electronic documents described in a series of standards, such as:

- ISO 32000-2: the most recent core PDF specification,
- ISO 19005-2 and 19005-3: the most recent standards for the long-term preservation of PDF documents, aka PDF/A,
- ISO 14289-1: the standard for universal accessibility in the context of PDF, aka PDF/UA,
- ETSI TS 102 778: a series of standards for PDF Advanced Electronic Signatures (PAdES)
- ZUGFeRD: a standard for invoices that combines PDF/A-3 and the UN/CEFACT Cross Industry Invoice (CII) standard.

**PDF ID**: every PDF can be identified by an ID that consists of a pair of identifiers. The first identifier is created at the time a new PDF file is created, and it's permanent in the sense that it won't change when the PDF is updated. The second identifier is initially identical to the first part, but it changes each time the document is updated. The ID of a PDF needs to be unique. Two PDF documents with the same ID should be exact copies of each other; both files should contain the exact same bytes in the exact same order. If the two identifiers of a document's ID pair are identical, then you know that the document is a first version. If the first identifier of two different PDF documents is identical, but the second identifier is different, then you know that both documents are somehow related to each other.

These concepts will be important once you start reading about the mechanisms iText has patented.

# About our Blockchain patents

The mechanisms outlined in our patents can be used for any type of file (documents in any format, raw data in JSON or XML format, and so on), but we've chosen to focus on PDF because the industry has accepted the format as being reliable, trustworthy, and ubiquitous. Furthermore, PDF has some unique features. The concept of the PDF ID is one example. The fact that PDF can be used as a container for different types of media (XML, video, audio,…) is another one.

# PDF and Digital Signatures

The digital signature technology, as described in ETSI's PDF Advanced Electronic Signatures (PAdES) standard and ISO's ISO-32000 standards, was created to meet specific requirements.

## Requirements that PDF digital signatures need to meet

The purpose of the PDF digital signatures functionality is to ensure:

- **Integrity**: we want assurance that the content of the document hasn't been changed somewhere in the workflow,
- **Authenticity**: we want assurance about the origin of the document, e.g. about the identity of the author or the instance that issued the document,
- **Non-repudiation**: we want assurance that the issuing party can't deny its authorship,
- **The time of signing**: we want assurance about the date and time a document was signed.

ISO-32000-2 also introduces a concept that was first published in PAdES-4:

- **Long-term validation (LTV)**: we want the assurance that the integrity, authenticity, non-repudiation, and the time of signing can still be validated on the long term.

Let's find out how this works, step by step.

## Creating a digital signature for PDF

In figure 1, we can see how the bytes are organized in a PDF file.

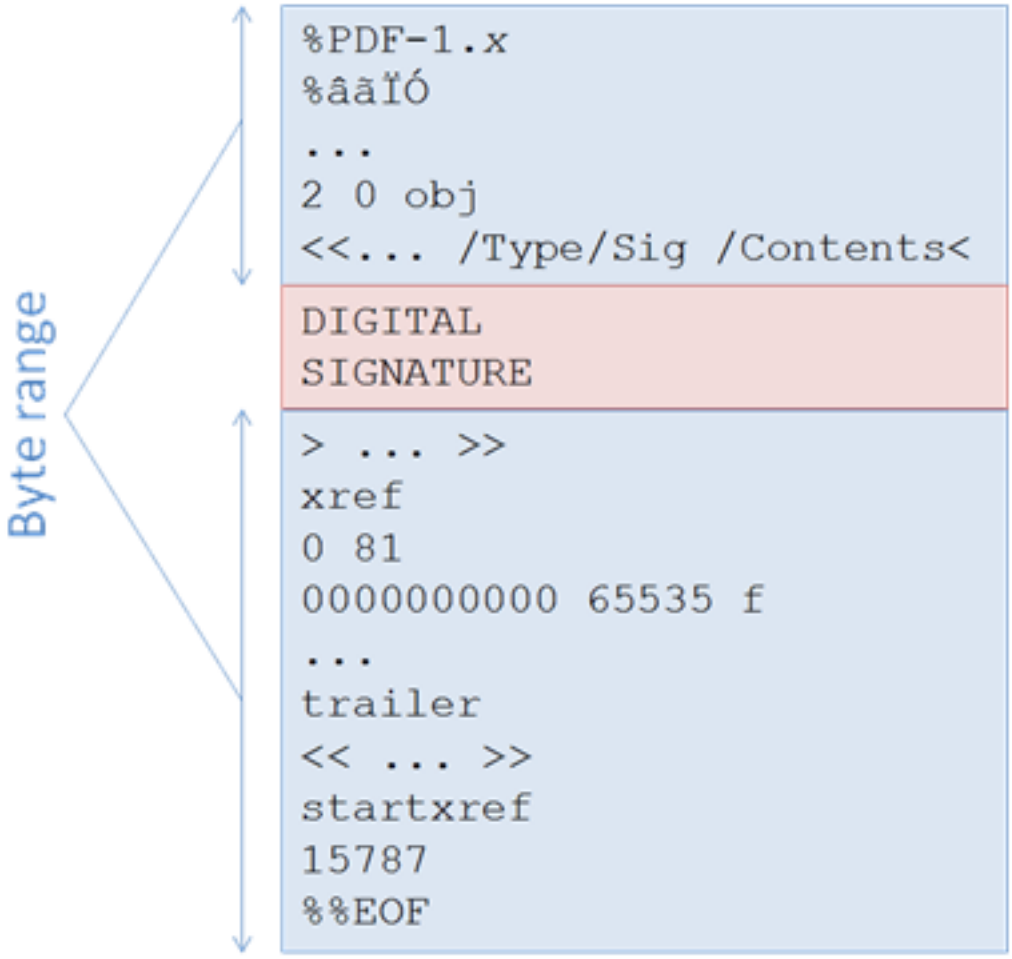


**Figure 1: Digital Signature in PDF**

When we sign a PDF document, we take all the bytes of the file, except for the area where the digital signature will be stored. We create a message digest using a cryptographic hash function from these bytes. We sign this hash value using a private key, and we store that signed hash along with some extra information (the signer certificate, signature attributes,...) that is signed or unsigned, depending on the type of signature container.

Signing is done using Public Key Infrastructure (PKI). The signer owns a key-pair, consisting of a public key and a private key. We talk about encryption when someone uses the public key to encrypt a message. Only the party who possesses the corresponding private key can decrypt the message. We talk about signing when someone uses the private key to encrypt a message. Everyone who has access to the public key can decrypt the message.

A document can be signed more than once, by different signers. This needs to happen sequentially to avoid that every new signer invalidates the previous signers' signatures. This is shown in figure 2.

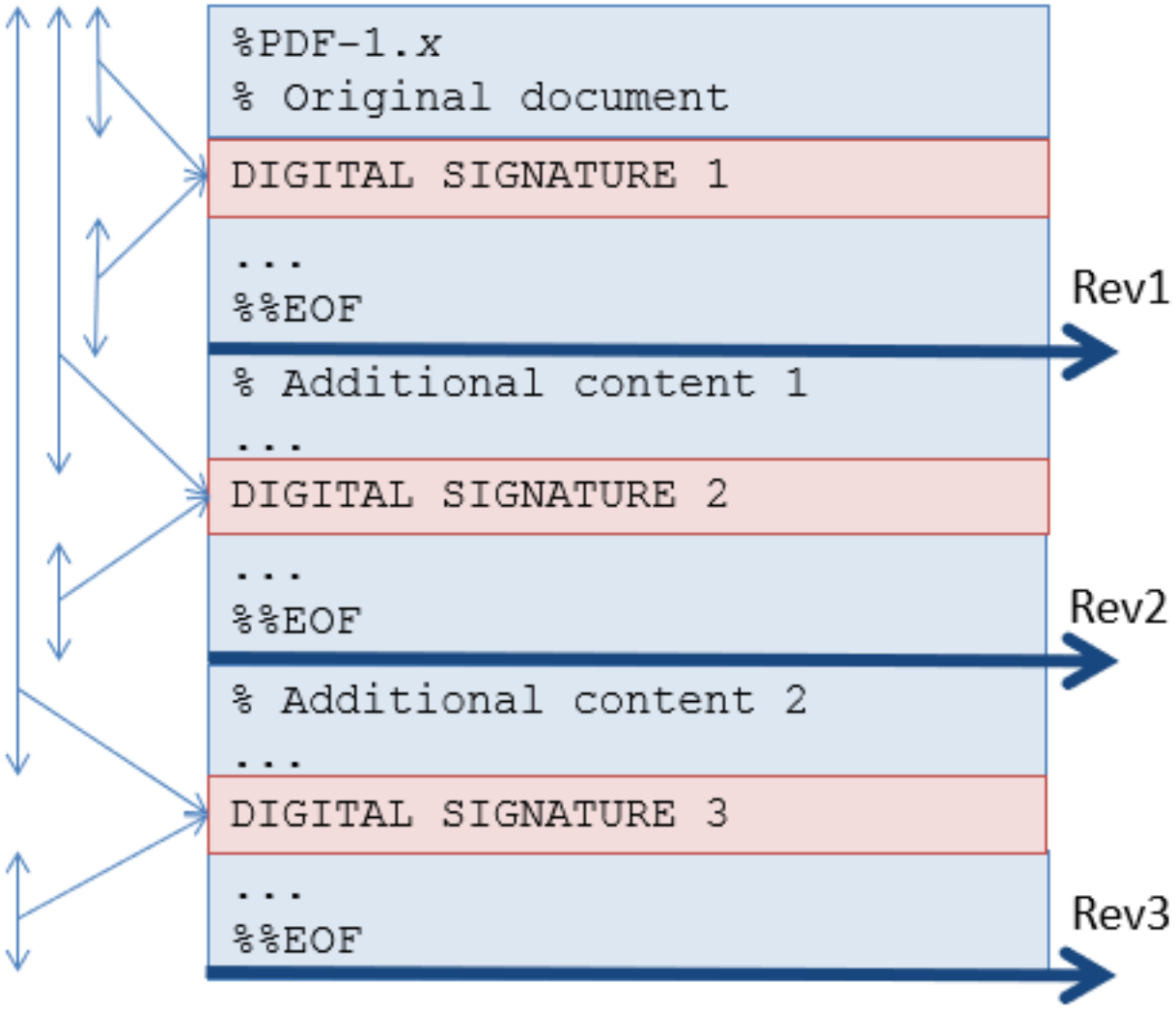


**Figure 2: Sequential Signatures**

This document has three revisions. Revision 1 is signed by signer 1. Revision 2 is signed by signers 1 and 2. Revision 3 is signed by signers 1, 2, and 3. Every new signer signs all the previous signatures.

The minimum information that needs to be stored inside the signature consists of:

- The signed message digest, and
- The signer's certificate (containing the public key that corresponds with the private key that was used for signing).

Best practices also require the presence of:

- The rest of the certificates in the certificate chain (leading up to the root certificate),
- Revocation information, in the form of a Certificate Revocation List (CRL) or an Online Certificate Status Protocol (OCSP) response,

- A timestamp.

We'll need all of these elements to verify the signature.

## Are our initial requirements met?

When you receive a digitally signed PDF, you can verify the integrity of the document using the following steps:

- Decrypt the signed hash using the signer's certificate; the result is message digest hash1,
- Hash the bytes of the PDF, excluding the bytes of the signature itself; the result is a message digest hash2,
- Compare hash1 with hash2; if they aren't identical, the document has been tampered with.

This is a way to check the **integrity** of the document.

The identity of the signer is stored in the signer's certificate. If the certificate is self-signed, there is no way to verify the authenticity, or to get any assurance of non-repudiation. To solve this problem, the signer needs to involve a certificate authority (CA) who will vouch for the information about the signer's identity. The signer's certificate will be on one end of the certificate chain; the root certificate of the CA will be on the other end. When verifying a signature, we'll check the revocation information provided by the CA and all the trusted parties in-between. The signer won't be able to deny that he signed the document unless he can prove that his private key was compromised, in which case he should have revoked it. This meets the requirements of **authenticity** and **non-repudiation**.

Best practices also imply that you involve a Timestamp Authority (TSA). A TSA service accepts your signed hash and signs it with its own private key adding a timestamp that can be trusted. This meets the requirement regarding **the time of signing**.

This timestamp is stored inside the digital signature, but PAdES-4 also introduced the concepts of a Document Security Store (DSS) and Document Timestamp (DTS) signatures. If a signed document is missing Validation Related Information (VRI), this information can be added to the PDF in a DSS. See figure 3 where we add missing certificates and missing revocation information.

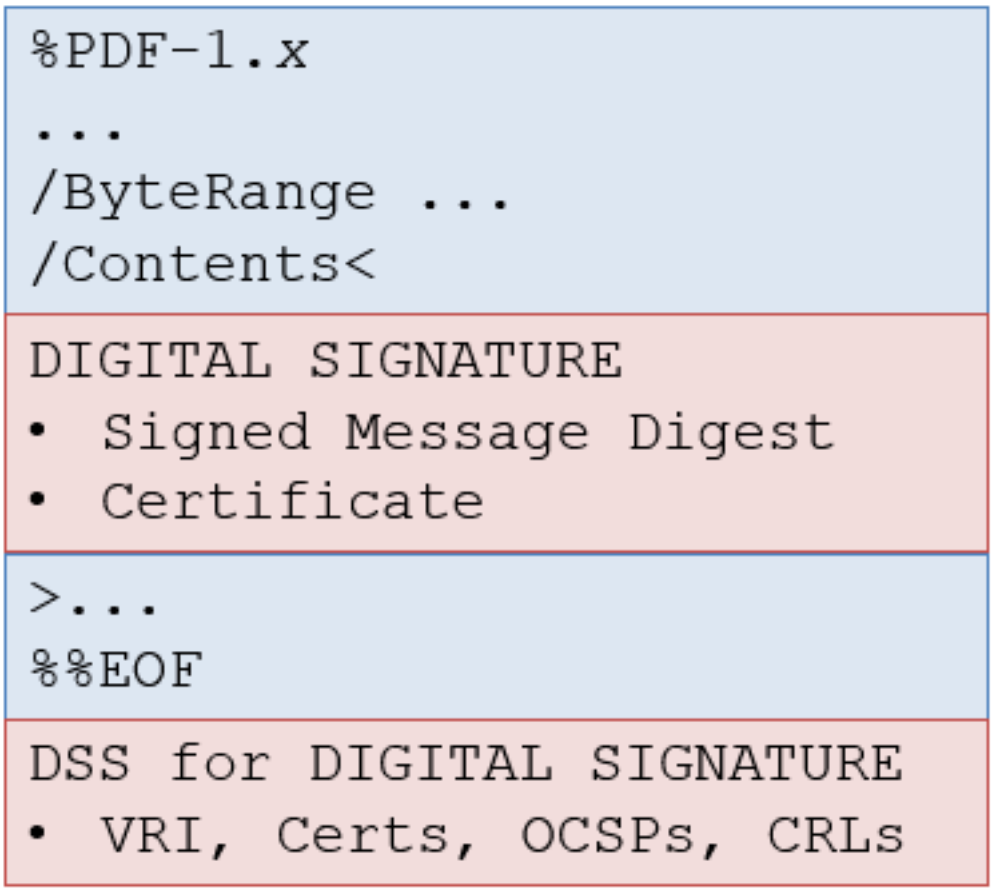


**Figure 3: Adding a Document Security Store (DSS)**

Digital signatures expire, either because certificates expire, or because the algorithms are proven to be flawed in which case the signature loses its trustworthiness.

For instance: SHA-1 once used to be considered as a safe cryptographic hash function, but it was deprecated by NIST in 2011 because it was considered broken in theory. In 2017, researchers from CWI and Google succeeded in breaking SHA-1 in practice. They succeeded in altering the content of a PDF file without breaking the signature proving the theoretical flaw in the SHA-1 algorithm.

With DTS signatures, we can extend the life of the digital signatures inside a document beyond their expiration. To achieve this, the most recent (valid!) VRI should be retrieved and added in a DSS, after which the document should be signed with a DTS. This signature should be created by TSA using the most recent hashing and encryption algorithms; see figure 4.

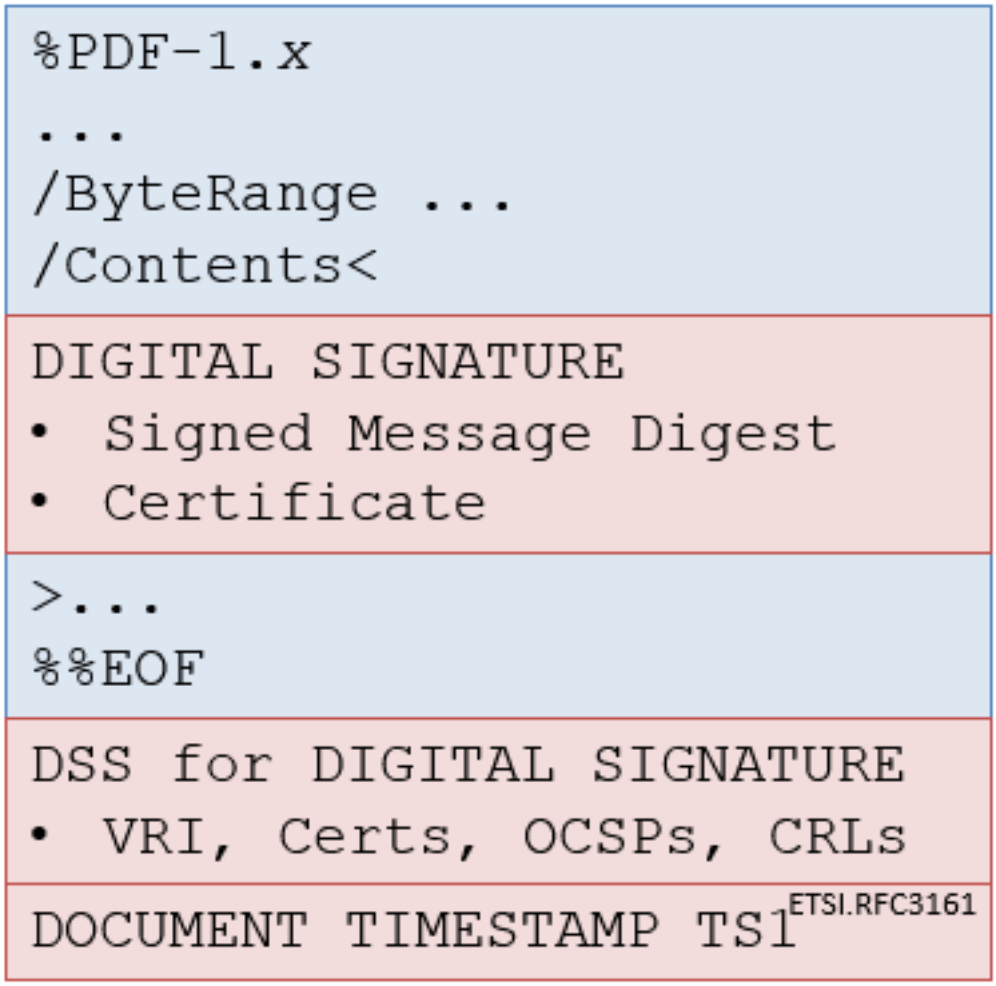


**Figure 4: Adding a Document Timestamp Signature**

Obviously, the DTS also expires after a certain time, hence the operation needs to be repeated to further extend the life of the signatures in the document. This is shown in figure 5.

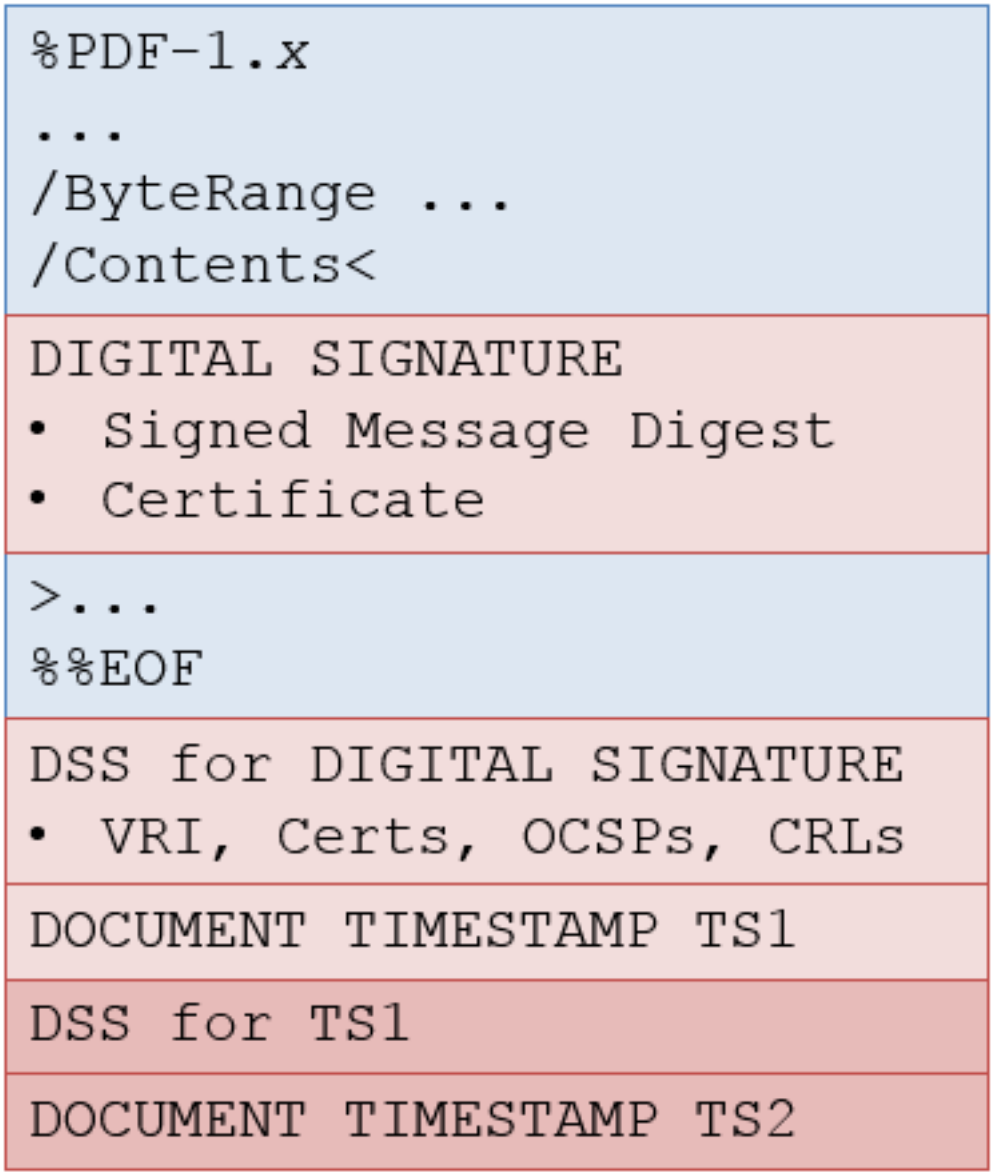


**Figure 5: Extending the Life of a Digital Signature**

This is a condensed description of how digital signatures work in PDF. This functionality works, and there's no reason why it won't continue to work in the future. Nevertheless, you might have detected some potential problems.

## Potential problems inherent to digital signatures and PDF

The adoption of digital signatures in PDF was slow, among others because one or more of the following reasons:

- If you want people to trust the signatures in your document, you need to involve some central authorities, such as CAs and TSAs. It would be great if we could think of a system that doesn't require any central authorities.
- Documents that need to be signed by different people can only be signed sequentially. Think of a conference call that requires a dozen people to sign an NDA before the call can take place. Person A receives the NDA document, signs it, and sends it to person B. Person B receives it, signs it, and sends it to person C. And so on. Now suppose that person D is on vacation the week before the conference call. In that case, all the people who come after person D can't sign until person D returns. There's a chance that person E, F, G, H,… won't be able to add their signature in time for the call. (This example is based on a true story.)
- As documents are signed sequentially, person A will own a copy with a single signature, person B will own a copy with two signatures, and so on. If 12 people have to sign, person L will own a copy of the document with 12 signatures. It is then the responsibility of person L to send the document that contains all the signatures to all the previous signers. Unfortunately, this doesn't always happen. Often, people early in the chain only have copies that are missing one or more signatures. (This example is based on a recurring true story.)
- Sometimes it's hard to know if you're signing the correct version of a document. Think of an agreement that was drafted in a process that involved tough negotiations. During the negotiations, different versions of the agreement were sent back and forth. There's a risk that eventually the wrong version of an agreement is signed due to a confusingly high number of versions created by different people introducing confusion version numbers. (As you may have guessed, this example is also based on a true story.)
- LTV is a cumbersome process that heavily relies on central authorities. Suppose that you've kept a signature alive for a century, and you discover that you can't create a new DSS because of a sudden, perpetual unavailability of a TSA.

PDF documents that aren't signed often have the problem that you don't know if they can be trusted. Once the document is consumed offline, there is no green bar on top of the viewer that says: this document was served to you using the HTTPS protocol, and by the way: you can trust the domain from which this page is served.

If you have a URL to a PDF document, e.g. the address of the PDF reference, you can encounter the problem that the URL doesn't work anymore. You get a 404 error message, because the document was moved to another location. We call this link rot, and whoever needed to download the PDF reference might have experienced this problem first-hand because the PDF reference has changed address many times. There are centralized services such as doi.org that provide a Digital Object Identifier (DOI) that allows you to retrieve one or more currently active URLs to a resource. DOI allows you to register and maintain the location of a document for a fee. Wouldn't it be great if we could reduce that fee, and decentralize the service?

We can solve all of these problems, and probably even some problems we didn't even know we had, by introducing blockchain into the world of PDF documents.

# Advantages of using Blockchain for Signing PDF documents

Suppose that we don't store the signature of a document in the document itself, but instead store it in the blockchain. In a blockchain system that relies on PKI, users already have a private key, which already lowers the threshold for adoption. Users of the system could be human beings, companies, or systems used within companies (such as a CRM, an ERP, a DMS,…). In the context of blockchain, the keypair is usually generated in a Web of Trust (WoT), but that's not a must. In some use cases (e.g. in the context of a permissionless blockchain), we might prefer to work with a keypair generated by a CA. As for the private key, some use cases might require that it's stored on a hardware device, such as an HSM, a smart card, or a USB token.

Signing a document would involve creating a hash of the document, signing that hash with a private key, and store that signed hash in the blockchain along with the corresponding public certificate. In this case, we use blockchain as a database for storing signatures with the unique ID pair of a document as the primary key for each record. Optionally, we can add metadata to the signature records, such as the status of the document (draft, final, unpaid, paid,…) and one or more locations (e.g. the URLs of different mirrors from which the document can be downloaded).

Verifying a signature can be done by retrieving all the signature records from the blockchain, based on the ID pair of the document you have at hand. You get a historical overview when each signature was applied. You can verify all of those signatures, or only a selection, for instance only the most recent signatures of every signer, or, if some signatures are expired, verify those records that confirm that the integrity of the document is still preserved.

## How Blockchain can make you love the things you hate about PAdES

Storing the signature in the blockchain instead of the document itself has many advantages:

- We still meet the requirement of integrity, since we create the signature in exactly the same way. We only store the signature in a different place;
- We can still work with a CA to meet the requirements of authenticity and non-repudiation, but we don't have to. We can also rely on a Web-of-Trust stored in the blockchain;
- We don't need to work with a TSA anymore; timestamping is inherent to blockchain; once a record is added to a block, its contents are immutable and can't be changed;

- Since the signatures are stored outside of the document, signatures can be applied in parallel without a predefined order. Every signer can add a signature to the blockchain at any moment in time, and
- Everyone can check who signed a document, when, and in which order; this can be done without having to rely on a central service (e.g. Docusign);
- Keeping the signature of a document alive doesn't require any other approach; any eligible signer with access to the document can create a new signature in the blockchain using the latest hashing and encryption algorithms.

When discussing different use cases, you'll notice that we'll often talk about "registering a document" instead of "signing a document." In some sectors, people won't accept the fact that we store a signed hash in the blockchain as a valid signature, e.g. because the government doesn't accept such a signature as legally binding. Even in those cases, it makes sense to store hashes of documents along with some metadata in the blockchain.

- You can retrieve the records of a document based on its ID pair to inspect the metadata added by the person who signed the document. For instance: a document that is registered as an invoice with status "unpaid" by the CRM system of a vendor, can be registered as an invoice with status "accepted" by the ERP system of a buyer, and eventually be registered as an invoice with status "paid" by the buyer's bank.
- You could also do a search on the first part of the ID pair to find records of documents that are related. A document that starts its lifecycle as a quote request, could change into a quote, then be accepted as a purchase order, then fork into an invoice and a delivery note.
- If you find a document in the wild, e.g. a draft of a technical specification, and you want to know if that document is the latest version, you could search for a record that registers a more recent version of that draft, and use the location information stored in that record to retrieve that updated version. Obviously, there are also some disadvantages.

# When not to use this approach?

Some governments might not accept signatures stored in the blockchain as legally binding. It might take time before this signature functionality is widely accepted in some sectors. We'll also have to make a distinction between *registering* a document (not legally binding) and *signing* a document (legally binding). In Belgium, this is typically solved by providing each citizen with two public/private key pairs: one for authentication and one for non-repudiation. Documents that are *"digitally signed"* (mind the use of quotes) with the private key for authentication aren't legally binding. They are *"signed"* in technical terms, but not in legal terms. Only documents that are signed with the private key for non-repudiation are legally binding.

The other big disadvantage is that you'll need a blockchain client to register sign a document. Today, the majority of CRM's, ERP's, DMS's don't provide PAdES functionality out-of-the-box. If we want

our approach to be successful, every system that involves documents should also be a blockchain client.

Finally, this approach has all the same disadvantages as any other system based on DLT. A record with information about a single document can be kept really small, but we might be talking about quadrillions of documents that will eventually be stored per blockchain, especially if we want to keep track of the history of a document, and extend the life of a signed document. Fortunately, these problems can and will be addressed as blockchain adoption grows.

# When Contracts aren’t Smart Contracts

We’re constantly talking about documents, and some of these documents might be agreements, or “contracts.” The word “contract” is also used in the context of DLT, more specifically when we talk about “smart contracts.” We need to make sure that we don’t confuse one with the other.

The concept of smart contracts was introduced by Nick Szabo in the nineties. Originally, the concept wasn’t related to DLT in any way. Today, there’s no longer a uniform understanding of the term “smart contract.” Some consider a smart contract as computer code that is executed on a distributed ledger â€“e.g. in the case of Ethereum, the code runs on the Ethereum Virtual Machines (EVMs)â€”, others consider it to be a legally binding contract that is coded in a programming language â€“e.g. in the case of Corda, contracts can be executed on a Java Virtual Machine (JVM).

In the context of registering documents in the blockchain, smart contracts can be used to determine whether and how a document can be registered in the blockchain, at which time they were registered, and possibly at what cost, but that’s outside the scope of this ref card. We’ll focus on storing with the hash of a document in a record, (optionally) along with some metadata. This document may or may not contain a smart contract.

This is a completely different use of DLT than the way blockchain is used in the context of Bitcoin, Ethereum, or other applications that use DLT as a kind of large computer that is distributed over the whole world. We’ll use DLT more in the sense of a distributed database, where the id of a document is on one hand the primary key, and on the other hand one of the main triggers for events to be performed by an external service. such as a CRM, ERP, or DMS system.

There are advantages of storing “the contract” in a document outside the blockchain. The contract itself isn’t distributed, and can therefore remain a secret. Also, storing complete documents in the blockchain would cause huge problems in terms of storage and bandwidth. The disadvantage of storing “the contract” in a document outside the blockchain, is that the contract isn’t distributed, and that an additional system has to be put in place to distribute the document, or to give access to it. We can counter that problem by storing one of more links to the document in the ledger record.

If the contract is stored in the blockchain as a smart contract, there is agreement on how the contract needs to be interpreted and executed. This isn’t the case in the approach we have chosen:

- The contract could be a traditional agreement, in which case a mutual understanding on how to interpret the contract is needed between any two parties that are involved. Most likely, these contracts are written by human beings, and there is often room for interpretation. This may be a desired effect, as not all contracts can be programmed. Think of an employment or a contractor agreement where the job to be done or the work at hand can’t be described up-front in an ironclad way, or

- We could work with human-readable documents that also contain machine-readable information allowing automated processing of the document. The ZUGFeRD standard, for example, offers a way to create invoices that can either be read by human beings in the form of an PDF/A-3 document, or processed by machines because of the XML attachment that complies with Cross Industry Invoice (CII) standard, or
- If the document does consist of code, and code only, all parties involved should be in agreement on the way the code is executed. We may chose this option in cases where we don't want to expose the code to the other clients of the blockchain we are using. This example is out of scope in the context of this ref card as none of the use cases we'll cover requires the document to consist of code.

The mechanisms outlined in our patents don't primarily rely on smart contracts stored in the blockchain. This was a deliberate choice to cater for use cases where the contract needs to remain private in the context of an environment with parties that can't always be trusted.

# Use cases

## Invoices

In our book ZUGFeRD: the Future of Invoicing, we explain how a combination of PDF/A-3 with an XML file that follows the Cross-Industry Invoice (CII) standard, makes a great solution for automating the payment process of invoices. Unfortunately, the ZUGFeRD standard doesn't provide a solution to avoid fraud. Just like a paper invoice, any digital invoice can be intercepted and altered. Some criminal organizations specialize in changing the wiring information on invoices, so that people receiving an invoice pay the invoiced amount to the wrong party. Digital signatures on invoices can protect customers or companies against this type of fraud, but adoption of digital signature technology in the context of invoices is slow because the technology turned out to be too complex.

It would be great if standard PDF viewer would have a button, so that when someone opens a PDF and clicks that button, some information about that invoice is shared, e.g. if the invoice was altered, and who issued the invoice when. That information could be retrieved from the blockchain, and the identity of the vendor could be checked in a web of trust, including for instance whether or not the vendor has fulfilled his tax obligations. See also an article published in De Tijd, a Belgian financial newspaper.

Another real-world use case consist of a bank that grants a loan for buying a car. In this case, the bank requests an invoice. More often than not, a paper invoice is used in this context, and that invoice can be forged to get a higher loan. Wouldn't it make more sense if someone applying for a loan just gave the bank the unique ID of the invoice, after which the bank can look up a URL of that invoice in the blockchain, so that the original invoice can be downloaded from a secure location?

Registering invoices in the blockchain also has added benefits for governments, in the sense that they can scan the blockchain for invoices for tax purposes. If customers only pay invoices that are registered in the blockchain, companies will no longer be able to "forget" to declare some invoices. It will also be impossible to create fake invoices in the hope to get a VAT reduction or to artificially lower the profits of the company. The fact that a hash of each invoice is stored in the blockchain, the government will also be able to sample batches of invoices to check whether or not the documents were tampered with.

## Supply chain

Invoices are merely a first step, we could use the principle for every document in the supply chain. A document that starts off as a quote request can spawn different quotes being issued by different

vendors. After examining the different quotes, one quote could be chosen and result in a quote request. That quote request can then fork into an invoice and a delivery note. A bank that processes the invoice can mark the invoice as paid.

A simplified example of this process is shown in figure 1.

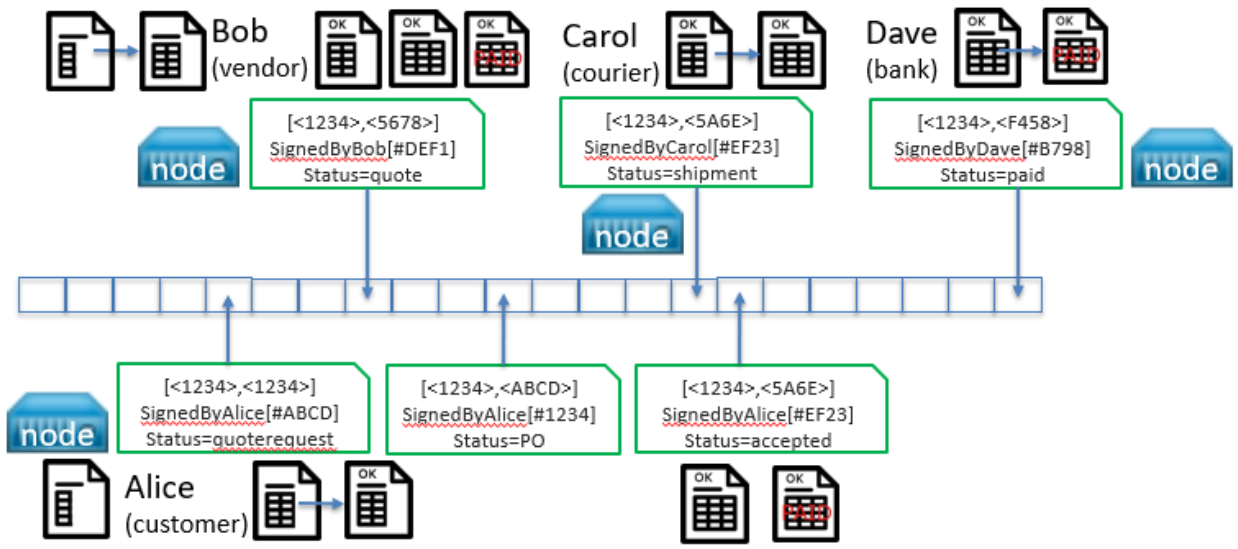


**Figure 1: Document Workflow**

The different nodes in these figures represent CRM and ERP systems at different companies. Each time an invoice leaves the CRM at the vendor, that invoice could be registered automatically in the blockchain. When this invoice is received in the ERP system on the buyer's side, it can be validated in the blockchain before it is sent to the bank to process the payment. Once the bank marks an invoice as paid, the CRM of the vendor will notice this immediately when the block containing a record created by the bank reaches the CRM's blockchain client.

It's easy to imagine how this system can be used to automate all kinds of workflows.

# Port of Antwerp

The port of Antwerp, largest port of Belgium and second largest port of Europe in terms of container capacity, is experimenting with blockchain technology for the automation of logistics. This way, the port aims to speed up the many interactions between clients and stakeholders. They also aim to reduce the cost: not only the cost of the enormous paper trail, but also price fluctuations not being communicated clearly, a dollar to euro conversion not having been fixed, or a bill of lading mismatch.

Transporting even a single container includes a minimum of 30 different parties, going from the truck drivers that delivered the goods that went into the container, to the people loading the container, the people taking the goods out of the container, as well as the harbormaster. Communication between all these parties occurs over a wide variety of platforms and media: the most common ones are e-

mail, fax, and phone. It's no surprise that information is leaked, gets lost, or is distorted along the way.

Security is the largest issue. In order to ensure containers can only be moved and opened by the correct people, they are often locked with a code. The code has to pass through all the communication-channels mentioned above. Most of these are insecure, and containers often go missing, their content stolen. And because the container isn't continuously tracked, nobody can be blamed for its disappearance.

This is where blockchain provides the perfect fit. A single source of truth, e.g. the bill of lading, can be passed and signed (as a token) between all parties. This offers stakeholders a way of tracking the container in real time. It also enables stakeholders to see where the container went missing, and under whose responsibility.

## Avoid Link rot

Suppose that you have a huge repository of documents that you want to share publicly. To do this, you use third party Cloud storage from vendors such as Akamai, Box, Dropbox, Amazon, Google,… However, due to the nature of your business, these documents can move around from one place to another. Either after an upgrade to a different Content Management System (CMS), or because you want to switch from one vendor to another. Your repository is very much alive, and this results in your customers getting a "404 - File Not Found" error. This phenomenon is also known as link rot. The document is available somewhere, but the original link no longer works.

Often you see that a CMS solves this problem by creating a redirect, but when a file moves too frequently, you risk getting a Too Many Redirects error, or worse: an infinite redirect loop. All of this can be avoided by registering every document in the blockchain, and have your CMS add a new record every time the location of the document changes. Then, instead of sharing the document or the document's location, you merely have to share the document's ID. Whoever wants to consult the document, can get the location by searching the blockchain for the latest location record on the blockchain. Given the right credentials –in case the URL can't be accessed publicly–, the corresponding document can be downloaded. Since the record also contains a signed hash of the file, whoever consumes the document can also check if the file is authentic and hasn't been tampered with. This also implies that it's not mandatory to centralize all of your documents on one and the same storage system. Different cloud storage services can be used with blockchain routing document consumers to the appropriate service.

## Asset management

When setting up a blockchain for documents, it will be important to make a clear distinction between the content that needs to remain secret and the information that can be public. Keep in mind that all the information that is stored in the blockchain can eventually be exposed publicly, even if it's encrypted.

Imagine that we'd use the blockchain to register certificates of real estate property, works of art, diamonds, and other valuable assets. An owner might not want the specific description of all of his assets to be displayed publicly, yet he might want to be able to prove that he is indeed the owner of an assets that corresponds with the description on a certificate. This can be achieved by having the certificate registered in the blockchain e.g. signed by the previous owner selling the asset, along with metadata about the new owner.

This also implies that given an ID of a certificate, you'll be able to go back in time and discover every previous owner of the asset. However, without access to the certificate, you'll never know the specifics about the asset, except for the information stored as extra metadata. You'll have to choose that metadata wisely. For instance, if the assets consist of real-estate, do you really want a public ledger listing the address of each property, along with a historic overview of every owner?

## Last will and testament

Some documents need to be verifiable for a long period in time. For instance: if you write your last will and testament at the age of 40, you hope that no one will have to consult that document for another 40 years in the future. Unfortunately, the document will have to surface one day, and on that day, you want your heirs to give the assurance that the document they are looking at is genuine. They won't have that assurance if you created a hash of the document in the year 2000 using SHA-1.

One way to ensure the Long-Term Validation of your last will and testament could be to deposit it with a trusted third party that registers an original of your last will and testament on a public blockchain with centralized ledger control on a regular basis, always using the latest hashing algorithms.

## Sensitive information

Journalists were able to write about tax evasion because they had access to a massive amount of documents. They discovered these documents in an indirect way. Instead of investigating the people and the companies mentioned in the Panama or Paradise Papers, they targeted the consultants working on behalf of the alleged tax evaders. Thanks to whistleblowers and data breaches at these consultants, they even discovered targets they previously didn't even know of.

Obviously, this isn't good news for the people and companies involved, nor for their consultants. All of this could have been avoided if only the consultants wouldn't have kept an archive consisting of documents with sensitive information. Consultants will probably argue that they needed to keep a trail of historical documents to ensure the continuity of their service. True as that may be, they could also have put the burden of archiving documents on their customers, making the consultant less vulnerable for exposing sensitive data. Consultants will argue that they can't always trust the customer to provide original, authentic, genuine documents, but that problem can easily be solved. A consultant could easily create a private blockchain with centralized ledger control to store hashes

of the documents, instead of the actual documents. The moment a consultant –such as a law firm, a fiscal advisor– is no longer working on a specific case, all documents related to that case can be registered in the private blockchain, transferred to the customer, and physically removed from the consultant's own archive.

To avoid "expiration", one could think of a subscription, where the customer returns to the consultant on a regular basis –e.g. every three years– giving temporary access to the customer's private archive of documents. The consultant can then verify if that set of documents is complete, authentic, untampered with, and register the set of documents anew using the latest hashing algorithms. Once the verification and renewal of the registration is done, the consultant can once more physically delete the documents, so that, when there's a breach at the consultant –e.g. by a hacker or a thief–, that intruder can only access the documents the consultant is currently working on. He'll only find a hash of all the other, historical documents. If an intruder wants access to all the documents, he has to break into the system of every single customer.

This use case is also important in the context of GDPR. GDPR is a European regulation that –among others– gives European Citizens the right to have their personal data removed from the systems of a company in specific situations. In the case such a company only stores a hash of the personal data, and not the actual data, there is no personal data to remove.

## There's more

These are only a handful of examples. Once you start looking at existing document problems, you'll easily find many other use cases of registering documents in the blockchain. Feel free to share those ideas with iText; we can provide the technology that will help you getting started.