EDTA, iText and INBATEK Conference

Bangkok, July 27, 2017

Digital Signatures in PDF

- Basic concepts…
- … applied to PDF
- Architectures: server-side vs. client-side
- Digital signatures and document workflow
- Long term validation
- How Blockchain made me love everything I hate about digital signatures

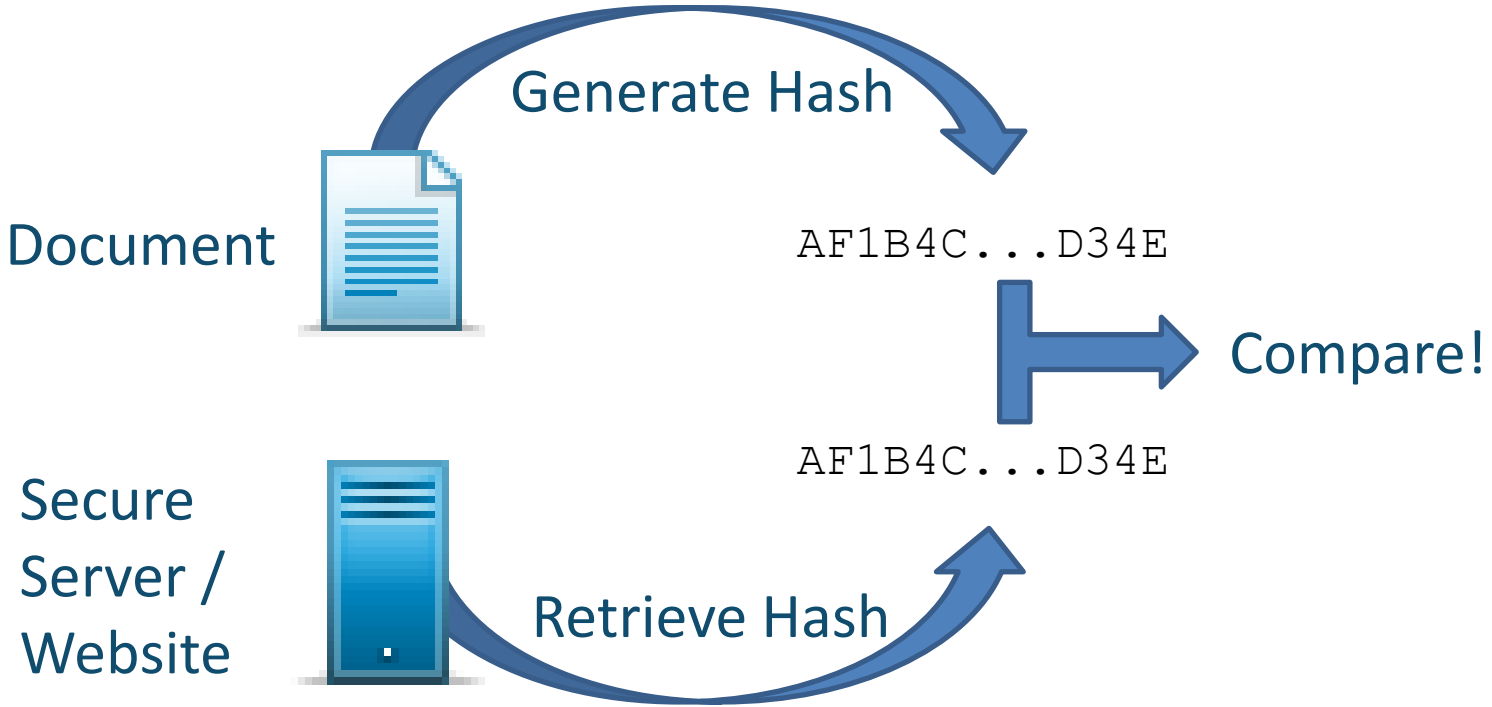
Basic Concepts…

- Hashing algorithms
- Encryption algorithms
- Certificate Authorities
- Digital signatures

Three goals

- Integrity — we want assurance that the document hasn't been changed somewhere in the workflow.
- Authenticity — we want assurance that the author of the document is who we think it is (and not somebody else).
- Non-repudiation — we want assurance that the author can't deny his authorship.

Concept 1: Integrity check using hash



Concept 1: Hashing

- **Hashing algorithm**
  - a cryptographic hash function to turn an arbitrary block of data into a fixed-size bit string.
- **Available algorithms**
  - **MD5:** Ron Rivest (broken)
  - **SHA:**
    - SHA-1: NSA (broken! See https://shattered.io/)
    - SHA-2: NSA / NIST
    - SHA-3: Keccak (made in Belgium!)
  - **RIPEMD:** KULeuven

Concept 2: Encryption

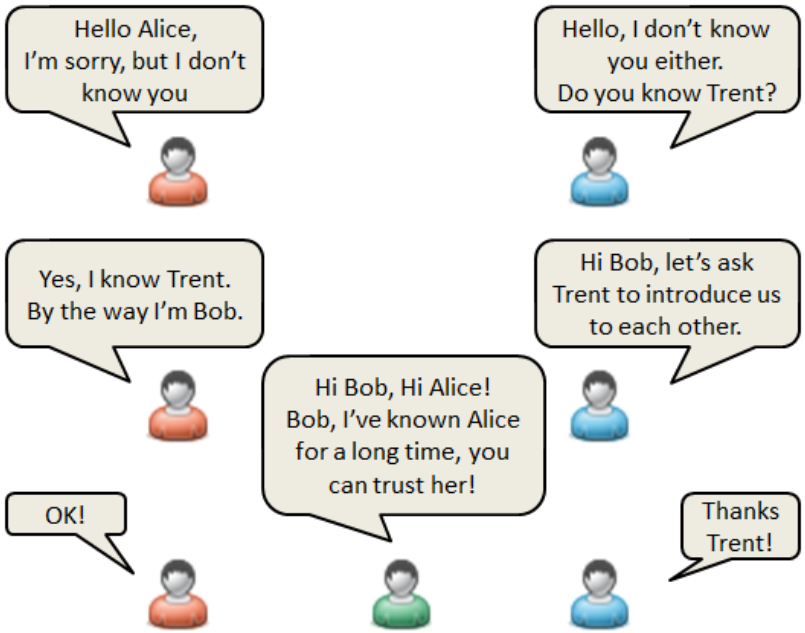
- Asymmetric key algorithms
  - Encryption

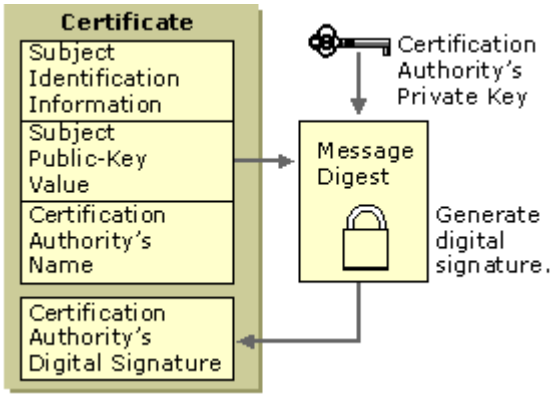
  - Digital signing

Concept 2: Some name dropping

- Public Key Cryptography Standards
  - PKCS#1: RSA Cryptography Standard (Rivest, Shamir, Adleman)
  - PKCS#7: Cryptographic Message Standard (CMS)
  - PKCS#11: Cryptographic Token Interface
  - PKCS#12: Personal Information Exchange Syntax Standard
  - PKCS#13: Elliptic Curve Cryptography Standard (ECDSA)
- Federal Information Processing Standards (FIPS)
  - DSA: Digital Signature Algorithm (DSA)
- European Telecommunications Standards Institute (ETSI)
  - CMS Advanced Electronic Signatures (CAdES)

Concept 3: Certificate Authorities



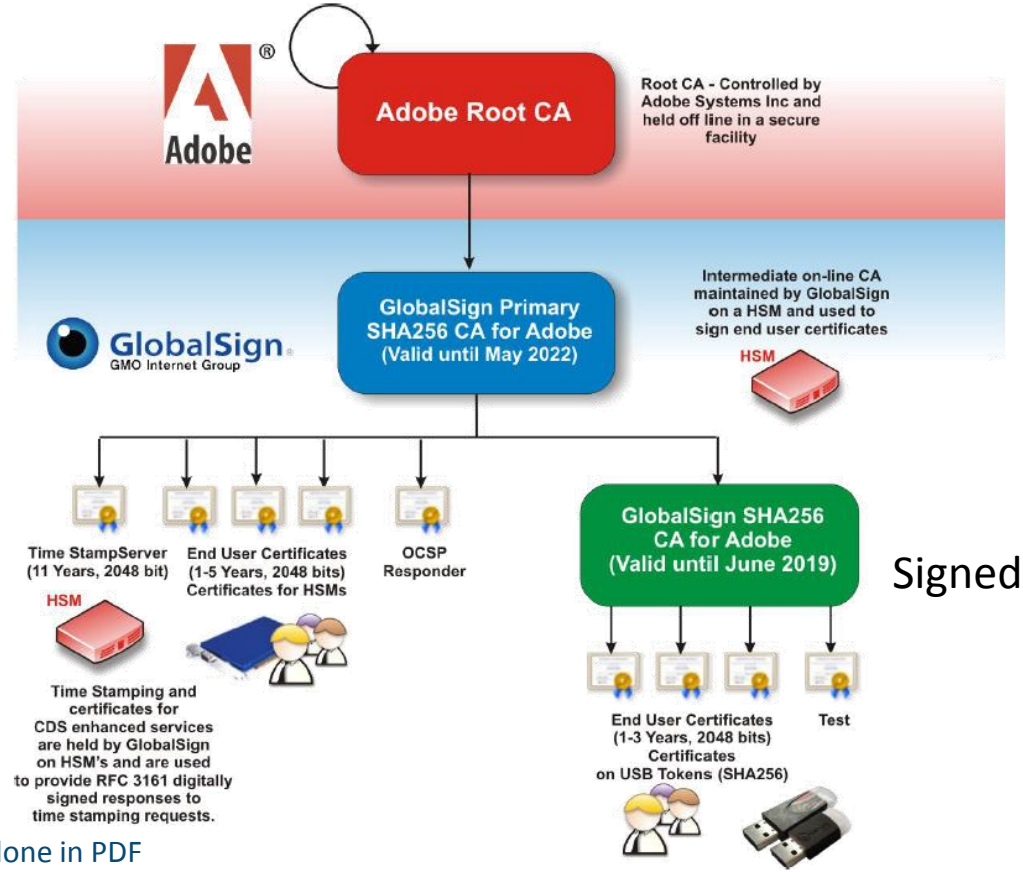


Concept 3: example

Self-signed:

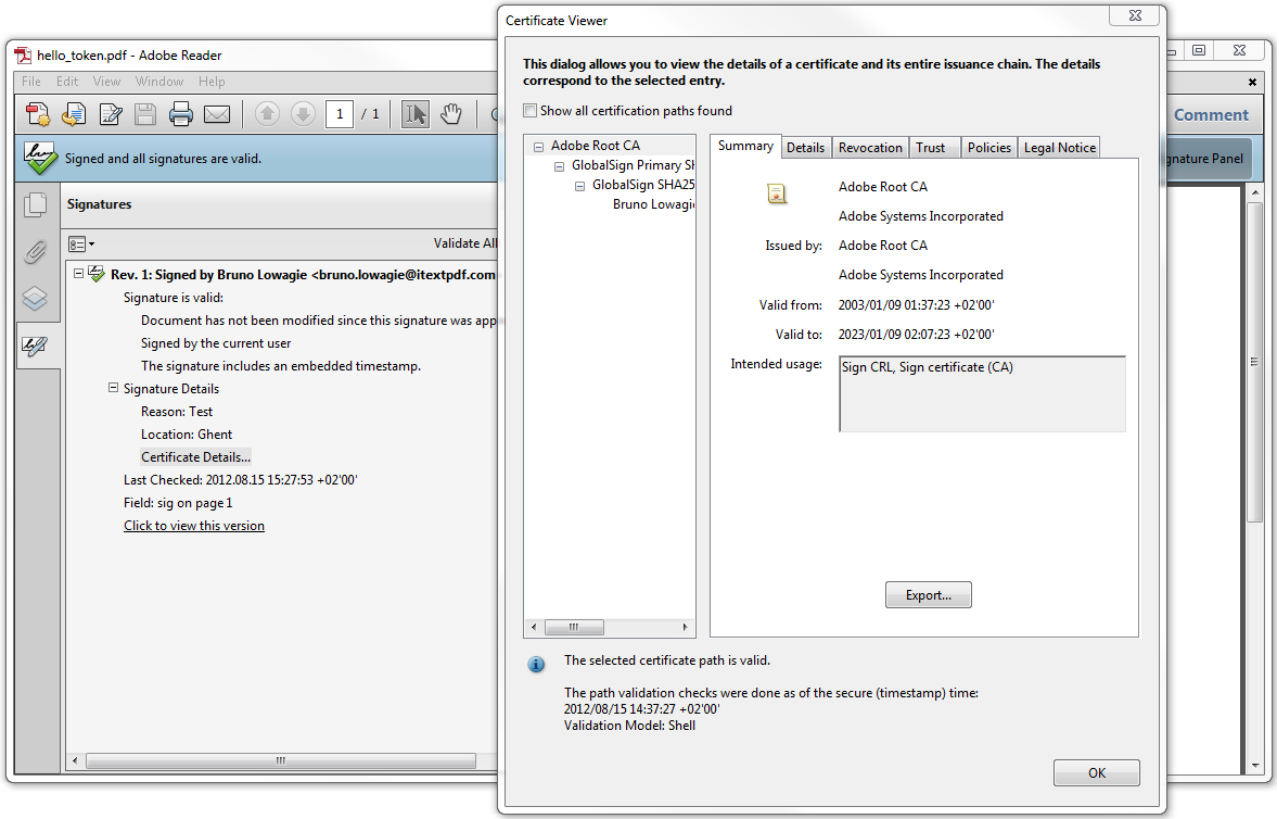
Signed by Adobe:

Signed by GlobalSign:



Signed by GlobalSign

Concept 3: example



Concept 3: the green check mark

- PKCS#12: Personal Information Exchange Syntax Standard
  - public and private key are stored in a file
- PKCS#11: Cryptographic Token Interface
  - public and private key are stored on a device
  - In the context of PDF and the “green check mark”:
    - Certified Document Services (CDS): Adobe’s root certificate
    - Adobe Approved Trust List (AATL): Trusted root certificates (since Acrobat 9)

Concept 1 + Concept 2 + Concept 3

- **Producer**
  - Provides data as-is: [A]
  - Provides hash of data, encrypted using private key: [B]
  - Provides public key
- **Consumer**
  - Creates hash from data [A]: hash1
  - Decrypts hash [B] using public key: hash2
  - If (hash1 == hash2) document OK!

Goals met?

- **Integrity**
  - Hashes are identical
- **Authenticity**
  - Identity is stored in public key signed by CA
  - A time-stamp can be added
- **Non-repudiation**
  - If hash can be decrypted with public key, the document was signed with the corresponding private key

Differences between EU and US

- In the US, we make a distinction
  - Electronic signatures don't necessarily involve PKI
  - Digital signatures when a PKI infrastructure is involved
- In Europe, we speak of electronic signatures
  - As a **synonym** for digital signatures
  - All laws and regulations take this wording
  - There's no sharp distinction between electronic and digital signatures (which leads to confusion)
- I always speak of digital signatures

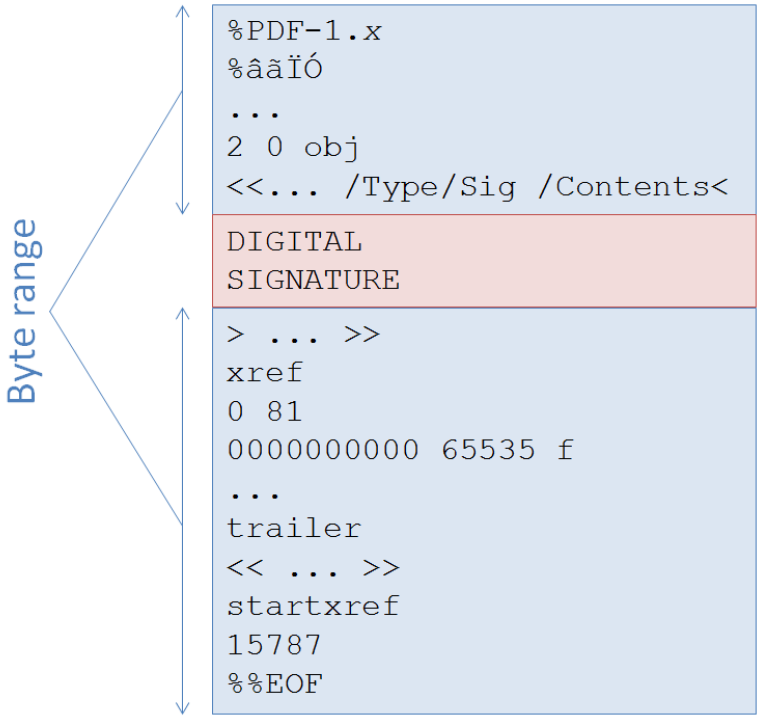
… Applied to PDF

- ISO 32000-1
- ETSI TS 102 778 (PAdES)
- ISO 32000-2

Standards

- ISO
  - ISO-32000-1 (2008) based on PDF 1.7 (2006)
  - ISO-32000-2 defines PDF 2.0 (2017)
- ETSI: TS 102 778 (2009 - 2010)
  - PAdES 1: Overview
  - PAdES 2: Basic – CMS based (ISO-32000-1)
  - PAdES 3: Enhanced – CAdES based (ISO-32000-2)
  - PAdES 4: LTV – Long Term Validation
  - PAdES 5: XAdES based (XML content)
  - PAdES 6: Visual representation guidelines
- ETSI: TS 103 172 (2011 - 2013)
  - PAdES Baseline Profile

Signatures in PDF



- There are no bytes in the PDF that aren't covered, other than the PDF signature itself. (*)
- The digital signature isn't part of the ByteRange.
- The concept "to initial a document" doesn't exist; you sign the complete document at once, not on a page per page basis. (*)

Signature stored in the document

What's inside a signature?

ISO-32000-2:

At minimum the PKCS#7 object shall include the signer's X.509 signing certificate. This certificate shall be used to verify the signature value in **/Contents**.

Best practices ("should" also have):

- Full certificate chain
- Revocation information (CRL / OCSP)
- Timestamp

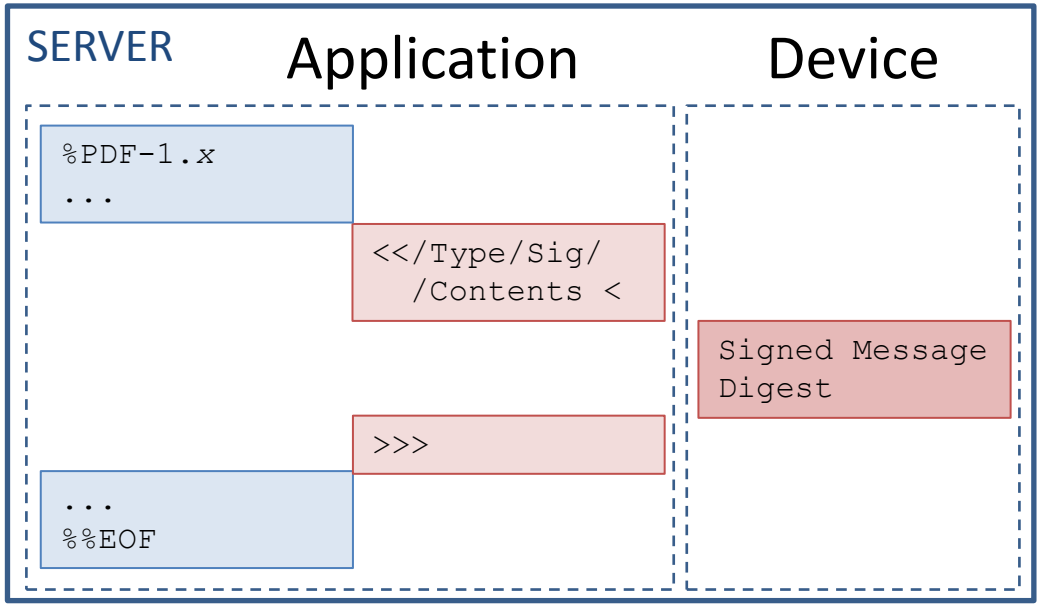
Certificate authority needed
Timestamp authority needed

```
%PDF-1.x
...
/ByteRange ...
/Contents<
DIGITAL SIGNATURE
• Signed Message Digest
• Certificate chain
• Revocation information
• Timestamp
>...
%%EOF
```

Architectures

- Server-side signing
- Client-side signing
- Deferred signing

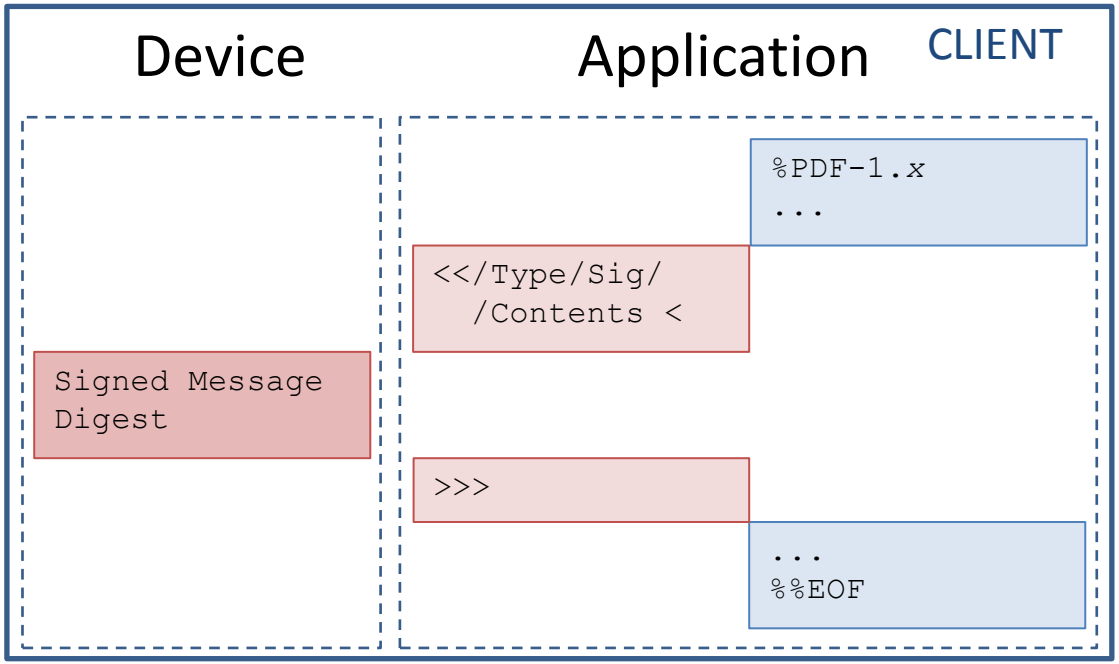
Server-side signing



Use cases server-side signing

- Company signature
  - Invoices
  - Contracts
  - …
- Signing services in the Cloud
  - E.g. Docusign
- Security management responsibilities!

Client-side signing

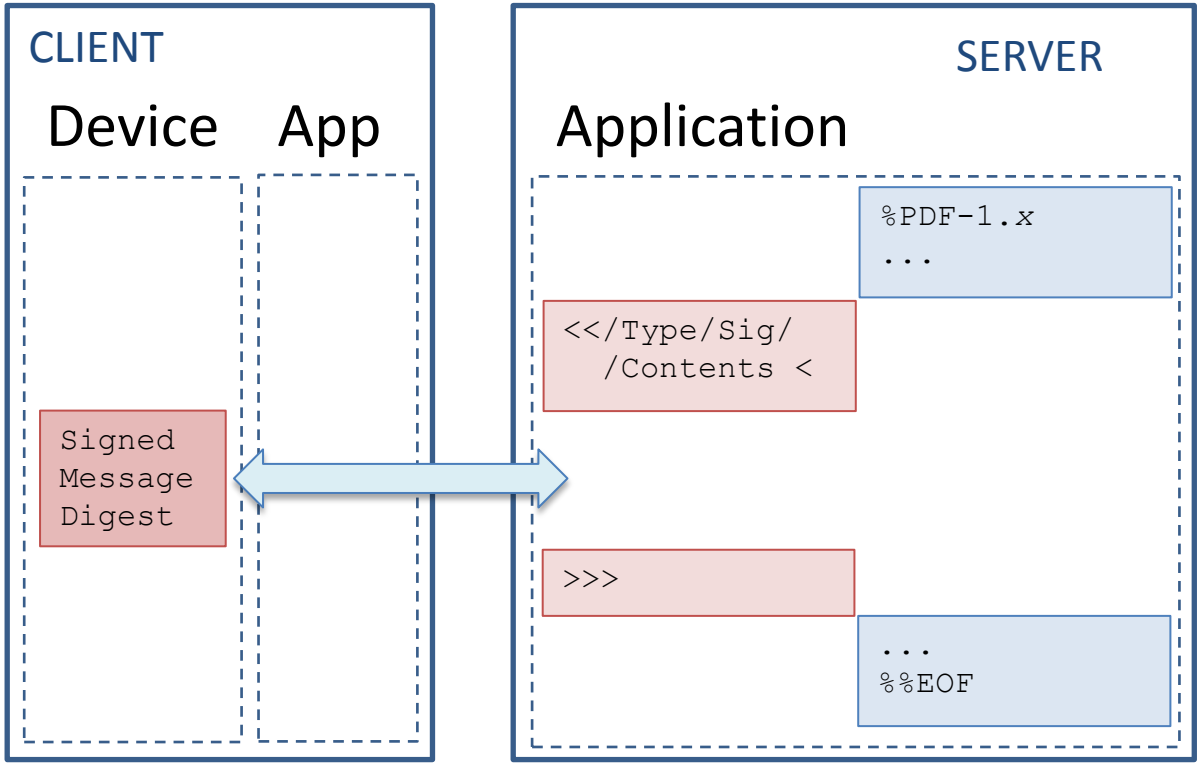


Use cases client-side signing

- Desktop applications
  - Adobe Acrobat / Reader
  - Home-made, e.g. using iText
- In a web context
  - The PDF software runs on the client, e.g. using Java Web Start
- Access to the token or smart card through
  - MSCAPI
  - PKCS#11
  - Custom smart card library

  } 1 signature / second
- Security
  - User has smart card and PIN or USB token and passphrase

Deferred signing



Use cases deferred signing

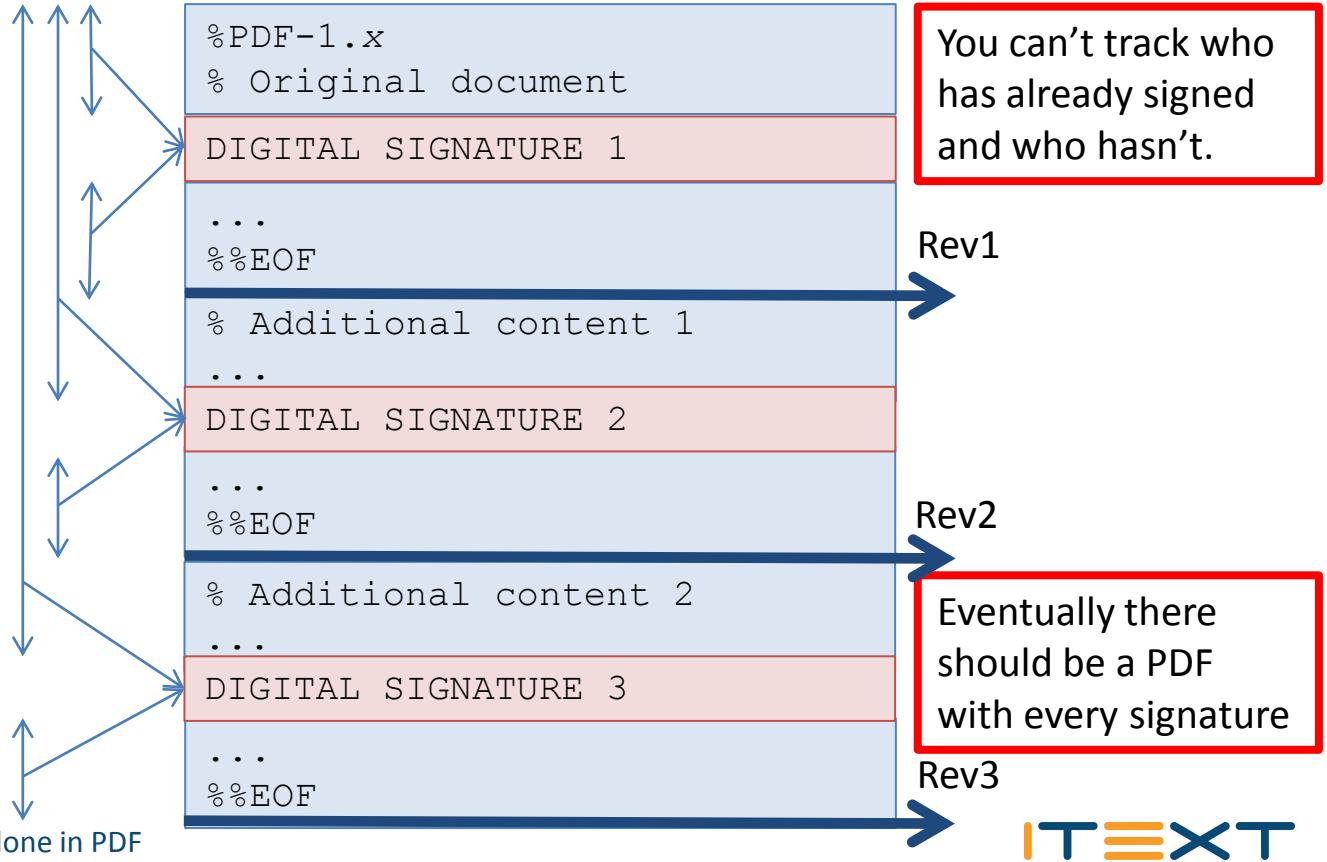
- Signing on an iPad/Tablet
  - App on the device has a low footprint
  - Easy to integrate into a document management system
  - Example: eaZySign (Zetes)
- Disadvantage
  - At most 1 signature / second
  - You need to trust the server that the hash you receive is actually the hash of the document you want to sign.
- ISAE 3000
  - the standard for assurance over non-financial information. ISAE3000 is issued by the International Federation of Accountants (IFAC). The standard consists of guidelines for the ethical behavior, quality management and performance of an ISAE3000 engagement. Generally ISAE3000 is applied for audits of internal control, sustainability and compliance with laws and regulations.

Digital signatures and workflow

- Author signatures
- Recipient signatures
- Locking fields / documents

Serial signatures

A PDF document can be signed more than once, but parallel signatures aren't supported, only serial signatures: additional signatures sign all previous signatures.

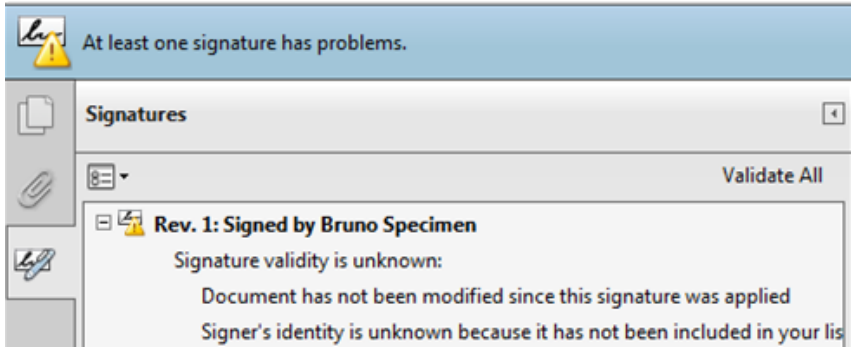


Digital signatures: types

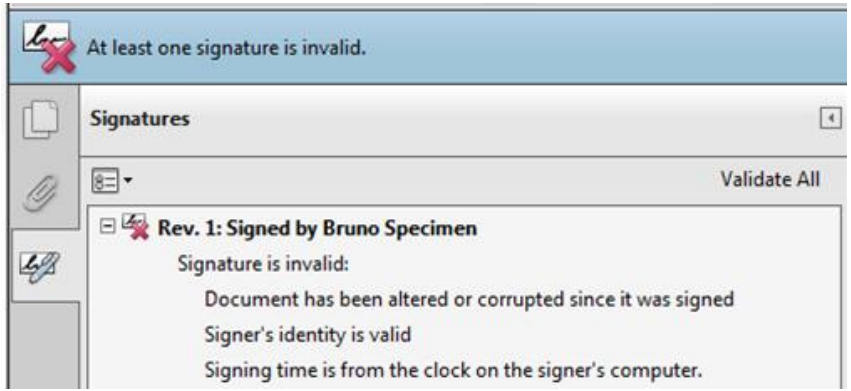
- Certification (aka author) signature
  - Only possible for the first revision
  - Involves modification detection permissions:
    - No changes allowed
    - Form filling and signing allowed
    - Form filling, signing and commenting allowed
- Approval (aka recipient) signature
  - Workflow with subsequent signers
  - New in PDF 2.0: modification detection permissions

Other possible icons

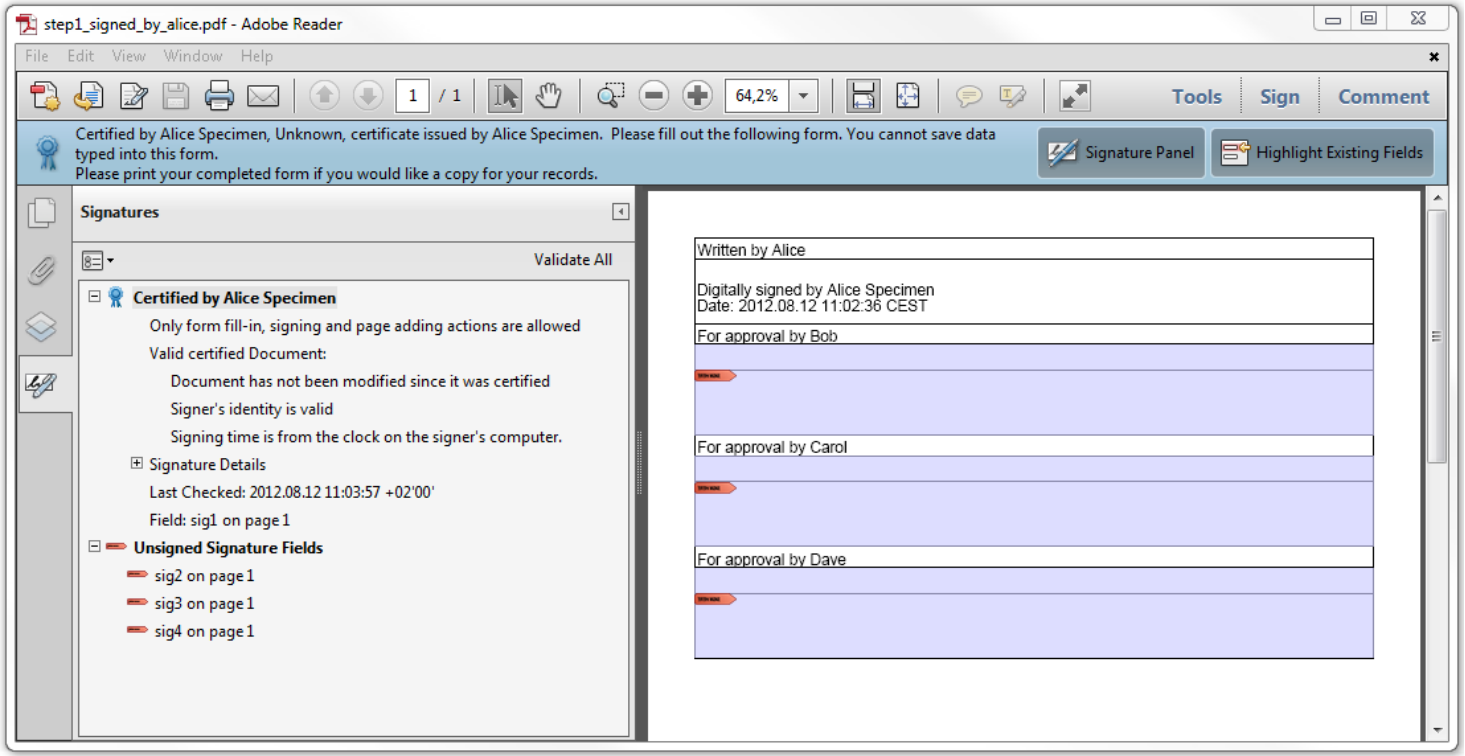
- Signer's identity is unknown



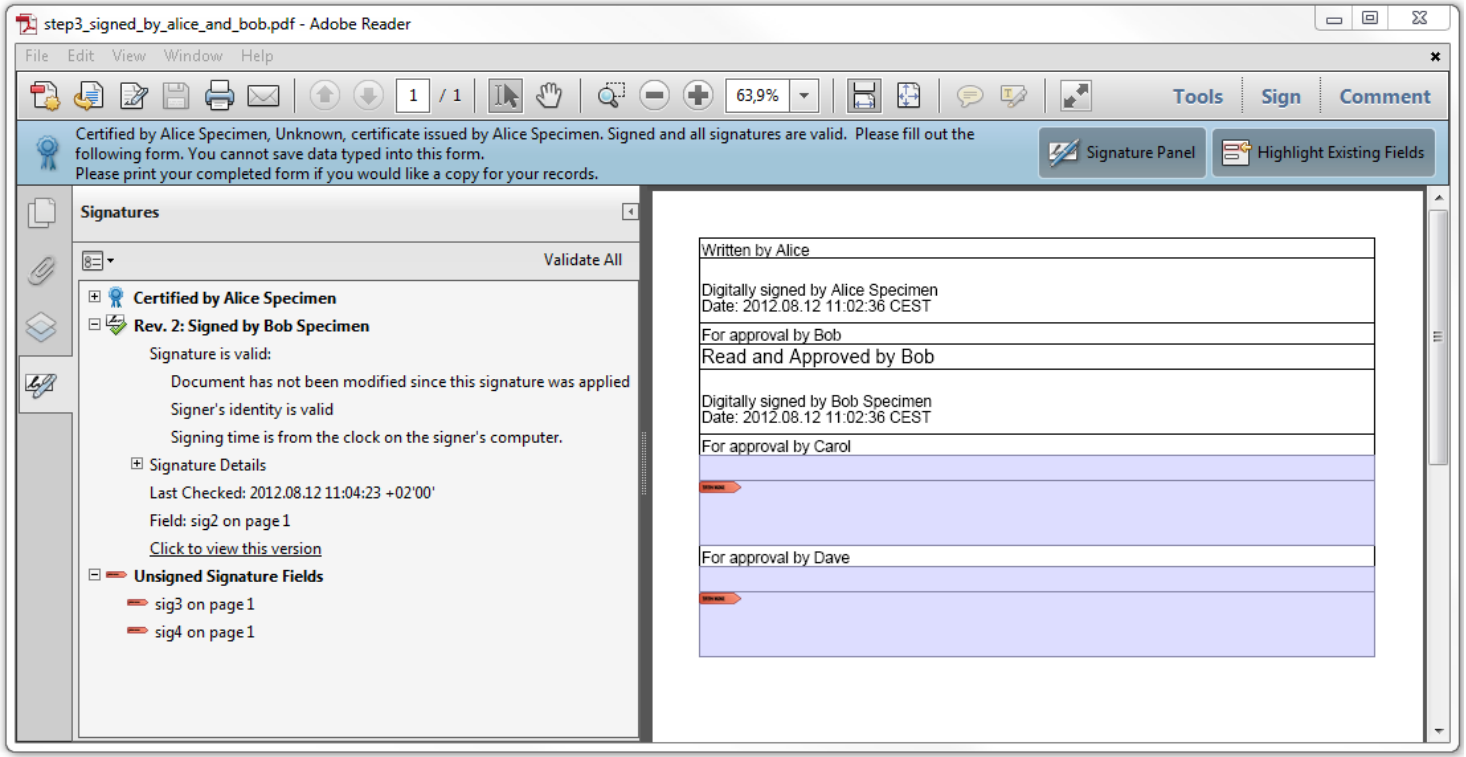
- Document has been altered or corrupted



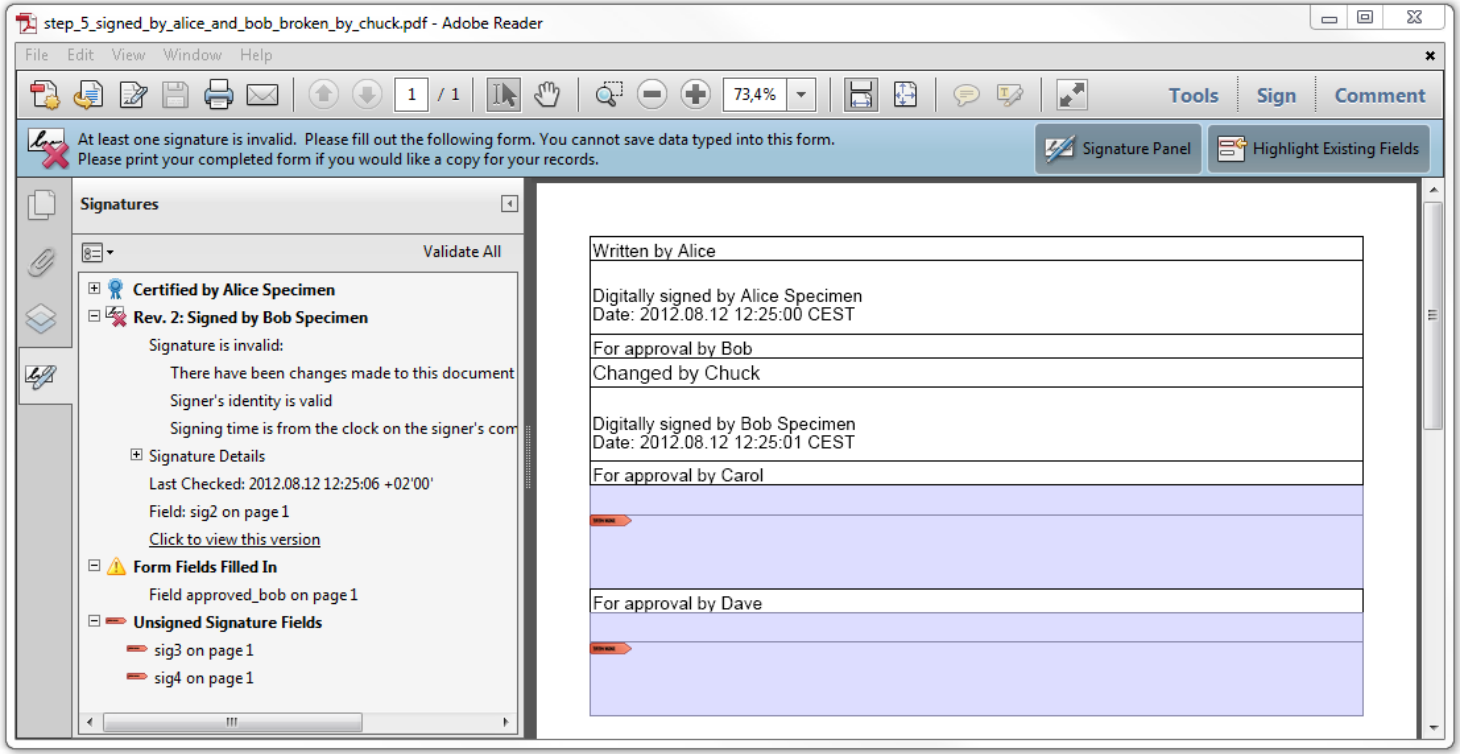
Certified by Alice



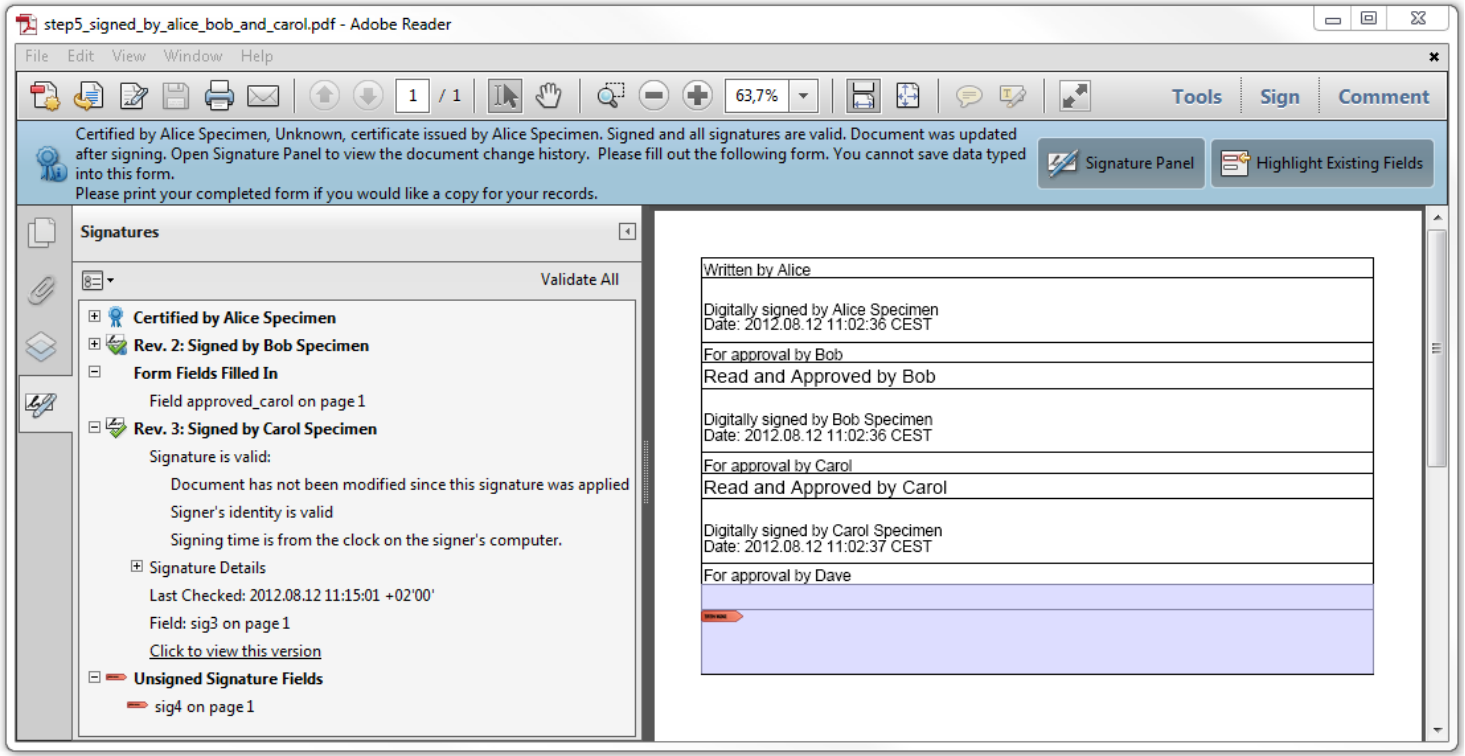
Read, approved and signed by Bob



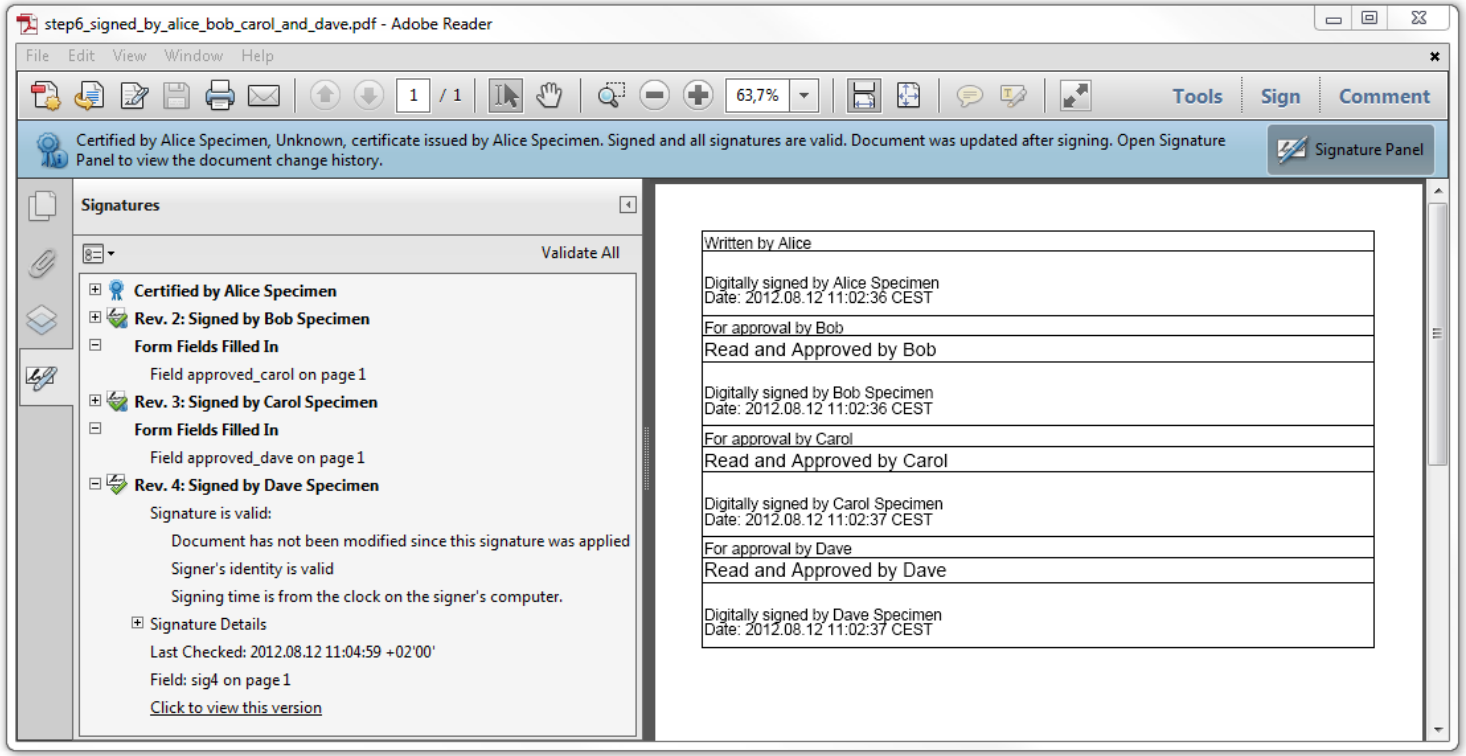
Bob’s signature invalidated by Chuck



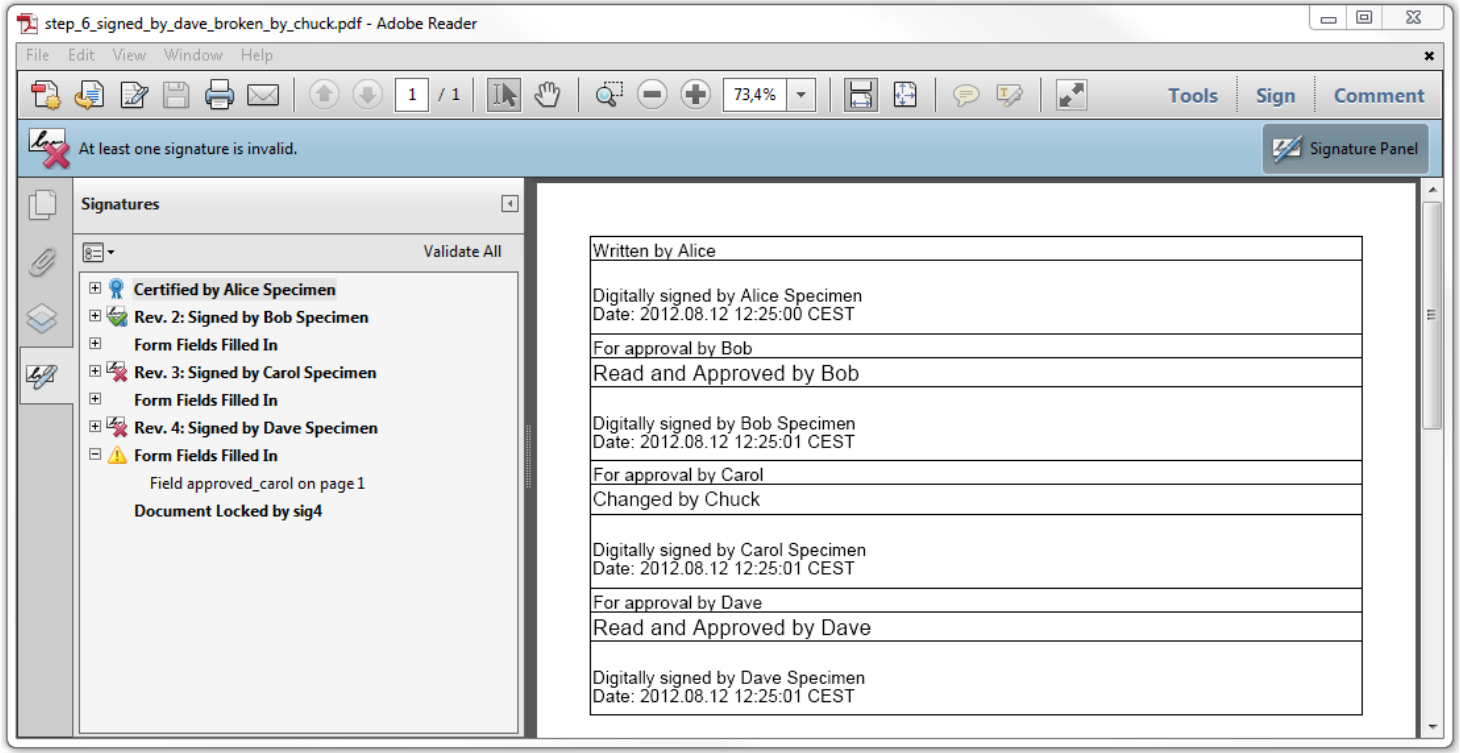
Read, approved and signed by Carol



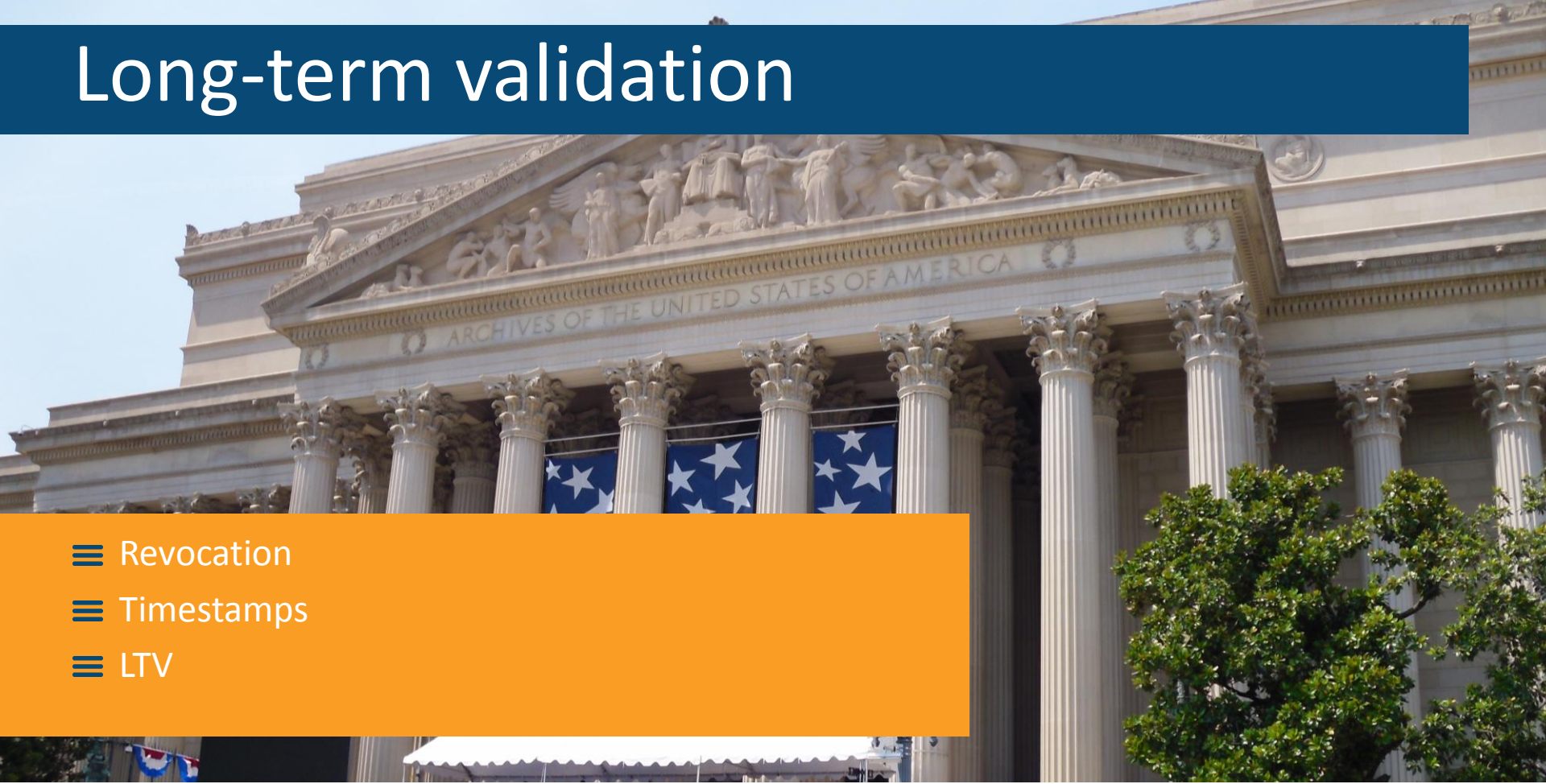
Read, approved and signed by Dave



Signature and lock broken by Chuck

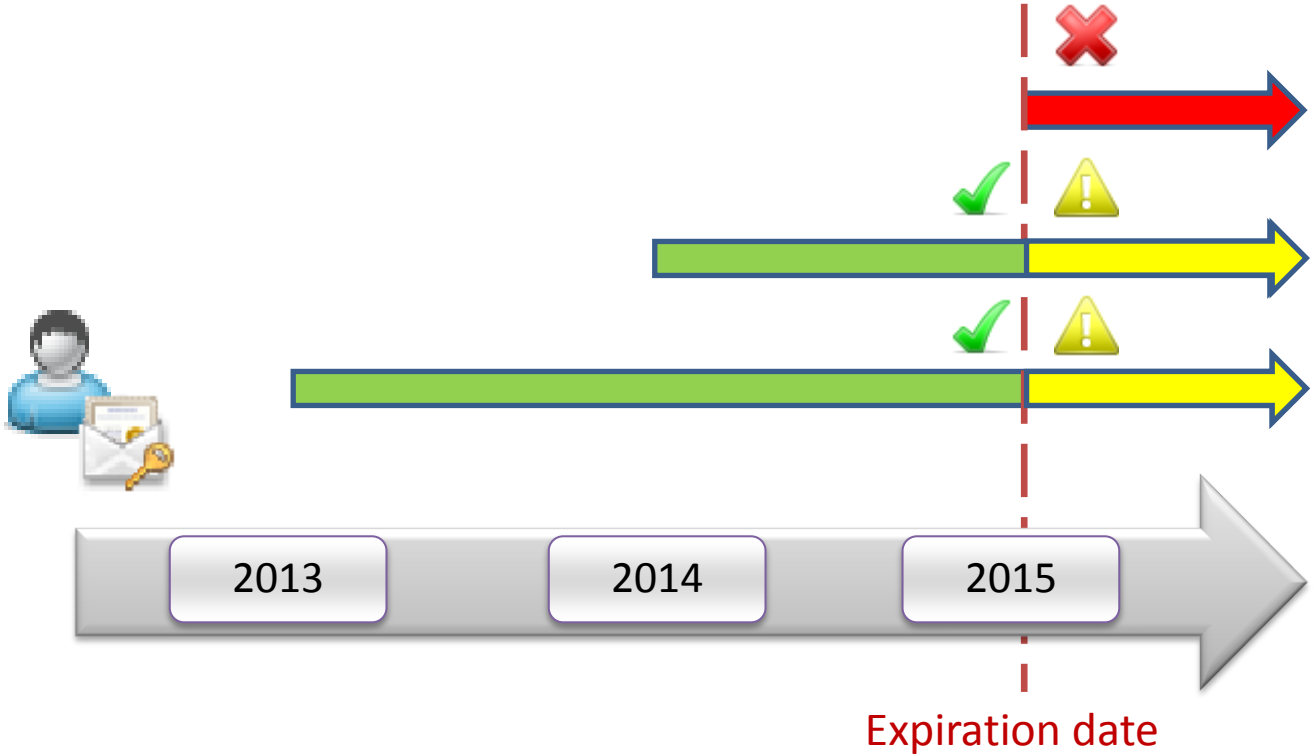


Long-term validation

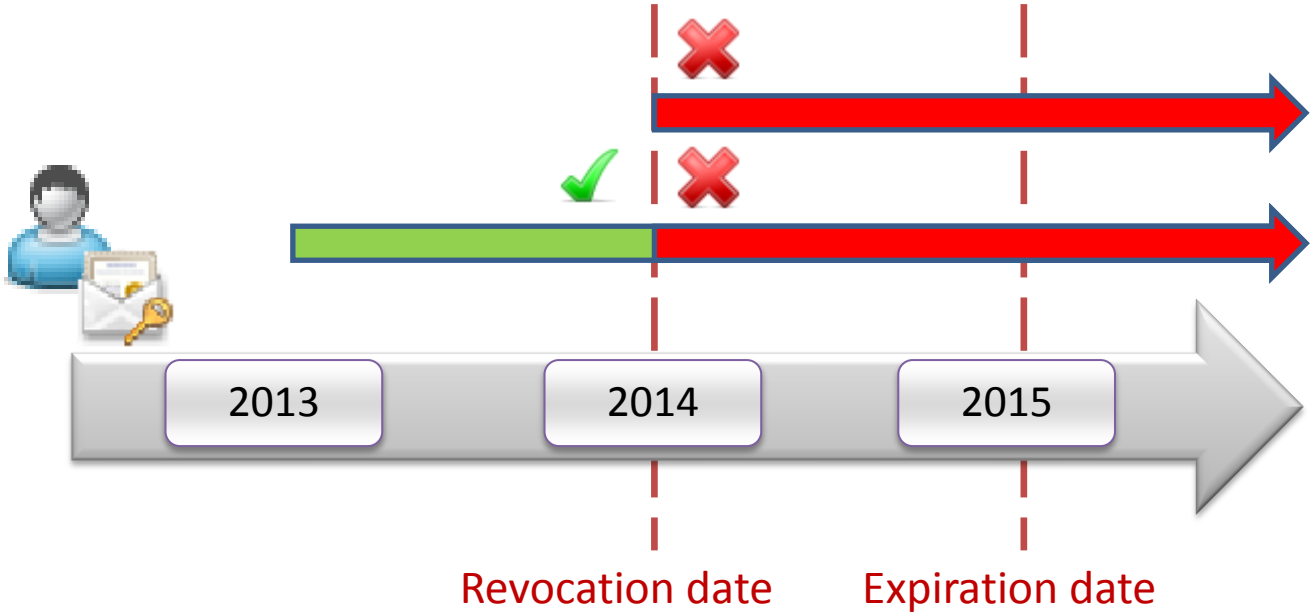


- Revocation
- Timestamps
- LTV

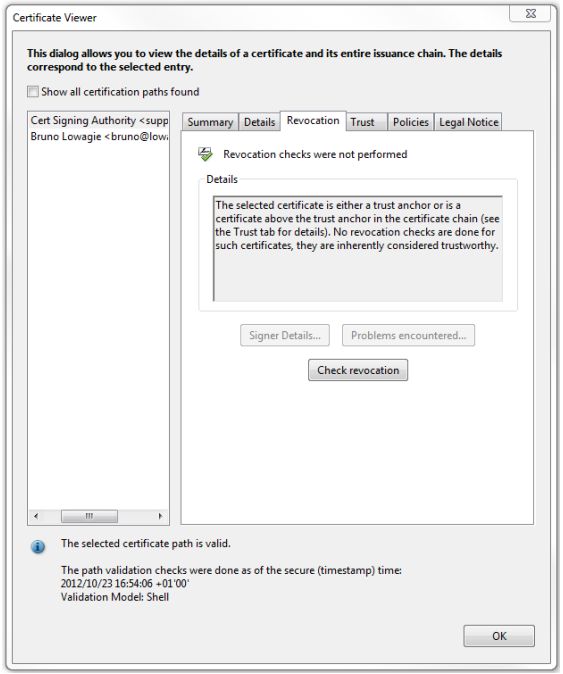
Certificates expire

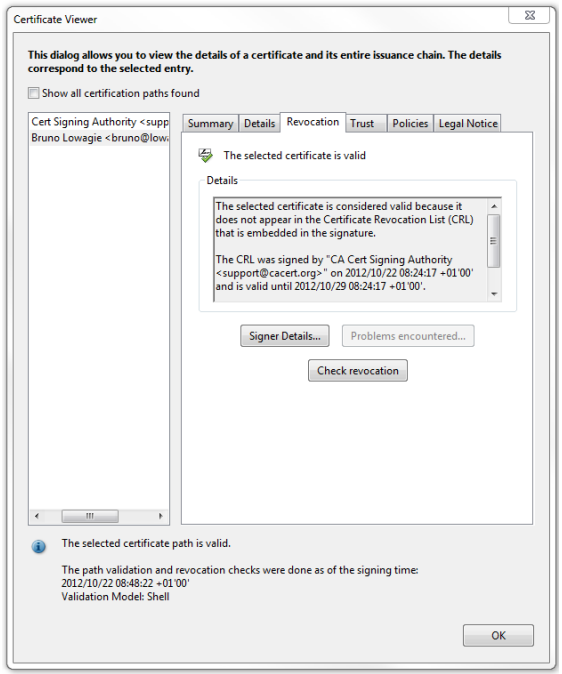


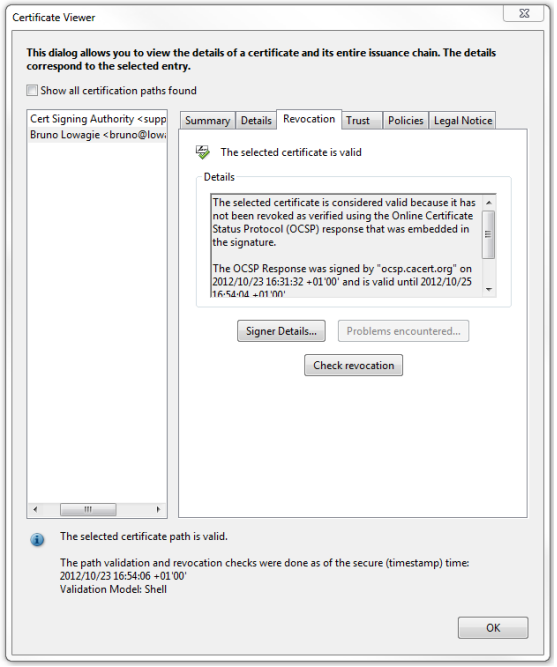
Certificates get revoked



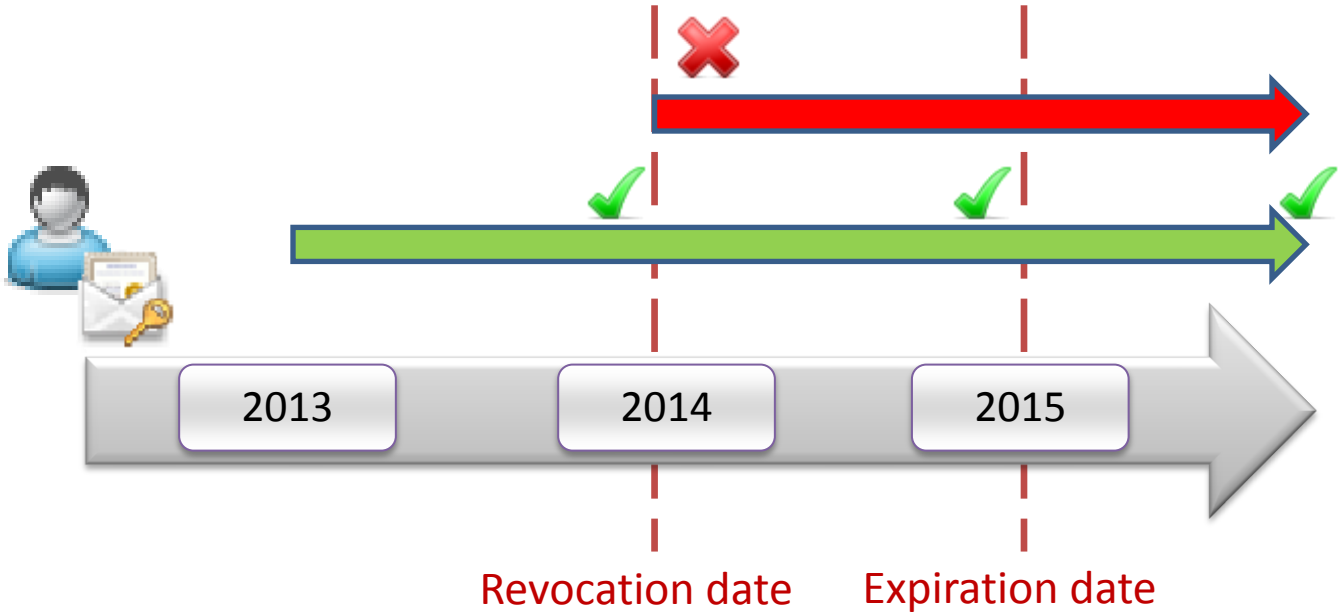
CA: CRL and OCSP



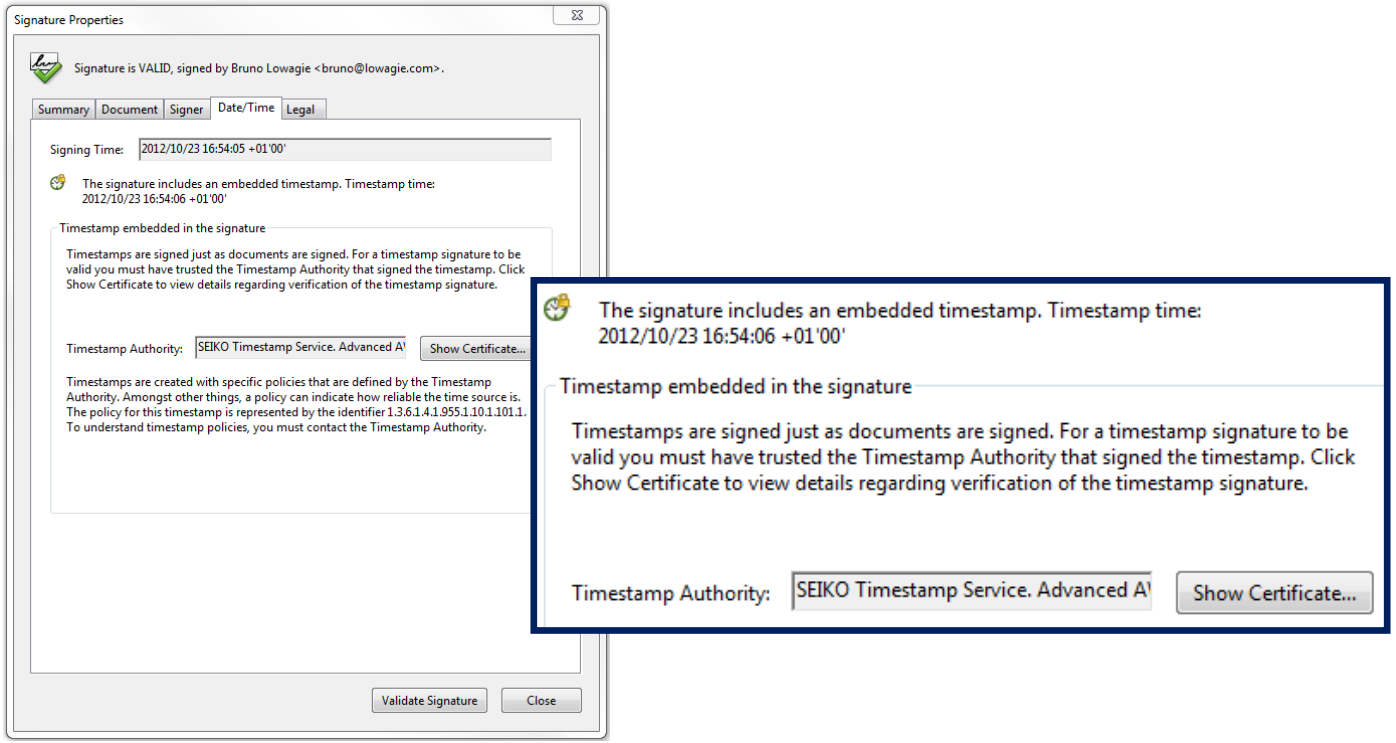




How to survive revocation / expiration?



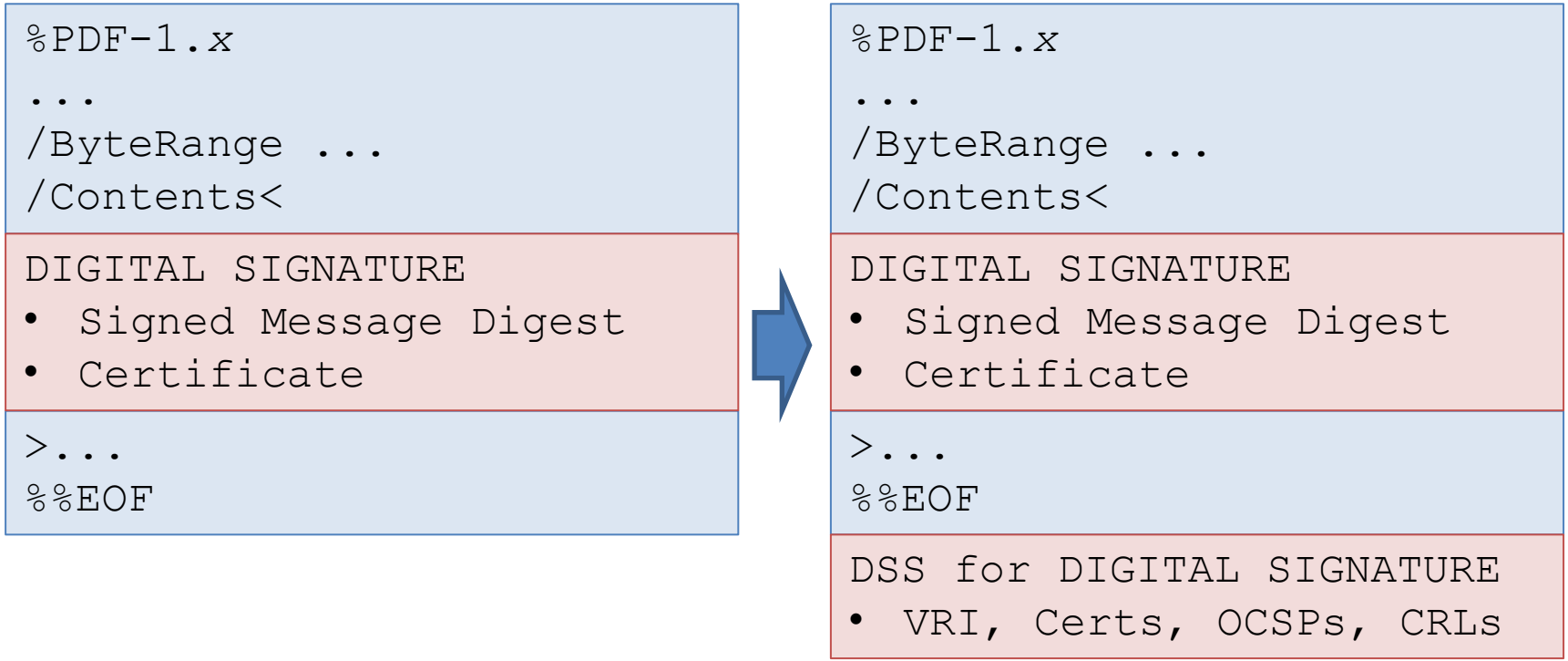
Timestamps



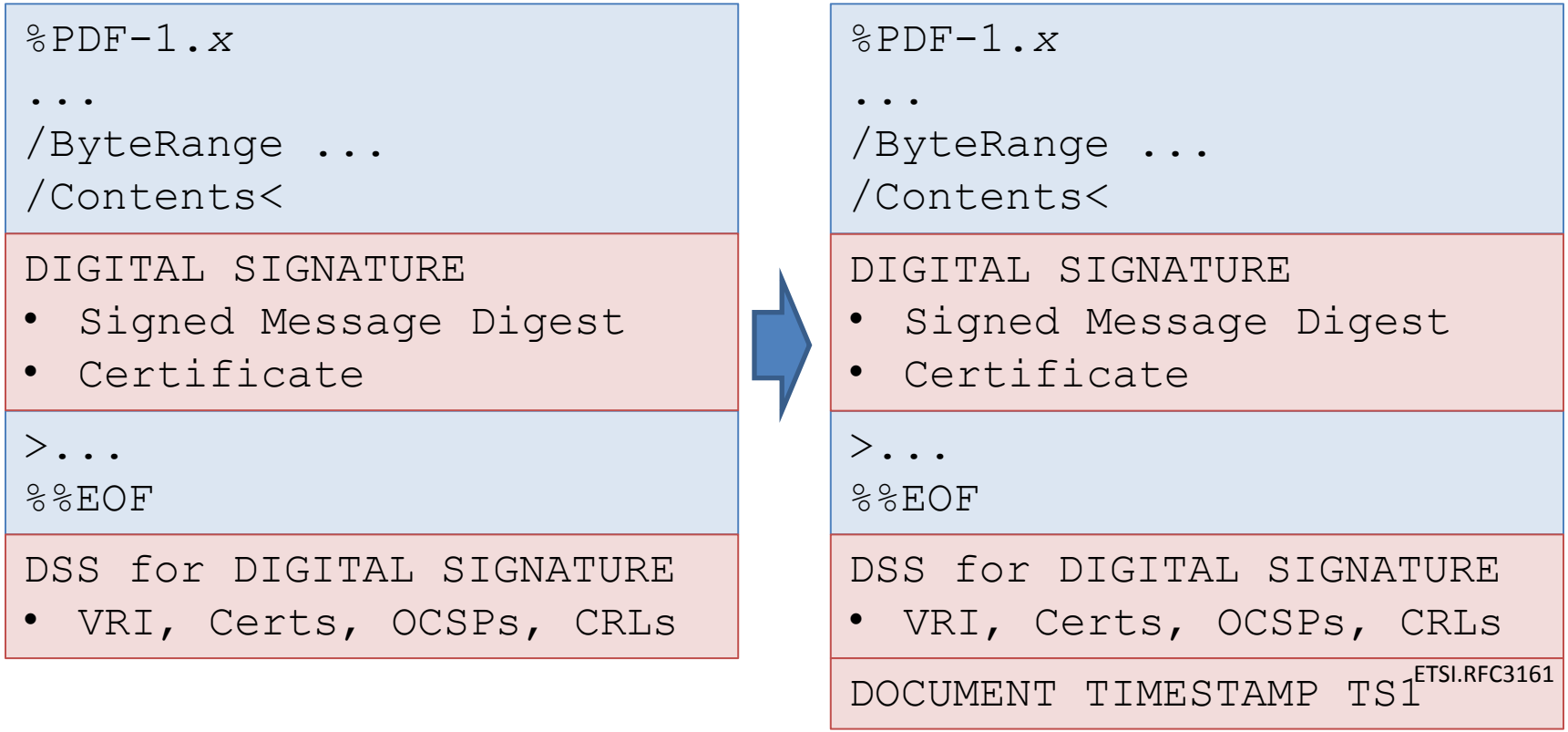
What to do when:

- There’s no CRL/OCSP/TS in the document?
- The certificate is about to expire in one of your documents?
- The hashing/encryption algorithm is about to be deprecated?

Document Security Store (DSS)



Document-level timestamp



%PDF-1.*x*
...
/ByteRange ...
/Contents<

DIGITAL SIGNATURE
- Signed Message Digest
- Certificate

>...
%%EOF

DSS for DIGITAL SIGNATURE
- VRI, Certs, OCSPs, CRLs

DOCUMENT TIMESTAMP TS1

Every signed document needs to be “kept alive”

%PDF-1.*x*
...
/ByteRange ...
/Contents<

DIGITAL SIGNATURE
- Signed Message Digest
- Certificate

>...
%%EOF

DSS for DIGITAL SIGNATURE
- VRI, Certs, OCSPs, CRLs

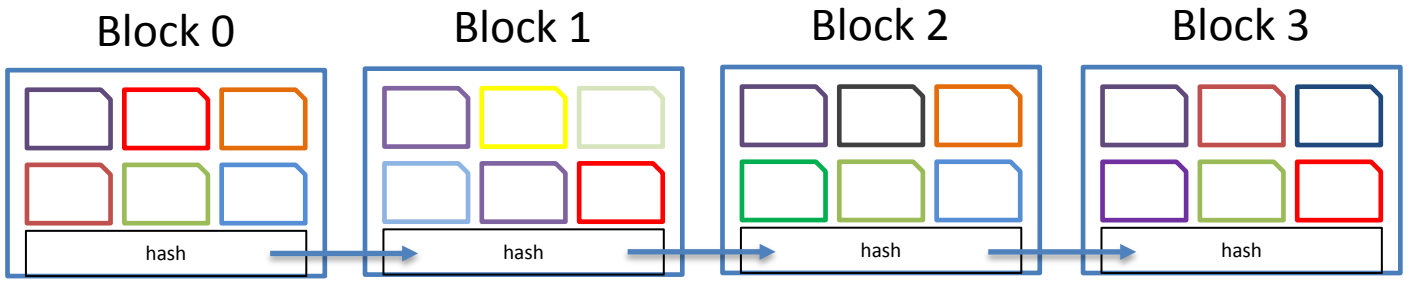
DOCUMENT TIMESTAMP TS1

DSS for TS1

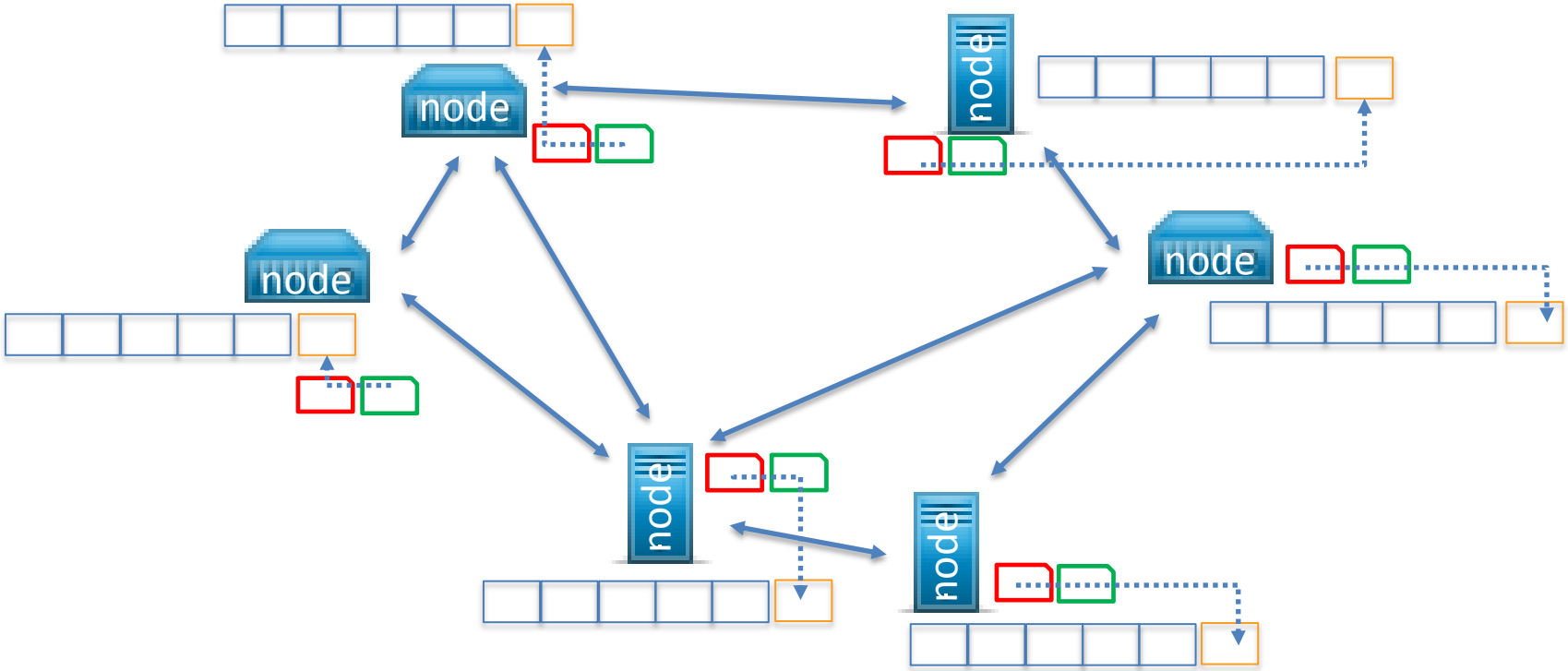
DOCUMENT TIMESTAMP TS2

How Blockchain can help…

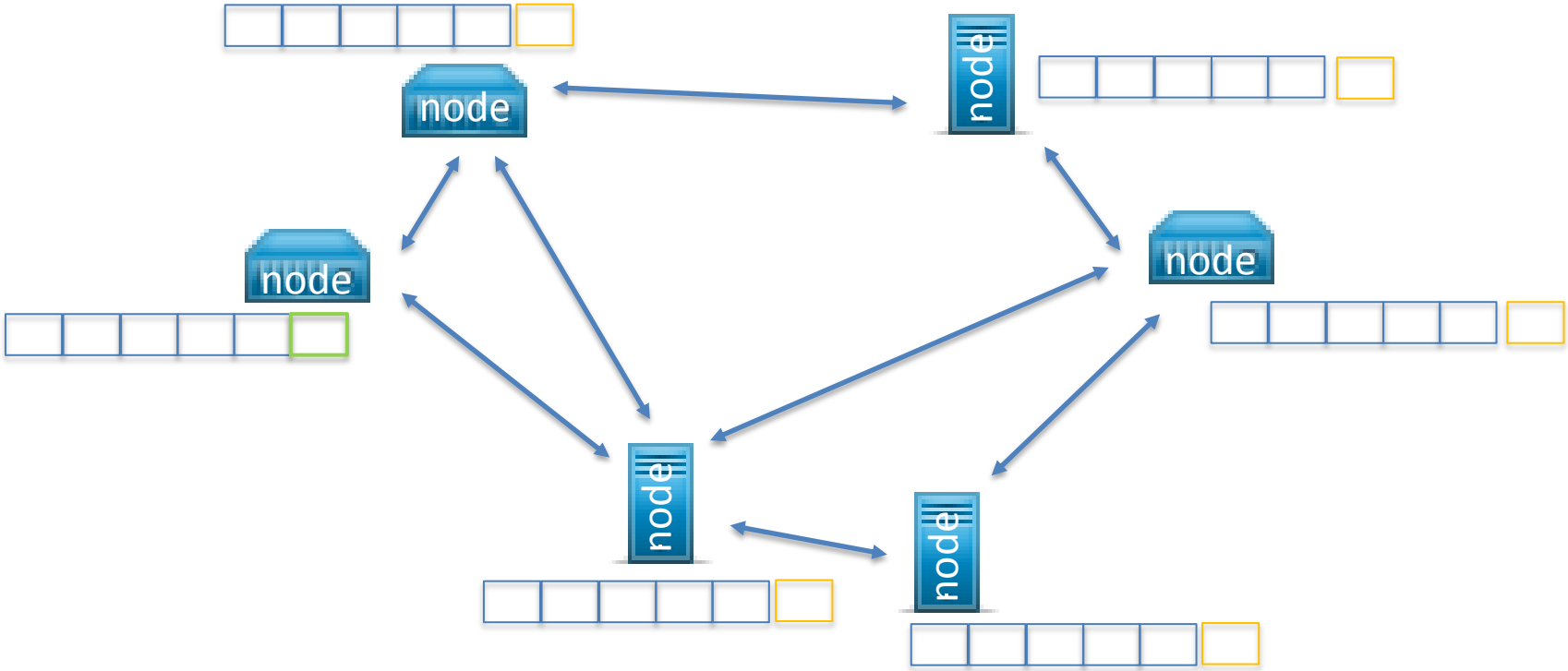
Chain of Blocks



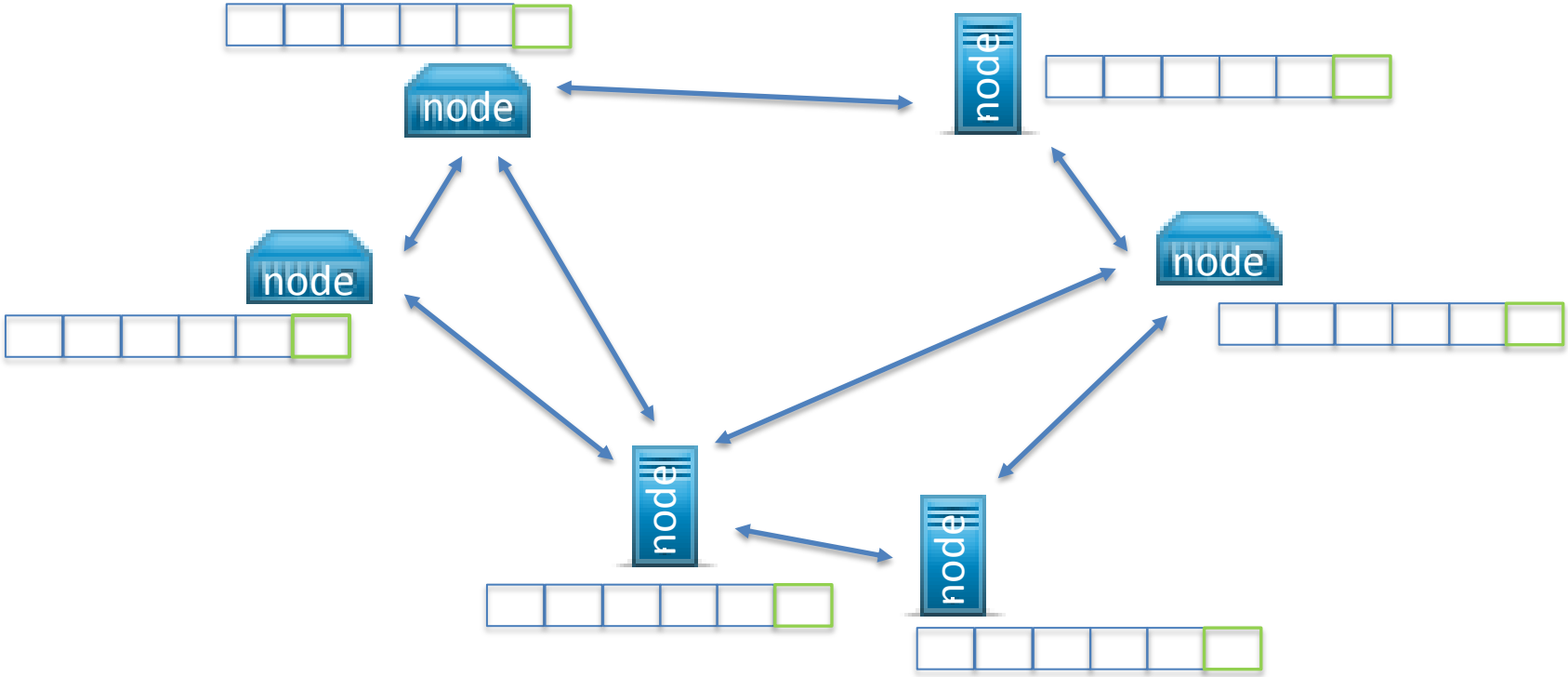
Records are distributed



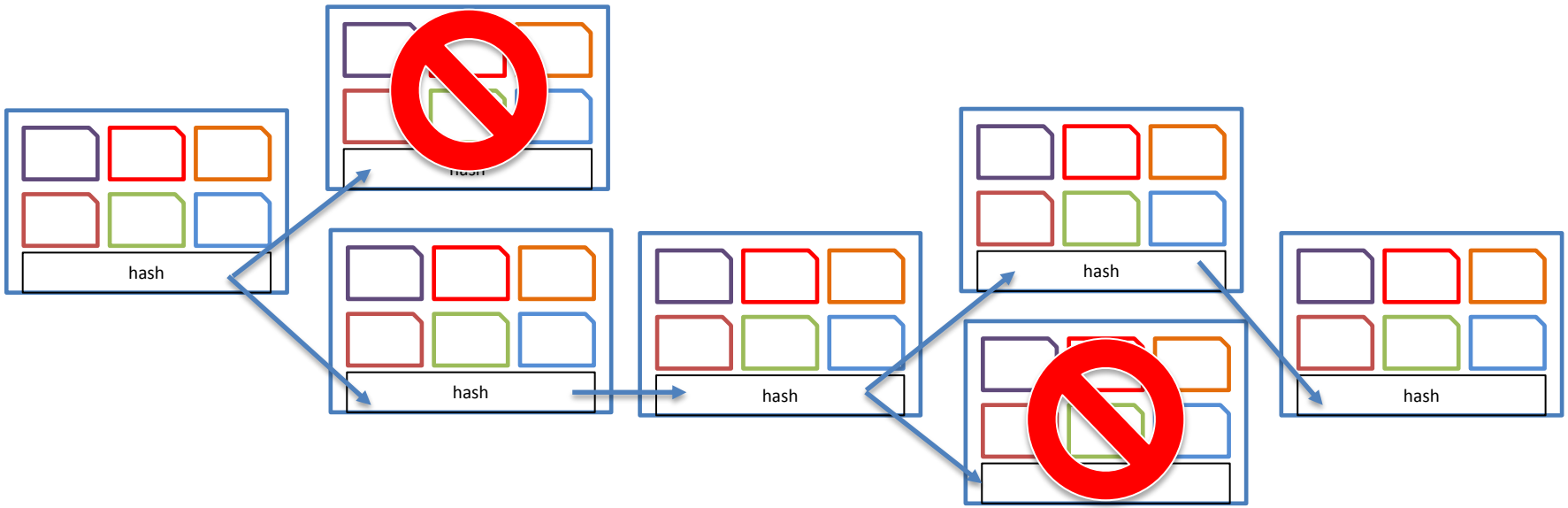
Challenge to add block to chain



Proof of work done!

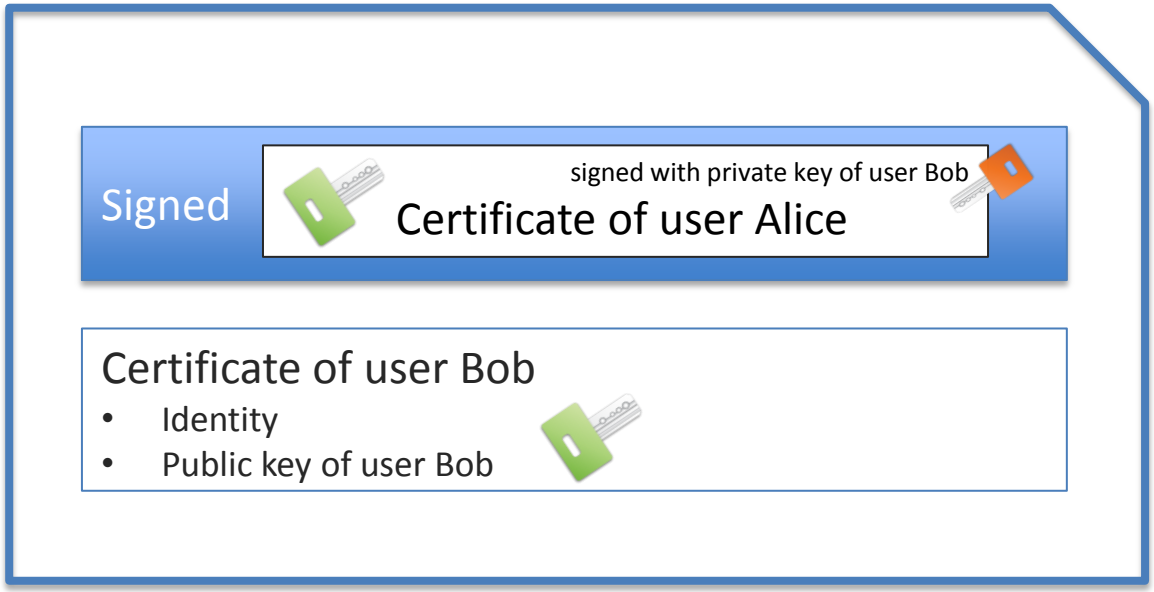


Longest chain is valid chain

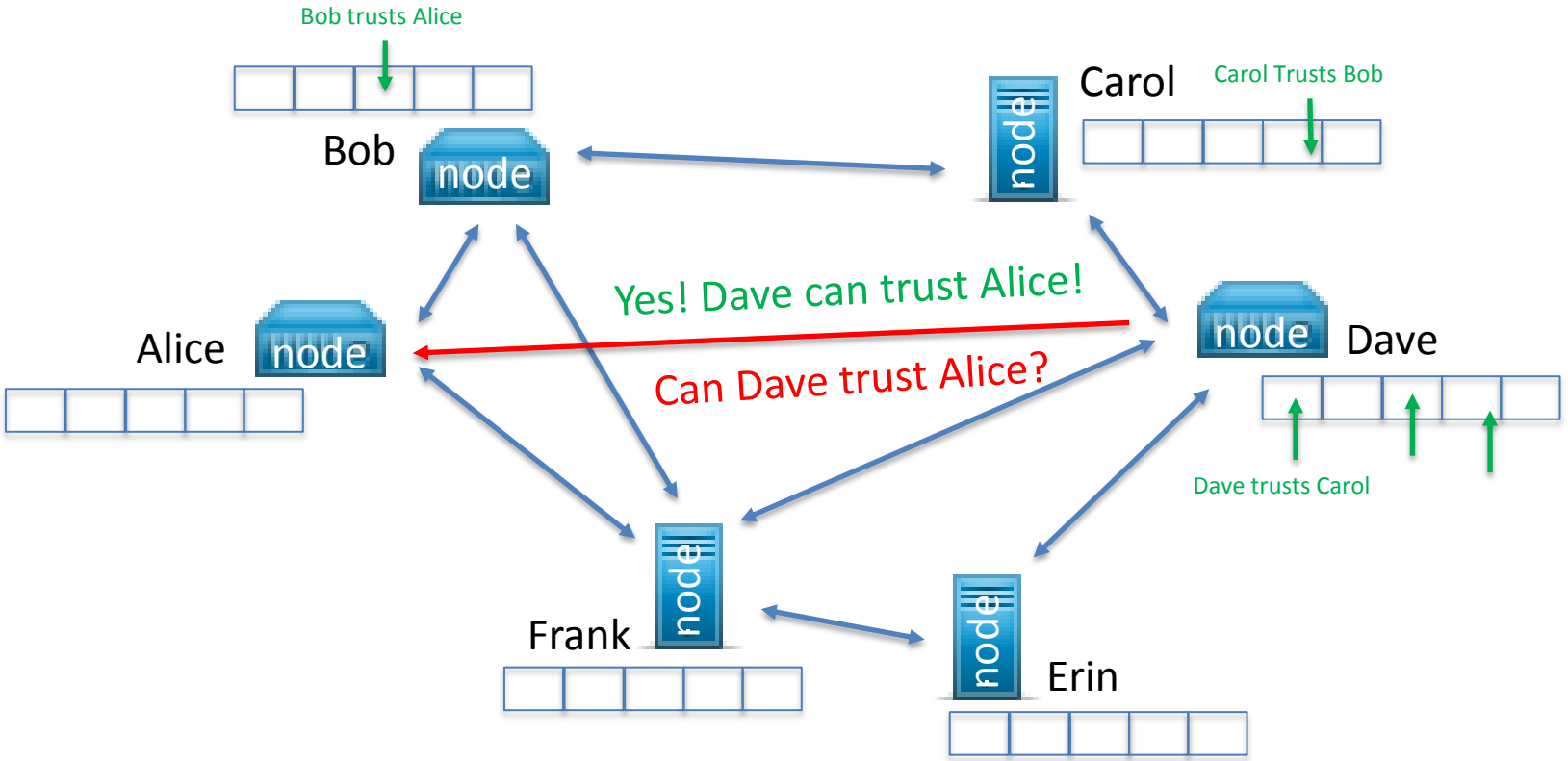


Blockchain and Web of Trust

Web of trust record

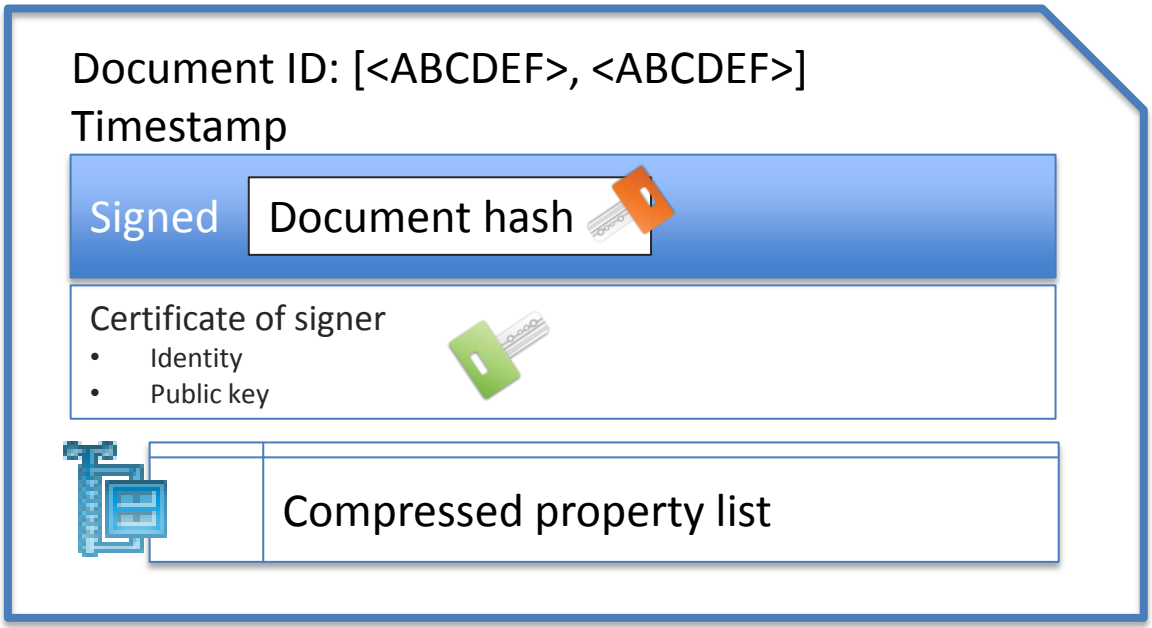


Web of Trust



Blockchain for documents

Document record



File identifiers: mandatory in PDF 2.0

ID	array	(*Required in PDF 2.0 or if an* ***Encrypt*** *entry is present; optional otherwise; PDF 1.1*) An array of two byte-strings constituting a file identifier (See 14.4, "File identifiers") for the file. The **ID** array shall (PDF 2.0) have a minimum length of 16 bytes. If there is an **Encrypt** entry, this array and the two byte-strings shall be direct objects and shall be unencrypted. *NOTE 2 Because the* ***ID*** *entries are not encrypted it is possible to check the* ***ID*** *key to assure that the correct file is being accessed without decrypting the file. The restrictions that the string be a direct object and not be encrypted assure that this is possible.* *NOTE 3 Although this entry is optional prior to PDF 2.0, its absence might prevent the file from functioning in some workflows that depend on files being uniquely identified.* *NOTE 4 The values of the* ***ID*** *strings are used as input to the encryption algorithm. If these strings were indirect, or if the* ***ID*** *array were indirect, these strings would be encrypted when written. This would result in a circular condition for a PDF reader: the* ***ID*** *strings need be decrypted in order to use them to decrypt strings, including the* ***ID*** *strings themselves. The preceding restriction prevents this circular condition.*

Impossible to know if an ID pair is unique if you don't know which IDs are already in use.

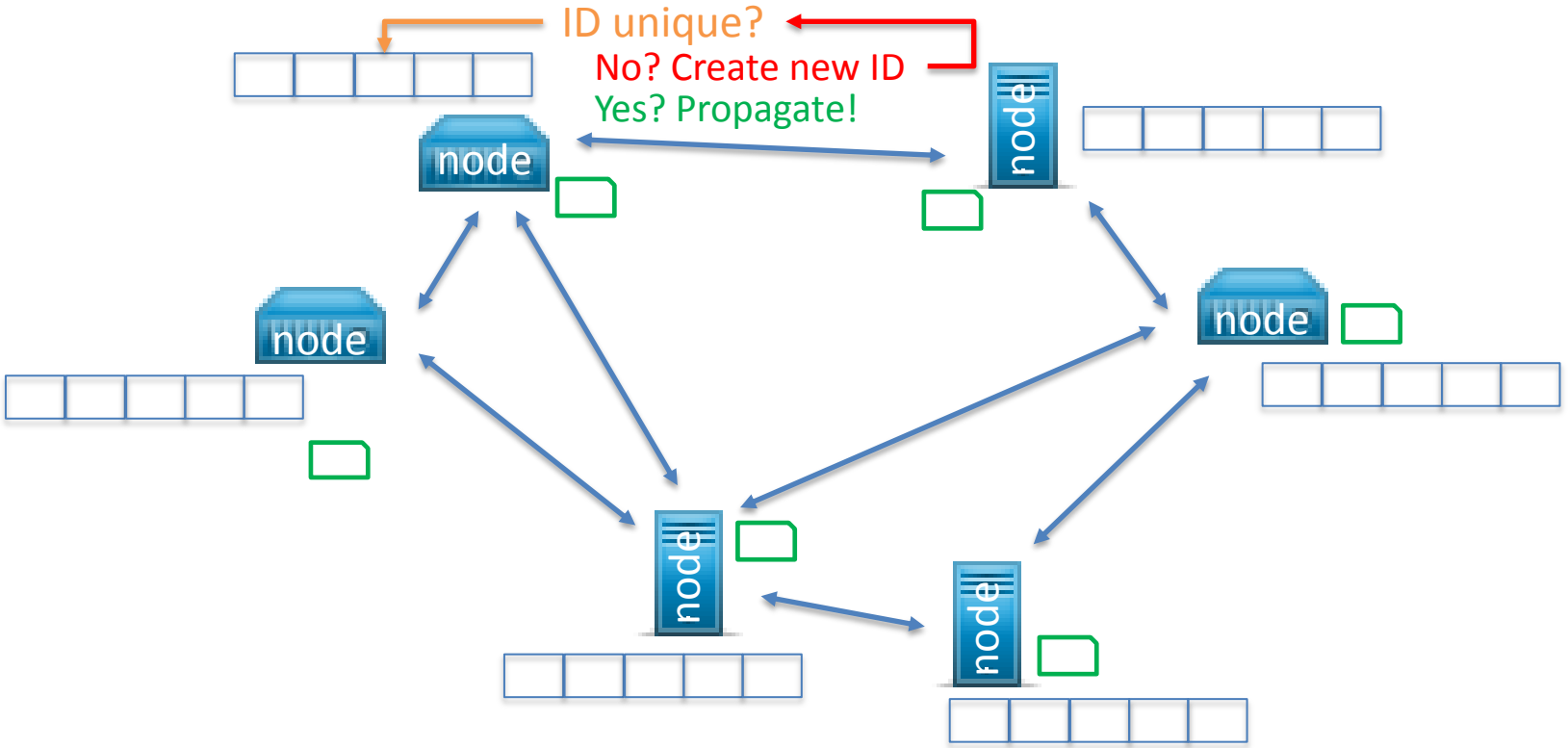
14.4 File identifiers

File identifiers shall be defined by the **ID** entry in a PDF file's trailer dictionary (see 7.5.5, "File trailer"). The value of this entry shall be an array of two byte strings. The first byte string shall be a permanent identifier based on the contents of the file at the time it was originally created and shall not change when the file is updated. The second byte string shall be a changing identifier based on the file's contents at the time it was last updated (see 7.5.6, "Incremental updates"). When a file is first written, both identifiers shall be set to the same value. If the first identifier in the reference matches the first identifier in the referenced file's **ID** entry, and the last identifier in the reference matches the last identifier in the referenced file's **ID** entry, it is very likely that the correct and unchanged file has been found. If only the first identifier matches, a different version of the correct file has been found.

PDF writers should attempt to ensure the uniqueness of file identifiers. This may be achieved by computing them by means of a message digest algorithm such as MD5 (described in Internet RFC 1321, *The MD5 Message-Digest Algorithm*), using the following information:

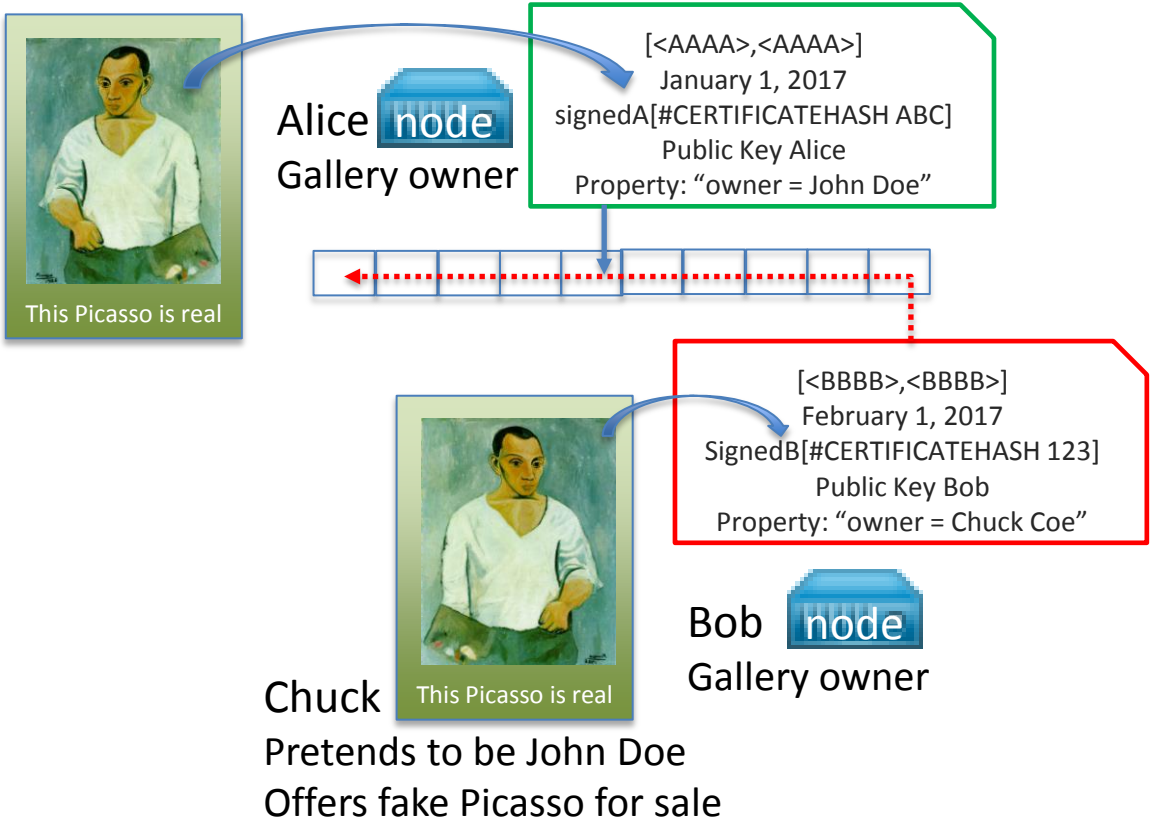
- The current time;
- A string representation of the file's location;
- The size of the file in bytes.

Check if a record already exists



Use case: certificate of authenticity

Use case: certificate of authenticity



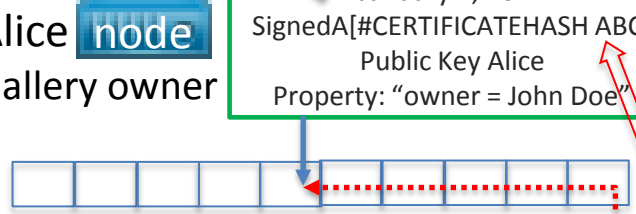
1st attempt to offer a forged painting with a fake certificate fails because the certificate can't be found on the chain.

Use case: certificate of authenticity

Alice
Gallery owner

node

[<AAAA>,<AAAA>]
January 1, 2017
SignedA[#CERTIFICATEHASH ABC]
Public Key Alice
Property: “owner = John Doe”



[<AAAA>,<AAAA>]
February 1, 2017
SignedB[#CERTIFICATEHASH DEF]
Public Key Bob
Property: “owner = Chuck Coe”

Bob
Gallery owner

node

Chuck
Pretends to be John Doe
Offers fake Picasso for sale

2nd attempt to offer a forged painting with a fake certificate fails because the hashes don’t correspond.

Use case: certificate of authenticity

Alice
node
Gallery owner

[<AAAA>,<AAAA>]
January 1, 2017
SignedA[#CERTIFICATEHASH ABC]
Public Key Alice
Property: “owner = John Doe”

John Doe
Offers real Picasso for sale

[<AAAA>,<AAAA>]
February 1, 2017
SignedB[#CERTIFICATEHASH ABC]
Public Key Bob
Property: “owner = Richard Roe”

Bob
node
Gallery owner

History shows that John Doe owned an object of art from January 1, 2017 until February 1, 2017, and that Richard Roe is the current owner

Use case: certificate of authenticity

Alice
Gallery owner

node

[<AAAA>,<AAAA>]
January 1, 2017
SignedA[#CERTIFICATEHASH ABC]
Public Key Alice
Property: “owner = John Doe”

[<AAAA>,<AAAA>]
January 1, 2017
SignedC[#CERTIFICATEHASH ABC]
Public Key Carol
Property: “owner = John Doe”

node

Carol
Insurance services

Carol notices pending sale
Alerts John and Bob

[<AAAA>,<AAAA>]
February 1, 2017
SignedB[#CERTIFICATEHASH ABC]
Public Key Bob
Property: “owner = Chuck Coe”

Bob
Gallery owner

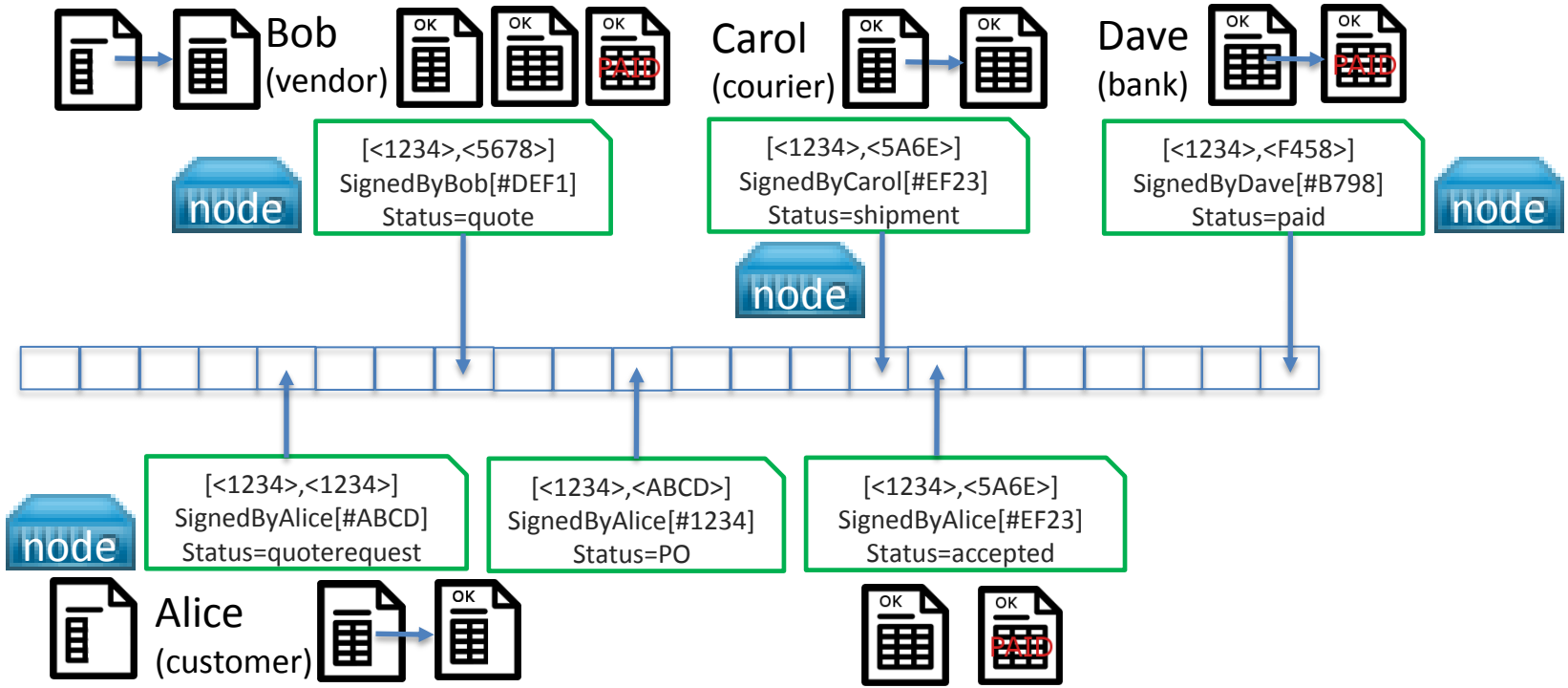
node

Chuck
Pretends to be John Doe
Offers fake Picasso for sale

3rd attempt to offer a forged painting with a real certificate fails because Carol monitors the certificate for John Doe.

Use case 2: Supply chain

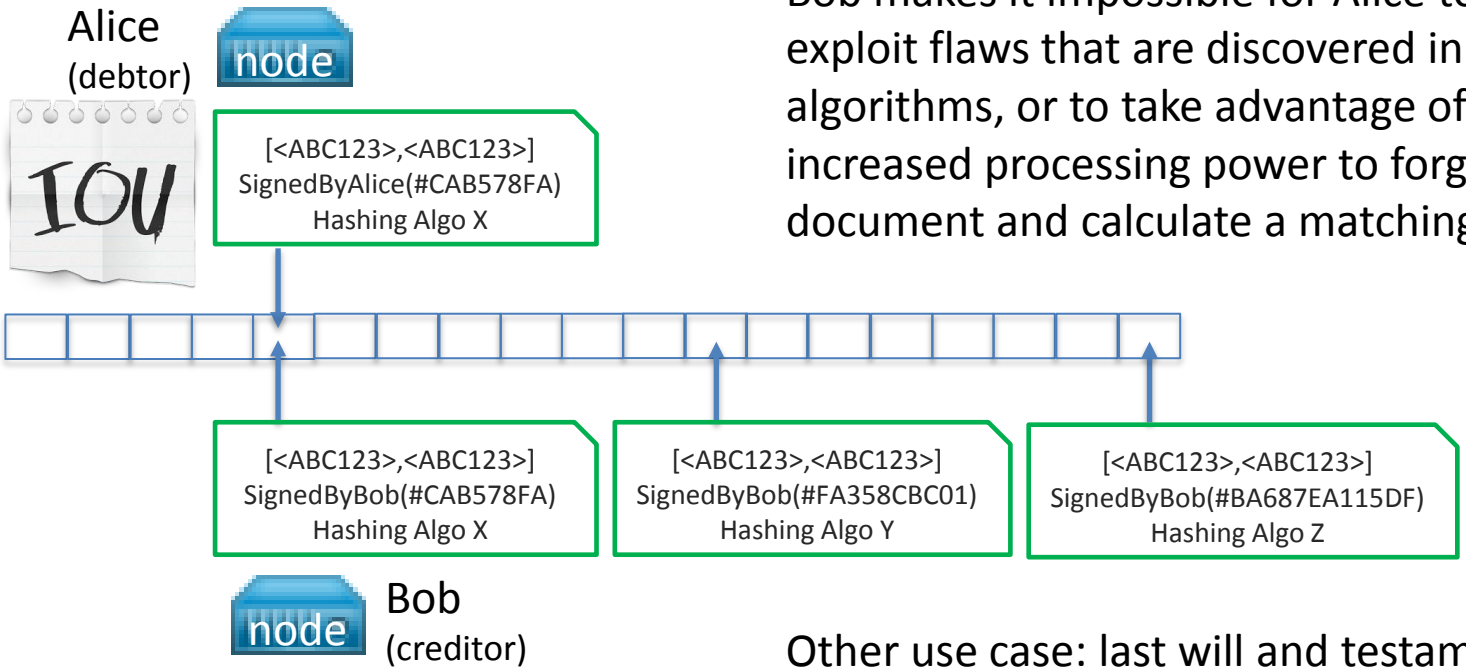
Supply chain



Use case 3: Long-Term Validation

Castello di Amorosa, Calistoga (Napa Valley), CA

Renewing a signature



Bob makes it impossible for Alice to exploit flaws that are discovered in old algorithms, or to take advantage of increased processing power to forge the document and calculate a matching hash.

Other use case: last will and testament.

Thank you!