



# In PKI We Trust: Validating Electronic Documents and Digital Signatures

James C. King  
Senior Principal Scientist  
Adobe Systems Incorporated



# “Signing” Electronic Documents

Using PKI



# Sample Form To Be Signed

## ~~~Lease Agreement~~~

Lease agreement for property at 6698 Happy Mountain Road, Sleepy Hollow, NH. Rent is \$1200.00 per month payable on the first day of each month. The lease is to run month to month with 5 day notice to terminate by either party. A deposit of one month's rent from Lessee is required upon signing this lease and will be return at the end of the term, excessive damage being paid for first.

Name of Landlord

Date signed

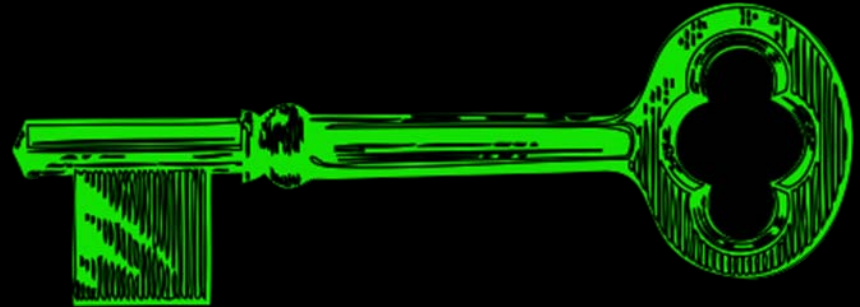
Signature of Landlord

Name of Lessee

Date signed

Signature of Lessee

~~~~~



# The Details

First: Public Key Encryption



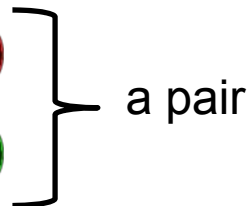
# Asymmetric Key Pairs (private and public pairing)

- A pair of **digital** “keys” generated from a random number

- Secret private key



- Published public key

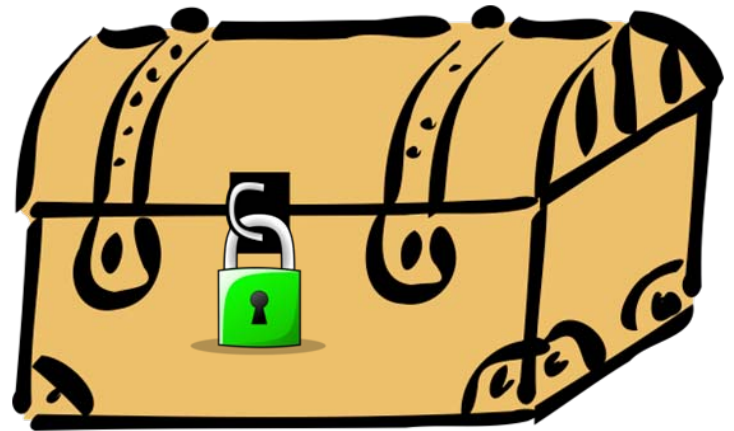
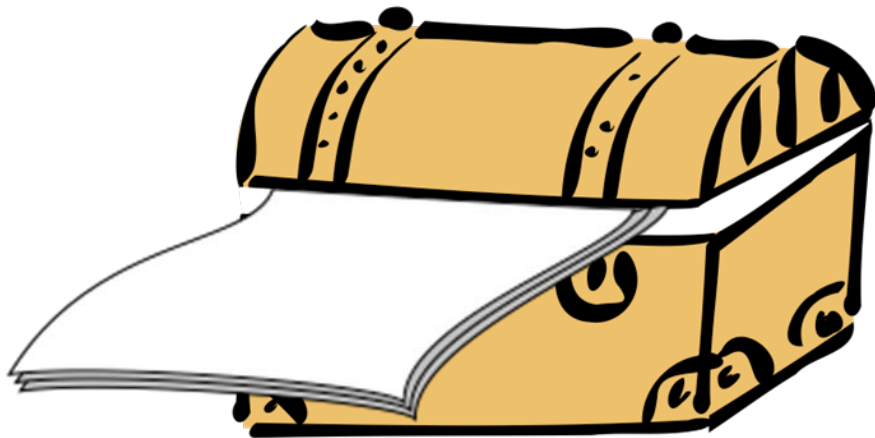


- Keys are represented as a set of bytes in the computer
- Keys sizes are measured in bits (e.g., 40 up to 1024) – technology dependent  
(5 bytes up to 128 bytes)



# Public Key Encryption

- Encryption
  - Uses a key to encrypt data
  - The data is garbled until decrypted

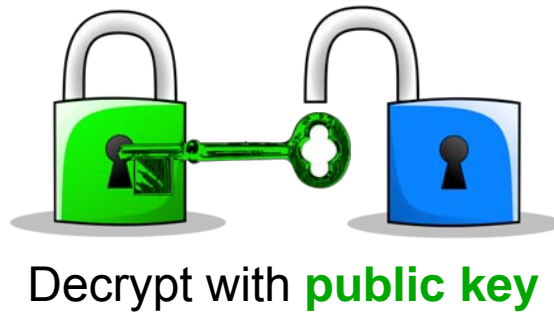
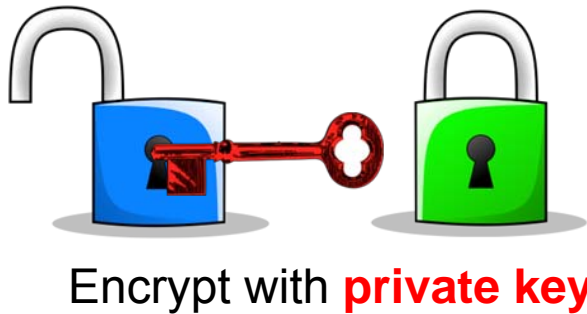


Encryption “locks” material

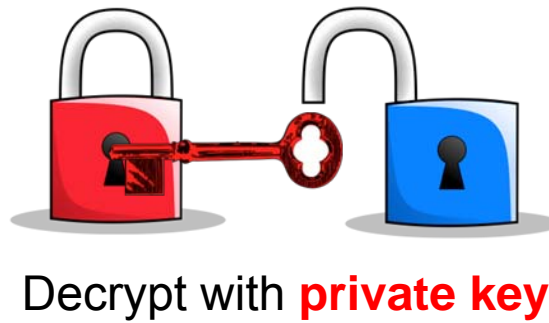
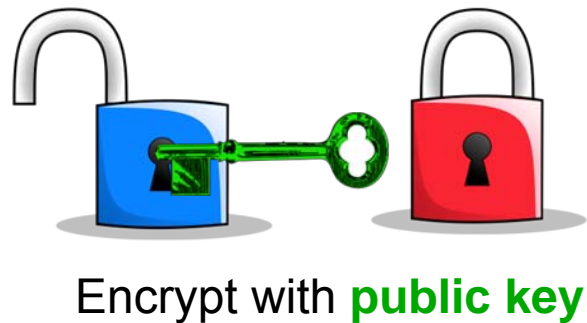


# Public Key Encryption Using Asymmetric Pairs

- What one key encrypts; (only) the paired key can decrypt



Owner encrypts;  
Anyone decrypts



Anyone encrypts;  
Owner decrypts



Kept private by owner



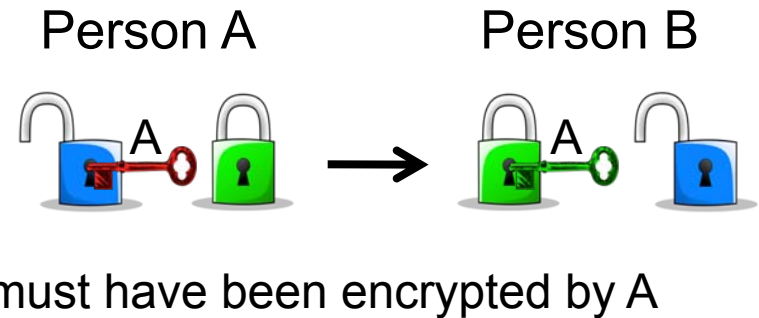
Available to public



# Public Key Encryption Using Asymmetric Pairs

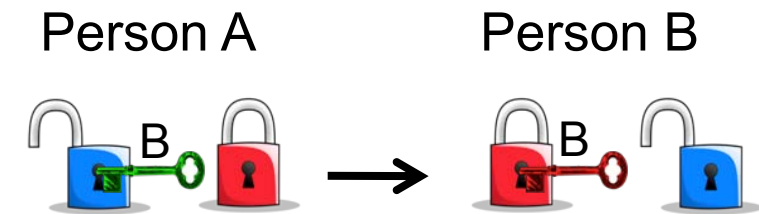
- Authentication & Signatures

- Person A encrypts with A's **private key**
- Anyone can decrypt using A's **public key**
- Will only decrypt with A's **public key** so it must have been encrypted by A



- Protected delivery

- Person A encrypts with B's **public key**
- Only B can decrypt with B's **private key**



- Strongly depends upon keeping **private keys** secret



# Putting Public/Private Key Pairs to Use





# Three Twists to the Base Story

- How do I obtain your public key and know it is really yours
  - The answer is **PKI** (Public Key Infrastructure)
- Encryption can be slow, so in many cases we use “hashes” or “digests”
  - Also desirable to be able to detect changes without encrypting
- If encrypted data is changed it will fail to decrypt properly
  - Prevents forgery



# Cryptographic Hashes

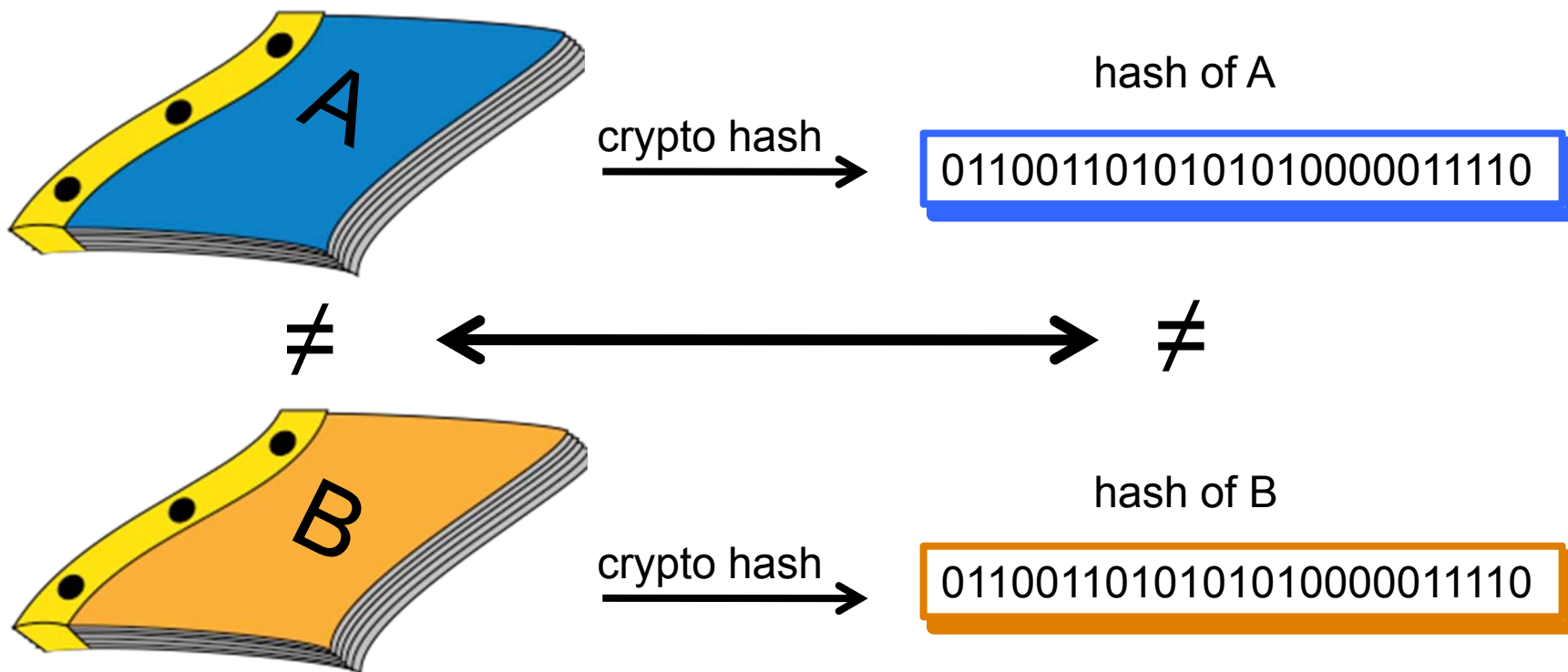
Also called Message Digests

0110011010101010000011110



# Cryptographic Hashes (i.e. Message Digests)

- A small string of bytes (hash) computed as a proxy for a large string of bytes
- Hash (small string—e.g., 64 bytes), large string typically a document or data file





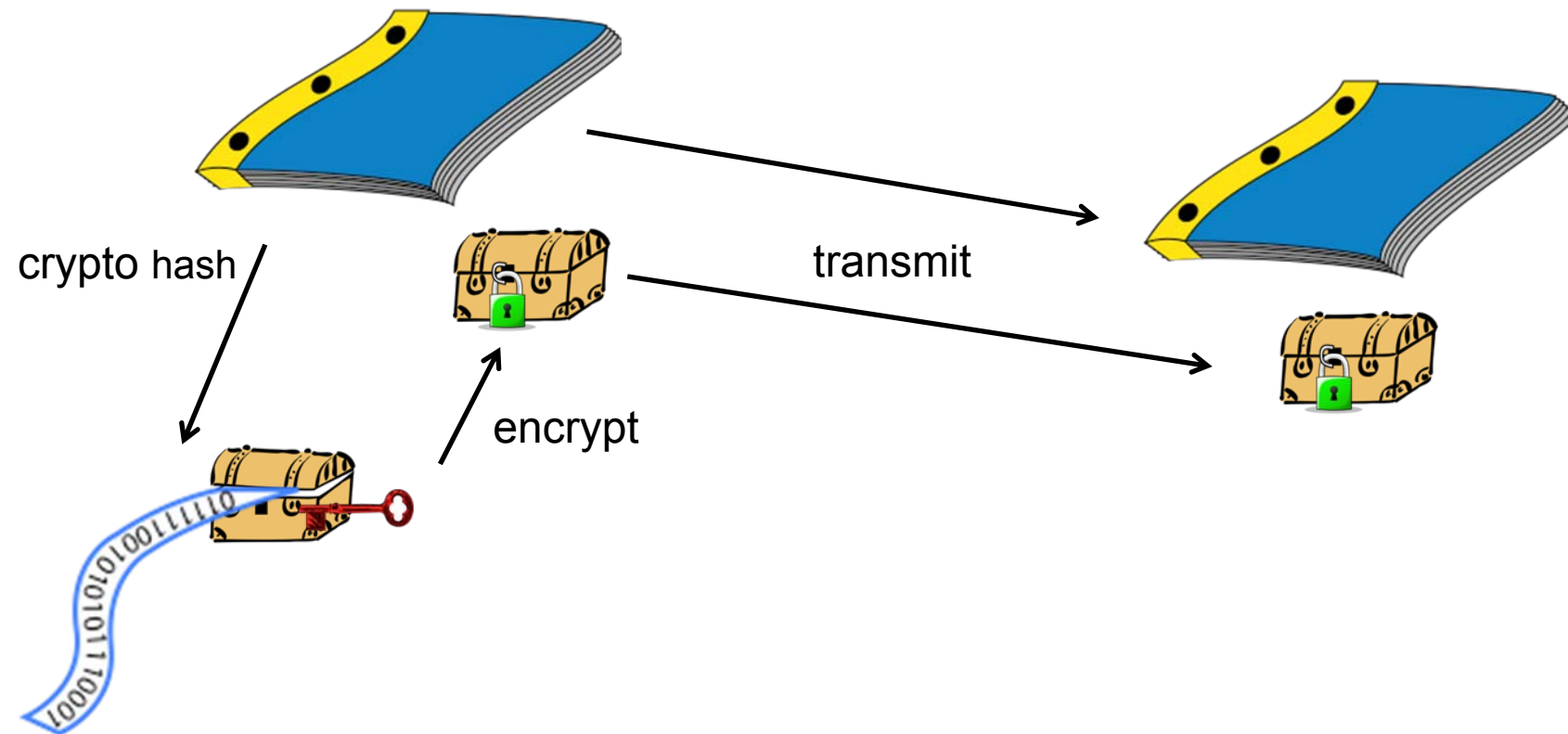
# Clashes of Cryptographic Hashes

- Given A and B, some meaningful documents or data
- Clash: large string A and large string B have same small string hash
  - Algorithms invented to make clashes **very rare** and
  - to make clashes **practically impossible** to create
- Breaking a hash algorithm:
  - Given large string A and its computed hash X,
  - devise a different large string B whose hash is also X
- Algorithms based upon computationally prohibitive calculations



# Using Cryptographic Hashes to Detect Change

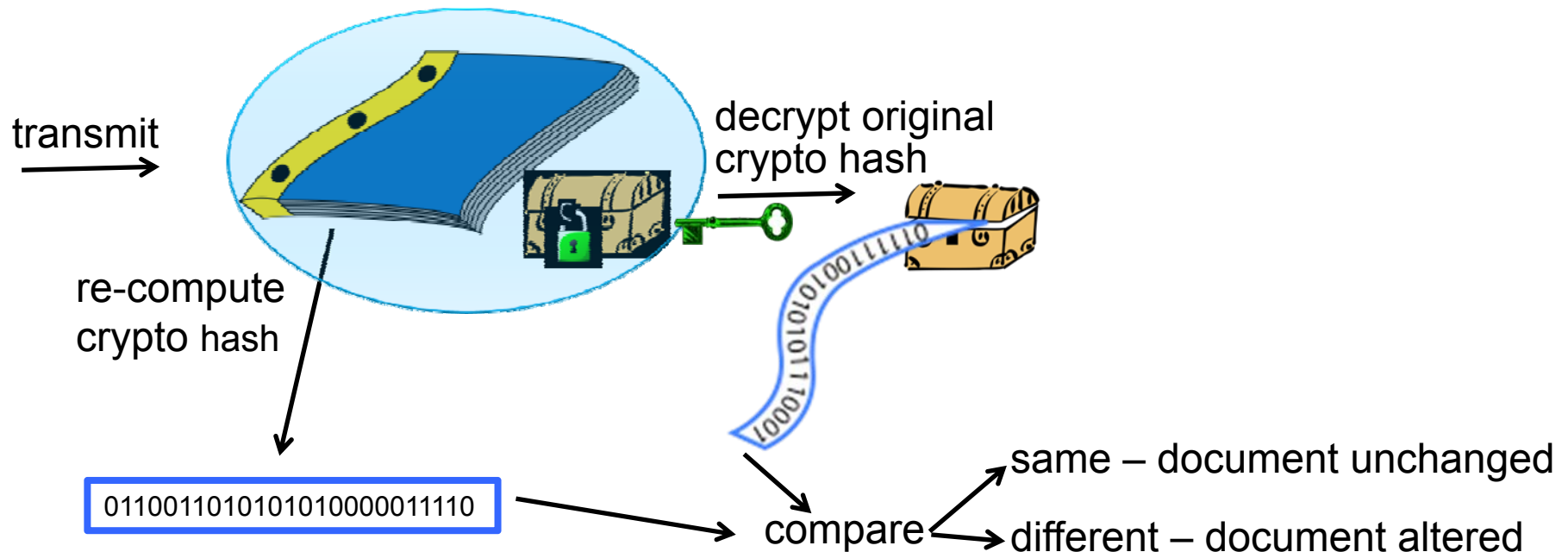
- Compute a cryptographic hash (c-hash) of the document or data
- Encrypt the hash with a private key
- Transmit the document/data and the encrypted hash





# Using Cryptographic Hashes to Detect Change

- Compute the c-hash again on the received document/data
- Decrypt the c-hash transmitted with the public key
- Compare old c-hash with newly computed result
  - Same – document is unchanged
  - Different – document has been altered





# Public Key Infrastructure (PKI)

*Finally!*



# Public Key Infrastructure (PKI)

- How do I obtain your public key and know it is really yours?
  - If someone vouches for you, how do I know who *they* are?
  - The answer is **PKI** (Public Key Infrastructure)



# Certificate Authorities (CA's)

- Today in practice, PKI is a collection of Certificate Authorities' (CA's) servers
- Certificate
  - Public key
  - Plus identifying information and dates
- Certificate is “signed” by an authority
  - answering the question: who says so



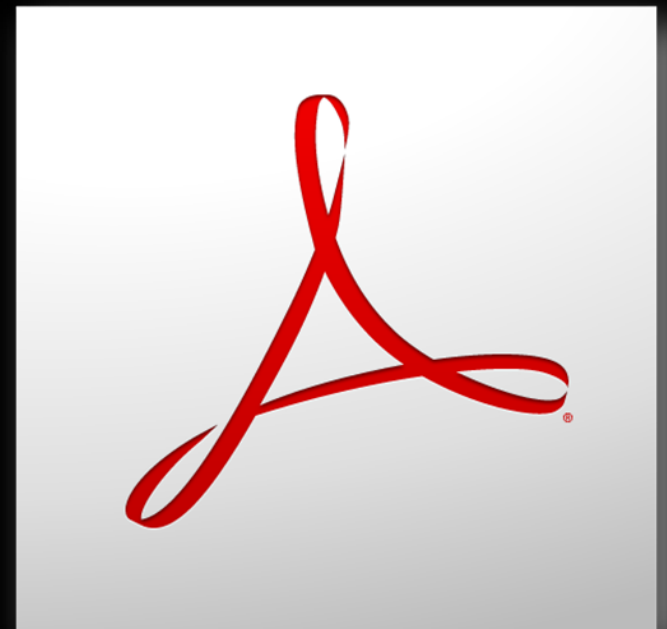
# Certificate Authority Hierarchy

- The CA has its own public/private key pair issued by a higher level CA
  - Hierarchy ends at a “root” CA
  - Trust of a root public key is done by other means
- Adobe supplies an Adobe root with Adobe Reader and Acrobat
  - Cooperating CA’s issue certificates referring to the Adobe root
  - Cooperating CA’s agree to stringent rules for issuing certificates
- Adobe Certified Document Services (CDS):  
[http://www.adobe.com/security/partners\\_cds.html](http://www.adobe.com/security/partners_cds.html)
- Adobe Approved Trusted List (AATL):  
<http://www.adobe.com/security/approved-trust-list.html>



# Portable Document Format (PDF)

Digital Signatures & PKI in Practice





# Two Signature Types Supported by Adobe Reader/Acrobat

Approval Signatures

Document Certification Signatures



# Certifying a Document's Source and Integrity

- Creator signs the document
  - Usually with non-visible signature
  - Assures who the document came from and
  - No changes were made to the document since it left the source
  
- Only one certification signature allowed in PDF
  - By the document creator at the start



# Approval Signatures

- Almost always visible signatures
- Often part of a workflow requiring further process of the document
- How can a document undergo further change after a signature



# Sample Form To Be Signed

## ~~~Lease Agreement~~~

Lease agreement for property at 6698 Happy Mountain Road, Sleepy Hollow, NH. Rent is \$1200.00 per month payable on the first day of each month. The lease is to run month to month with 5 day notice to terminate by either party. A deposit of one month's rent from Lessee is required upon signing this lease and will be return at the end of the term, excessive damage being paid for first.

Name of Landlord

Date signed

Signature of Landlord

Name of Lessee

Date signed

Signature of Lessee

~~~~~



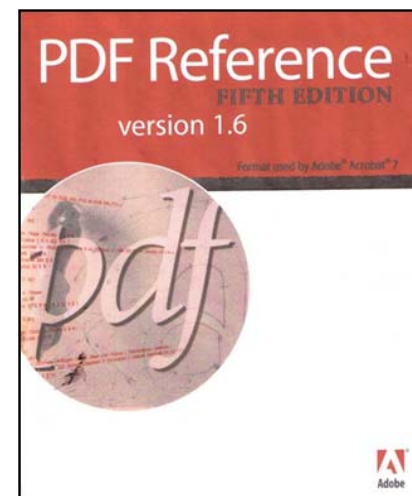
# PDF: An International Standard

ISO 32000



# Portable Document Format (PDF)

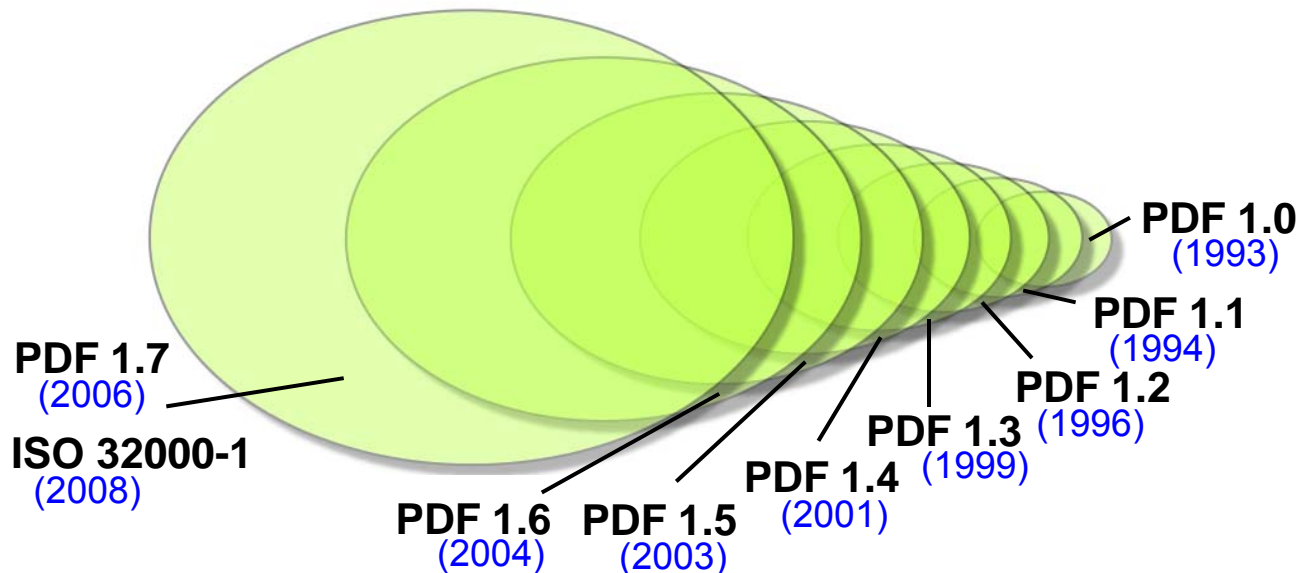
- Defined by Adobe
  - Full specification released as Addison Wesley book in June 1993
  - Specification on Adobe's website and/or revised book for each new release
- **Thousands** of software applications that process PDFs
- Created by **hundreds** of developers
- **Billions** of PDF files
- Large existing ecosystem





# PDF (1993 – 2008)

- An ISO standard: ISO 32000-1
  - Approved by ISO in January 2008
  - Published by ISO in July 2008
  - Same as PDF 1.7 which includes digital signatures
- Adobe has signed an intellectual property agreement with ISO
  - No Adobe restrictions to develop software to process PDF; never was

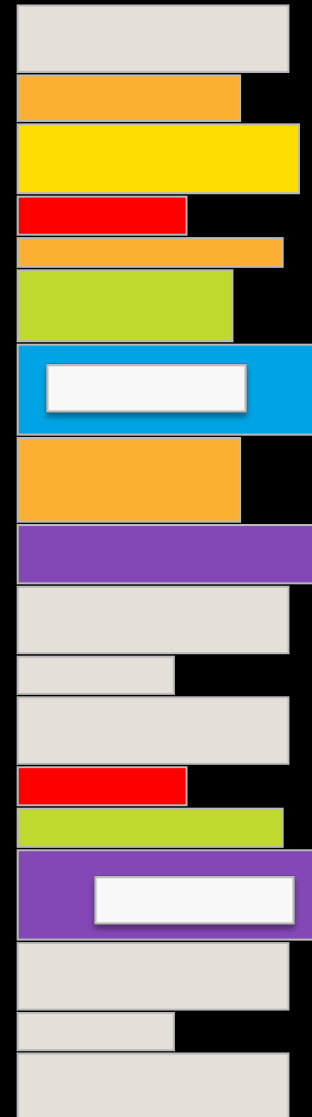


**Once a PDF,  
always a PDF**



# PDF Architecture

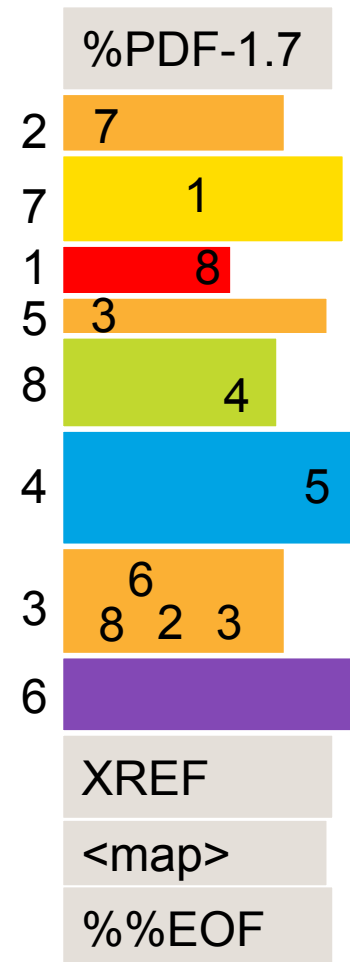
## PDF Objects (COS)





# PDF Objects

- PDF file is a collection of numbered objects
- Objects can reference each other by their numbers
- XREF at end of file maps numbers to file offsets
- Objects include: numbers, arrays, dictionaries, names, true, false, strings, streams

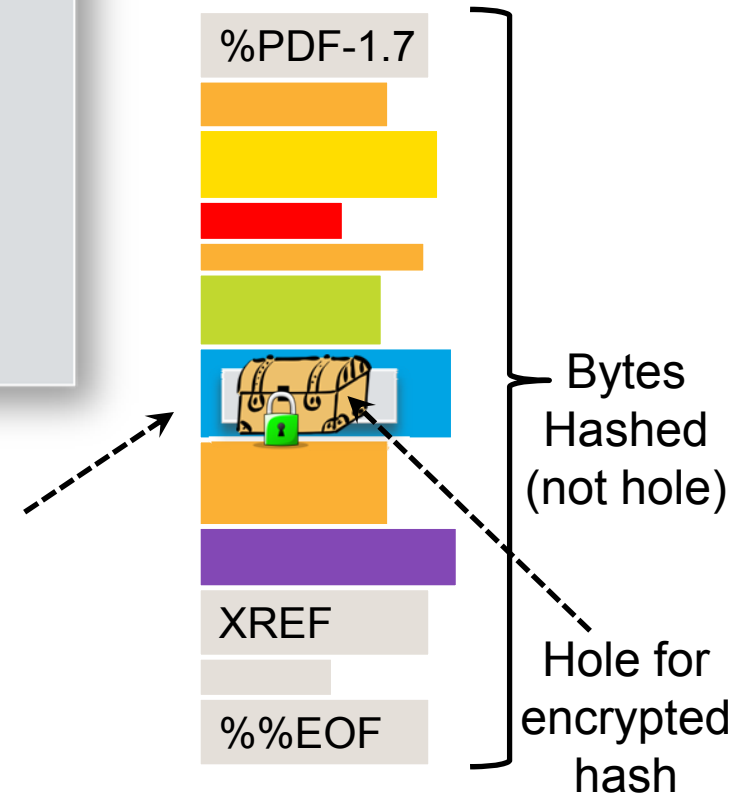




# Signature Object

- Signature Object contains encrypted message digest
  - Digesting skips the hole
  - Avoids the circular problem of digesting the digest
  - Digest dropped into hole after hash & encryption

Signature Dictionary Object

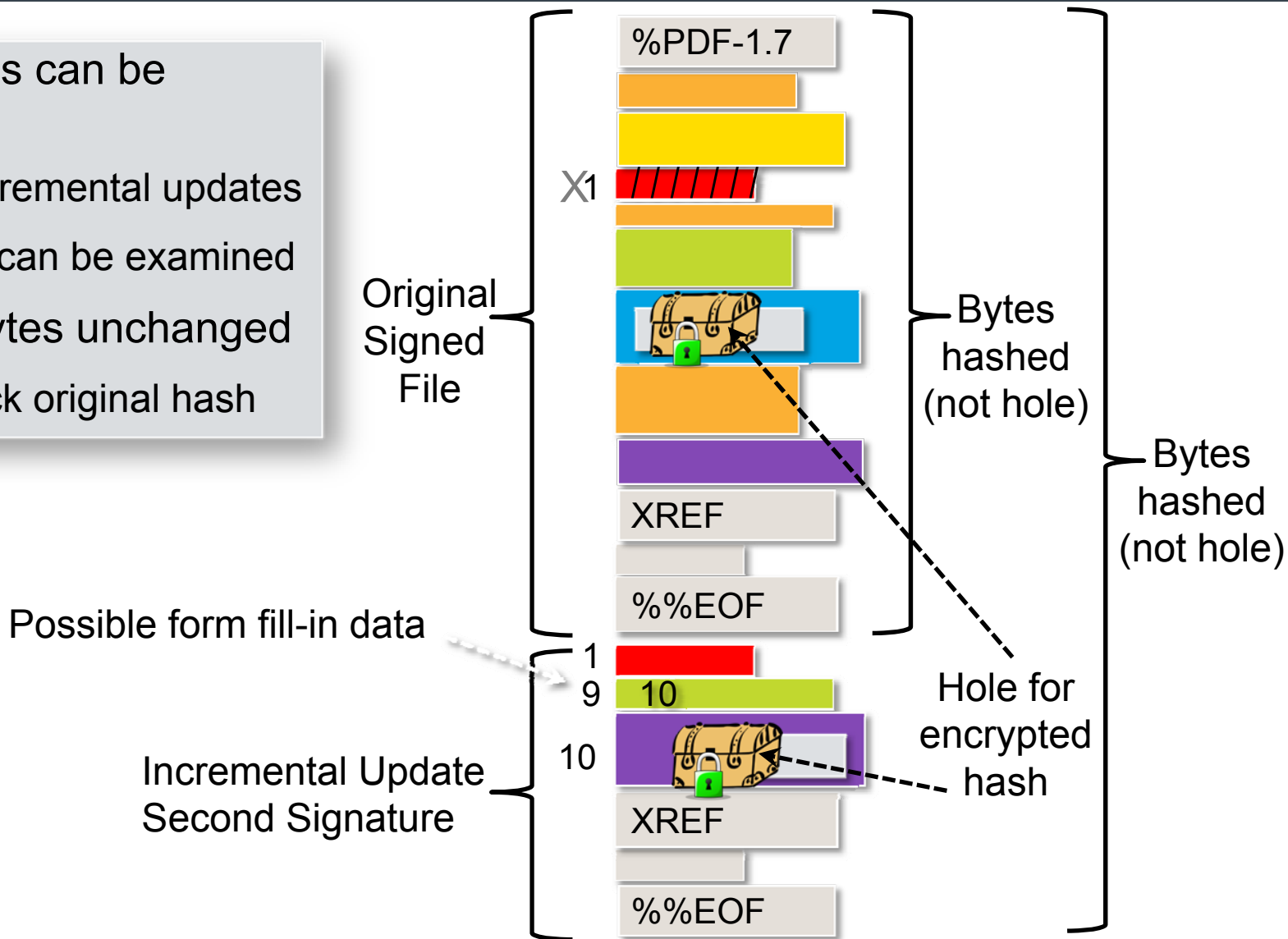


Signed PDF File



# Incremental Update to a Signed File

- Signed files can be changed!
  - Using incremental updates
  - changes can be examined
- Original bytes unchanged
  - Can check original hash





# References

- ISO 32000-1 (PDF 1.7) Specification
  - [http://www.adobe.com/devnet/acrobat/pdfs/PDF32000\\_2008.pdf](http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf) OR
  - <http://www.iso.org/iso/pressrelease.htm?refid=Ref1141>
- Blogs:
  - InsidePDF <http://blogs.adobe.com/insidepdf> (By Jim King)
  - Adobe Security <http://blogs.adobe.com/security> (by John B. Harris)
- King's home page  
<http://www.adobe.com/technology/people/sanjose/king.html>



**Adobe**