FIXITY & DATA INTEGRITY

DATA INTEGRITY

DATA INTEGRITY

PRESERVATION CONSIDERATIONS

- Data that can be **rendered**
- Data that is **properly formed** and can be **validated**
    - DROID, JHOVE, etc.

DATA DEGRADATION

HOW DO FILES LOSE INTEGRITY?

DATA DEGRADATION

HOW DO FILES LOSE INTEGRITY?

Storage: hardware issues

- Physical damage, improper orientation, magnets, dust particles, mold, disasters

Storage: software issues

- "bit rot", "flipped" bits, small electronic charge, solar flares, radiation

DATA DEGRADATION

HOW DO FILES LOSE INTEGRITY?

Transfer/Retrieval

- **Transfer** from one operating system or file system to another, transfer across network protocols,
- **Metadata loss**: example – Linux has no "Creation Date" (usually "**file system**" metadata)

Mismanagement

- **Permissions issues** (read/write allowed), human error

DATA PROTECTION

VERIFICATION

DATA PROTECTION

VERIFICATION

- Material proof or evidence that data is unchanged
- Material proof or evidence that data is well-formed and should be renderable
    - Example: Different vendors write code for standard formats in different ways

DATA PROTECTION

VERIFICATION

Verify that data is well-formed using...

DATA PROTECTION

VERIFICATION

Verify that data is well-formed using...

- **JHOVE**
- **DROID**
- **DVAnalyzer**
- **BWF MetaEdit**
- **XML Validator**
- **NARA File Analyzer**

WHOLE-FILE CONSISTENCY

FIXITY

FIXITY

BASIC METHODS

Manual checks of **file metadata** such as...

FIXITY

BASIC METHODS

Manual checks of **file metadata** such as...

- File name
- File size
- Creation date
- Modified date
- Duration (time-based media)

FIXITY

ADVANCED METHODS

FIXITY

ADVANCED METHODS

Cryptographic Hashcode

FIXITY

ADVANCED METHODS

Cryptographic Hashcode

checksum

hash

fixity check

digest

hashcode

digital signature

FIXITY

ADVANCED METHODS

Cryptographic Hashcode

checksum

hash

fixity check

digest

hashcode

digital signature

Mathematical function itself = "**hash**"

Hash used for fixity = "**checksum**"

FIXITY

WHAT IS A CHECKSUM?

FIXITY

WHAT IS A CHECKSUM?

Cryptographic Hashcode

- Algorithm analysis of file produces a **string**

FIXITY

WHAT IS A CHECKSUM?

Cryptographic Hashcode

- Algorithm analysis of file produces a **string**

f49c2e9f08490c055cddba6d3b049f94

FIXITY

WHAT IS A CHECKSUM?

Cryptographic Hashcode

- Algorithm analysis of file produces a **string**

f49c2e9f08490c055cddba6d3b049f94

- **MD5**: 128-bits consolidated into a 32-character hexadecimal (4 bits per character)

FIXITY

WHAT IS A CHECKSUM?

Cryptographic Hashcode

- Algorithm analysis of file produces a **string**

f49c2e9f08490c055cddba6d3b049f94

1882e58c22a2c132b7b9763d24ae9bffe68d00ab

- **MD5**: 128-bits consolidated into a 32-character hexadecimal (4 bits per character)
- **SHA-1**: 160-bits consolidated into a 40-character hexadecimal (4 bits per character)

FIXITY

CHECKSUM CODE

FIXITY

CHECKSUM CODE

```
md5 /Path-to-file/$filename.xml
```

FIXITY

CHECKSUM CODE

md5 /Path-to-file/$filename.xml

f49c2e9f08490c055cddba6d3b049f94

FIXITY

CHECKSUM CODE

md5 /Path-to-file/$filename.xml

f49c2e9f08490c055cddba6d3b049f94

openssl sha1 /Path-to-file/$filename.xml

FIXITY

CHECKSUM CODE

md5 /Path-to-file/$filename.xml

f49c2e9f08490c055cddba6d3b049f94

openssl sha1 /Path-to-file/$filename.xml

1882e58c22a2c132b7b9763d24ae9bffe68d00ab

ALGORITHMS

MD5

f49c2e9f08490c055cddba6d3b049f94

ALGORITHMS

MD5

Message Digest Algorithm 5

- Written in 1991 to replace... MD4
- 32-character hexadecimal string
- Used in cryptography and data integrity
  - Considered "**cryptographically broken**"
- Current usage: data integrity for file download and transfer
  - Example: Download of Android applications, Microsoft updates

ALGORITHMS

SHA-1

1882e58c22a2c132b7b9763d24ae9bffe68d00ab

ALGORITHMS

SHA-1

Secure Hash Algorithm 1

- Designed by **NSA** (US National Security Agency) in 1995
- 40-character hexadecimal string
- Used in cryptography and data integrity
  - Considered **cryptographically questionable**
- Online transactions will be encrypted under SHA-2, SHA-3 starting in 2017
- Current usage: data integrity for file download and transfer
  - Example: GIT repository "consistency check"

ALGORITHMS

OTHER CRYPTOGRAPHIC ALGORITHMS & HASH FUNCTIONS

ALGORITHMS

OTHER CRYPTOGRAPHIC ALGORITHMS & HASH FUNCTIONS

- BLAKE
- Tiger
- Whirlpool
- GOST
- HAVAL
- SHA-2, SHA-3

CHECKSUMS FOR

AUDIOVISUAL MEDIA

DAVE RICE

CHECKSUMS FOR AUDIOVISUAL MEDIA

DAVE RICE

CHECKSUMS FOR AUDIOVISUAL MEDIA

- Thesis: Whole-file checksums for AV files are incomplete
- **Scale**: Fixity for large files that contain **complex data** should be analyzed on a **more granular level**
- Failures for whole-file checksums for audiovisual data objects cannot pinpoint **specific errors**
- Files may appear to play back (render, decode) perfectly to viewer

CHECKSUMS FOR AUDIOVISUAL MEDIA

ERROR CONCEALMENT

CHECKSUMS FOR AUDIOVISUAL MEDIA

ERROR CONCEALMENT

- Video formats are designed to **conceal errors**
  - **Examples**: MPEG-2, MP3, DV formats (DVCam, miniDV, HDV)
- Built-in technology hides discrepancies from viewer during playback

CHECKSUMS FOR AUDIOVISUAL MEDIA

ERROR CONCEALMENT - HOW IT WORKS

CHECKSUMS FOR AUDIOVISUAL MEDIA

ERROR CONCEALMENT - HOW IT WORKS

MPEG-2, MP3, DV formats (DVCam, miniDV, HDV)

CHECKSUMS FOR AUDIOVISUAL MEDIA

ERROR CONCEALMENT - HOW IT WORKS

MPEG-2, MP3, DV formats (DVCam, miniDV, HDV)

- **Checksums** produced during the **encoding** process for each **data packet**
- Each packet contains data that makes up the **audio and video stream**
- Checksums are (usually) stored in the file's **header**
- During the **decoding** process (**playback** via application), packets with bad checksums are skipped
- Decoding application plays packet on either side of bad section

FRAME-LEVEL CHECKSUMS

FRAME-LEVEL CHECKSUMS

FORMATS WITHOUT ERROR CONCEALMENT

FRAME-LEVEL CHECKSUMS

FORMATS WITHOUT ERROR CONCEALMENT

- **ffmpeg's framemd5** produces frame-level checksums for AV formats
- This allows a digital repository to...

FRAME-LEVEL CHECKSUMS

FORMATS WITHOUT ERROR CONCEALMENT

- **ffmpeg's framemd5** produces frame-level checksums for AV formats
- This allows a digital repository to...
  - Locate the **exact frame** that yielded error
  - Reuse of original checksums after migration to **lossless formats** (new container)
  - Checksum for AV content remains unchanged even when **metadata** changes

CREATING BATCH CHECKSUMS

MINI-GUIDE

CREATING CHECKSUMS

STEP ONE

CREATING CHECKSUMS

STEP ONE

Designate a package

CREATING CHECKSUMS

STEP ONE

Designate a package

- OAIS package or similar
- SIP: Recently acquired submission or source data
- AIP: Group of data objects prepared for archiving
- BagIt "Bag"
- Preparation for long-term storage (LTO, cloud, or other)

CREATING CHECKSUMS

STEP TWO

CREATING CHECKSUMS

STEP TWO

Create actual checksums contained in a log file or manifest

CREATING CHECKSUMS

STEP TWO

Create actual checksums contained in a log file or manifest

- Manifest is probably a text file
- Programs for **batch** checksum creation: **md5deep**, **hashdeep**
- Algorithms: MD5 and/or SHA-1 (most likely)
- Manifest contains either **one or two strings** per file

CREATING CHECKSUMS

STEP THREE

CREATING CHECKSUMS

STEP THREE

Set up an audit schedule

CREATING CHECKSUMS

STEP THREE

Set up an audit schedule

- Check files agains the manifest at intervals through the year
- Can be performed **manually** or via **automation**
  - Some **DAMS** produce checksums according to schedule
  - Create a **CRON** job to perform scheduled audits

CREATING CHECKSUMS

STEP FOUR

CREATING CHECKSUMS

STEP FOUR

Run fixity checks after lifecycle events, such as...

CREATING CHECKSUMS

STEP FOUR

Run fixity checks after lifecycle events, such as...

- Data is transferred to a brand new server (every 4 years)
- Data is transferred to a new backup location (LTO, etc.)
- Data is compromised (hack, mismanagement, backups recovered)
- After a disaster (power outage, flood, fires, etc.)

WHY SO MANY CHECKSUMS?

WHY SO MANY CHECKSUMS?

1. To determine whether data has changed
2. **Uncover issues** in repository workflows or infrastructure
3. To **prove consistency** within the repository: show that data has definitively not changed or that storage and archival workflows are working properly

WHY NOT JUST RUN CHECKSUM ALL THE TIME?

WHY NOT JUST RUN CHECKSUM ALL THE TIME?

- **Heavy processing load**
  - Running checksums can make systems unusable for other processes
- **Time**
  - Whole-file checksums for 1TB of data takes hours
  - Frame-level checksums for 1TB of data takes… many more hours
  - (contingent on age/performance of systems)

PROPRIETARY SYSTEM CHECKSUMS

PROPRIETARY SYSTEM CHECKSUMS

- Some products produced by vendors make their own checksums
  - Typical of LTO software vendors
  - Some cloud storage vendors
  - DP application & DAMS vendors
- Learn the points in the workflow when fixity checks are performed
- Even if this process is proprietary, try to secure access to actual hashes (algorithm strings)

MANIFEST STORAGE

MANIFEST STORAGE

- Text or log files (created manually or with automation)
- Within your DAM or catalog
- Database created to manage fixity checks
- Proprietary software products
- Log "sidecar" files stored alongside digital objects
- Within file metadata itself

THE

END.