

**Fixity Checks: Checksums, Message Digests and Digital Signatures**  
**Audrey Novak, ILTS**  
**Digital Preservation Committee**

**November 2006**

**Introduction:**

Fixity, in preservation terms, means that the digital object has not been changed between two points in time<sup>1</sup> or events. Technologies such as checksums, message digests and digital signatures are used to verify a digital object's fixity. Fixity information, the information created by these fixity checks, provides evidence for the integrity and authenticity of the digital objects and are essential to enabling trust. To ensure trust, for example, a repository may generate a checksum, message digest or digital signature as part of the ingest process for submission of content. Then at regular intervals using the same algorithm, the repository can regenerate the values and compare them to the original. If the values are the same a trusted repository has evidence that there has been no unauthorized changes to a digital object. A fixity check may be used to verify that any action taken upon the digital resource does not alter the resource. Without such evidence, one is unable to prove that the action did not alter the digital resource. Many events in the digital resource's lifecycle introduce the possibility of unintentional change, including: submission, retrieval, migration, transfer to other media, network transmission, or simply the passage of time.

**Definitions:**

Fixity checks are all used in the same basic way. A value is initially generated and saved. It is then recomputed and compared to the original to ensure the object (file or bitstream) has not changed. Despite this similarity, all fixity checks are not the same. Although the terms are frequently used interchangeably, checksums, message digests and digital signatures are, in fact, very different tools.

A checksum is the simplest and least secure method of verifying fixity. Checksums are typically used in error-detection to find accidental problems in transmission and storage. The least complicated checksum algorithms do not account for such changes as the re-ordering of bytes or changes that cancel one another out. The more secure checksums, such as cyclic redundancy check (CRC) are hash functions that control for such changes. Checksums, however, because of the comparative simplicity of their mathematical algorithms, are vulnerable to deliberate and malicious tampering.

Unlike checksums, cryptographic hash functions such as message digests are not prone to attack. A message digest is computed by applying an algorithm to the file of any length to produce a unique, short, uniform length character string. What makes message digests more secure than checksums is the complexity of the algorithm. A message digest is like the fingerprint for a digital object. For example, a message digest for a digital image jpeg file is: 97b3847a4ac1dcb037e2a7914c6f684d. And a message digest for an audio mp3 file is: 93326bff6636655dcd6abff18ed2de997. Change one pixel or one note in

---

<sup>1</sup> <http://www.oclc.org/research/projects/pmwg/premis-final.pdf>

the files and their message digests will be completely different. (Note, however, that they will always be the same length when computed using the same algorithm.) Hashes are one-way operations; a hash can be created from a digital object, but the digital object cannot be re-created from the hash. (Encryption, on the other hand, is a two-way operation).<sup>2</sup> MD5 (<http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html>) and Secure Hash Algorithm, SHA-1, ([http://www.w3.org/PICS/DSig/SHA1\\_1\\_0.html](http://www.w3.org/PICS/DSig/SHA1_1_0.html)) are commonly used cryptographic hash algorithms.

Digital signatures combine a hash message digest with encryption. A digital signature starts with the creation of a message digest from the digital object. “The message digest is then encrypted using asymmetric cryptography. Asymmetric cryptography is based on using a pair of keys: a private key to encrypt and a public key to decrypt. The private key must be held secretly and securely by the signer... The signature can be verified by decrypting the signature with the signer’s public key and comparing the now-decrypted digest with a new digest produced by the same algorithm from the same content.”<sup>3</sup>

### **Fixity Checks in Digital Preservation**

As the PREMIS Final report outlines, digital signatures are used in preservation repositories in three ways.

- For submission to the repository, an agent (author or submitter) might sign an object to assert that it truly is the author or submitter.
- For dissemination from the repository, the repository may sign an object to assert that it truly is the source of the dissemination.
- For archival storage, a repository may sign an object so that it will be possible to confirm the origin and integrity of the data. In this case, the signature itself and the information needed to validate the signature must be preserved.

Despite this heavy emphasis in PREMIS on digital signatures, in practice most repositories that support some level of preservation utilize message digests to checksums to ensure fixity.

### **Examples of Practical Implementations**

A quick survey of ingest procedures indicates that MD5 hash algorithm is ubiquitous in preservation repositories. No evidence of just checksums or digital signatures creation and validation was found.

The Yale University Library Rescue Repository employs SHA1 and MD5 algorithms upon ingest. These values are saved in a companion file along with the ingested resource.

At Harvard an MD5 hash is required when an object is deposited in a Harvard Repository SFTP drop box. It is used to validate the integrity of the digital object during transfer

---

<sup>2</sup> <http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>

<sup>3</sup> Premis, 4-7

from the depositor's system to the drop box and into the repository itself. It is also used for file validity within the repository over time.

The NDIIPP Architecture and the Archive Ingest & Handling Test is designed to test the feasibility of transferring digital archives from one institution to another. Phases of the test will assess the process of digital ingest, document useful practices, maximize automated handling of digital material, and identify areas that require further research or development. (Section 2.3.1 of the AIHT Statement of Work (SOW) 26504-01, 12/19/2003). Participants include Library of Congress, Old Dominion, Johns Hopkins, Stanford and Harvard. The test focus made available a great deal of information available about general ingest procedures All of these AIHT projects included MD5 for pre- and post- ingest fixity verification.  
(<http://www.digitalpreservation.gov/index.php?nav=3&subnav=14>)

### **Issues to Consider Regarding Fixity Checks**

Trust in the authenticity of a digital resource is ensured by the fixity information. The fixity information needs to be protected in many of the same ways as the digital resource itself. Disaster recovery through redundancy and replication applies equally to fixity information and the digital resource.

A mismatch between message digests generated over time from the same source file indicates a change in either the source file content or a problem with the message digest itself. Although the probability is small, a match between the digests is not always a guarantee that the source file content is unchanged. “Digest algorithms are inherently subject to *collisions*, in which two different inputs generate the same digest. Digest algorithms are designed to make collisions unlikely, but some of the assumptions underlying these designs do not hold in digital preservation applications. For example, the analysis of the algorithm normally assumes that the input is a random string of bits, which for digital preservation is unlikely. “

<http://www.dlib.org/dlib/november05/rosenthal/11rosenthal.html>

Additionally, over time the encryption algorithms used to generate the message digests can become vulnerable. “Recently, for example, the widely used MD5 and SHA1 algorithms appear to have been broken. A digital preservation system that audits against previous message digests must preemptively replace its digest algorithm with a new one before the current one is broken To do so, it should audit against the current digest to confirm that the item is still good then compute a digest using the replacement algorithm. This should be appended to the stored list of digests for the item. “

<http://www.dlib.org/dlib/november05/rosenthal/11rosenthal.html>

Finally, but most importantly, generating, storing, replicating, and verifying message digests is not without significant staff and system costs. A tension exists between the value of the content and the cost of this level of protection. Should message digests be created, periodically verified, and maintained for all digital assets or should different levels of service be established?

## **Recommendations and Best Practices for Yale University Library**

Specifically in the Rescue Repository and any other repository managed by YUL:

- Message digests should be protected with the same rigor that is applied to the original content file in terms of redundancy and replication. Additionally, to protect against system failures, operator error and malicious attacks message digests should be stored separate from the original source content. More than one message digest using different algorithms should be generated.
- Using the Rescue Repository environment, a test conversion to a new message digest algorithm should be planned in order to help establish procedures that may be needed for preemptive replacement.
- Also using the Rescue Repository environment, a periodic message digest verification process should be implemented in order to establish the veracity of the source content over time, but equally importantly, to better understand the level-of-effort and system resource requirements required for this level of service
- Message digests should be maintained and managed within the PREMIS framework.
- Best practices within the digital preservation community regarding fixity checks should be continually reviewed. Particular attention should be applied to the use of digital signatures.

**PREMIS: Fixity**

<b>Semantic unit</b>	<b>fixity</b>		
<b>Semantic components</b>	messageDigestAlgorithm, messageDigest, messageDigestOriginator		
<b>Definition</b>	Information used to verify whether an object has been altered in an undocumented or unauthorized way.		
<b>Data constraint</b>	Container		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable (see usage note)	Applicable	Applicable (see usage note)
<b>Repeatability</b>		Repeatable	Repeatable
<b>Obligation</b>		Optional	Optional
<b>Creation/Maintenance notes</b>	Automatically calculated and recorded by repository.		
<b>Usage notes</b>	<p>To perform a fixity check, a message digest calculated at some earlier time is compared with a message digest calculated at a later time. If the digests are the same, the object was not altered in the interim. Recommended practice is to use two or more message digests calculated by different algorithms.</p> <p>The act of performing a fixity check and the date it occurred would be recorded as an Event. The result of the check would be recorded as the eventOutcome. Therefore, only the messageDigestAlgorithm and messageDigest need to be recorded as objectCharacteristics for future comparison.</p> <p>Representation level: It could be argued that if a representation consists of a single file, or if all the files comprised by a representation are combined (e.g., zipped) into a single file, then a fixity check could be performed on the representation. However, in both cases the fixity check is actually being performed on a file, which in this case happens to be coincidental with a representation.</p> <p>Bitstream level: Message digests can be computed for bitstreams although they are not as common as with files. For example, the JPX format, which is a JPEG2000 format, supports the inclusion of MD5 or SHA-1 message digests in internal metadata that was calculated on any range of bytes of the file.</p> <p>See “Fixity, integrity, authenticity,” page 4-5.</p>		

<b>Semantic unit</b>	<b>messageDigestAlgorithm</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The specific algorithm used to construct the message digest for the digital object.		
<b>Data constraint</b>	Value should be taken from a controlled vocabulary.		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		MD5 Adler-32 HAVAL SHA-1 SHA-256 SHA-384 SHA-512 TIGER WHIRLPOOL	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory

<b>Semantic unit</b>	<b>messageDigest</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The output of the message digest algorithm.		
<b>Rationale</b>	This must be stored so that it can be compared in future fixity checks.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		7c9b35da4f2ebd436f 1cf88e5a39b3a257ed f4a22be3c955ac49da 2e2107b67a1924419 563	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory

<b>Semantic unit</b>	<b>messageDigestOriginator</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The agent that created the original message digest that is compared in a fixity check.		
<b>Rationale</b>	A preservation repository may ingest files that have had message digests calculated by the submitter; checking these ensures that the file as received is the same as the file as sent. The repository may also ingest files that do not have message digests, and so must calculate the initial value upon ingest. It can be useful to know who calculated the initial value of the message digest.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		DRS A0000978	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Optional	Optional
<b>Creation/ Maintenance notes</b>	If the calculation of the initial message digest is treated by the repository as an Event, this information could be obtained from an Event record.		
<b>Usage notes</b>	The originator of the message digest could be represented by a string representing the agent (e.g., "DRS" referring to the archive itself) or a pointer to an agent description (e.g., "A0000987" taken here to be an agentIdentifierValue).		

## PREMIS: Digital Signature

<b>Semantic unit</b>	<b>signatureMethod</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	A designation for the encryption and hash algorithms used for signature generation.		
<b>Rationale</b>	The same algorithms must be used for signature validation.		
<b>Data constraint</b>	Value should be taken from a controlled vocabulary.		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		DSA-SHA1 RSA-SHA1	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory
<b>Usage notes</b>	Recommended practice is to encode the encryption algorithm first, followed by a hyphen, followed by the hash (message digest) algorithm.		

<b>Semantic unit</b>	<b>signatureValue</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The digital signature; a value generated from the application of a private key to a message digest.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		juS5RhJ884qoFR 8fIVXd/rbrSDVGn 40CapgB7qeQiT +rr0NekEQ6BHh UA8dT3+BCTBU QI0dBjlm19lwzEN XvS83zRECjzXb MRTUtVZiPZG2p qKPnL2YU3A964 5UCjTXU+jgFum v7k78hieAGDzNc i+PQ9KRmm//icT 7JaYztgt4=	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory

<b>Semantic unit</b>	<b>signatureValidationRules</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The operation to be performed as part of signature validation.		
<b>Rationale</b>	The repository should not assume that the procedure for validating any particular key will be known many years in the future without documentation.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>		Applicable	Applicable
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory
<b>Usage notes</b>	<p>This may include the canonicalization method used before calculating the message digest, if the object was normalized before signing.</p> <p>This value could also be a pointer to archive documentation.</p>		

<b>Semantic unit</b>	<b>signatureProperties</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	Additional information about the generation of the signature.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Repeatability</b>		Repeatable	Repeatable
<b>Obligation</b>		Optional	Optional
<b>Usage notes</b>	<p>This may include the date/time of signature generation, the serial number of the cryptographic hardware used, or other information related to the generation of the signature. Repositories will likely want to define a suitably granular structure to signatureProperties.</p>		

<b>Semantic unit</b>	<b>keyValue</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The value of the signer's public key.		
<b>Rationale</b>	The signer's public key might be included in the signer's X509 certificate, if this is recorded under keyVerificationInformation. However, since the key itself is necessary, it is useful to isolate it as a separate and required semantic unit.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory
<b>Usage notes</b>	Different types of key will have different structures and parameters. Recommended practice is to represent key values following the W3C's <i>XML-Signature Syntax and Processing</i> ( <a href="http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/">www.w3.org/TR/2002/REC-xmlsig-core-20020212/</a> ).		

<b>Semantic unit</b>	<b>keyVerificationInformation</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	Additional information needed to verify the signer's public key used to validate the digital signature.		
<b>Data constraint</b>	None		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Optional	Optional
<b>Usage notes</b>	This may include a certificate or certificate chain, and/or a revocation list. Repositories will likely want to define a suitably granular structure to keyVerificationInformation.		

<b>Semantic unit</b>	<b>keyInformation</b>		
<b>Semantic components</b>	keyType, keyValue, keyVerificationInformation		
<b>Definition</b>	Information about the signer's public key needed to validate the digital signature.		
<b>Rationale</b>	To validate a digital signature for an object, one first recalculates the message digest for the object, and then uses the public key of the signer to verify that the value of the signature (signatureValue) is correct. The repository must therefore have the public key value and some assurance that it truly belongs to the signer.		
<b>Data constraint</b>	Container		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Optional	Optional

<b>Semantic unit</b>	<b>keyType</b>		
<b>Semantic components</b>	None		
<b>Definition</b>	The type of key, denoted by the algorithm used to generate the key.		
<b>Data constraint</b>	Value should be taken from a controlled vocabulary.		
<b>Object category</b>	<b>Representation</b>	<b>File</b>	<b>Bitstream</b>
<b>Applicability</b>	Not applicable	Applicable	Applicable
<b>Examples</b>		DSA RSA PGP SPKI	
<b>Repeatability</b>		Not repeatable	Not repeatable
<b>Obligation</b>		Mandatory	Mandatory