Technische Richtlinien zur Authentizität

Diese Richtlinien ergänzen die Empfehlungen zur Gewährleistung der Authentizität und Integrität archivierter digitaler Unterlagen.

AutorInnen Arbeitsgruppe Authentizität des Pilotprojekts KOSTPROBE:
Georg Büchler, KOST
Markus Lischer, StALU
Claudia Schmucki, StAZH
Peter Witschi, StAAR
Redaktion Georg Büchler
Version 0.1

Disposition

1 Metadaten

Die Arbeitsgruppe Metadaten des KOST-Pilotprojekts KOSTPROBE schlägt einen Metadatenkatalog vor, welcher der PREMIS-Spezifikation entspricht[1]. Die folgenden Metadatenelemente sind für die Gewährleistung der Authentizität von besonderer Bedeutung:

fixity messageDigestAlgorithm messageDigest messageDigestOriginator	*Erlaubt die Ablage eines Hashwertes.*
significantProperties	*Vorgesehen für die Definition derjenigen Eigenschaften, die bei Migrationen erhalten bleiben müssen*
eventIdentifier eventIdentifierType eventIdentifierValue eventType eventDateTime	*Die Entität "Event" dient dazu, Informationen zu Ereignissen wie z.B. Migrationen festzuhalten. ANMERKUNG: Zur Zeit ist noch nicht definiert, welche Elemente innerhalb dieser Entität verwendet werden sollen. Nebenstehend finden sich deshalb vorerst nur die laut Standard obligatorischen Elemente*

[1] Siehe http://www.loc.gov/standards/premis/. Der PREMIS-Metadatenkatalog findet sich unter http://www.oclc.org/research/projects/pmwg/premis-final.pdf.

2 Integrität

2.1 Redundante Speicherung

Die Anzahl der redundant gehaltenen Exemplare ist ein Indikator für die Fehlerwahrscheinlichkeit. Wenn wir die Fehlerwahrscheinlichkeit für ein Exemplar mit $1/x$ ansetzen, wird sie bei n unabhängigen Exemplaren $1/x^n$. Die Anzahl redundant gehaltener Exemplare wird deshalb durch die Anforderungen an die Sicherheit bestimmt. Für archivierte Unterlagen im Kontext dieser Empfehlung und Richtlinien wird die Aufbewahrung von zwei Exemplaren empfohlen[2].

2.2 Hash-Werte

2.2.1 Vorgehen

Um die Integrität der archivierten Unterlagen jederzeit einfach kontrollieren zu können, empfiehlt es sich, auf Hash-Werte zurückzugreifen. Dabei ist ein mehrstufiges Verfahren angemessen: Die Hash-Werte sämtlicher Dateien einer Ablieferung werden errechnet und in den Metadaten[3] sowie einem Masterfile (einem einfachen Textdokument mit Liste der Dateinamen und zugehörigen Hash-Werten) abgelegt. Von diesem wird wiederum der Hash-Wert errechnet. Dieses Resultat kann danach ausgedruckt, unterschrieben und getrennt aufbewahrt werden. Beim Zugriff auf die archivierten Unterlagen kann nun automatisch der Hash-Wert berechnet und mit dem in den Metadaten hinterlegten Wert verglichen werden. Bei Problemen erlauben das Masterfile und der ausgedruckte Master-Hash-Wert eine menschenlesbare Überprüfung[4].
Falls die Hash-Werte nicht übereinstimmen, liegt ein Problem vor. Das Dokument kann, falls gewünscht, trotzdem ausgeliefert werden, das Problem muss allerdings untersucht werden. Dazu wird zuerst das redundant gespeicherte Dokument geladen und sein Hash-Wert überprüft. Falls dieser mit dem hinterlegten Wert übereinstimmt, handelt es sich um das authentische Dokument, welches übernommen werden kann. Falls auch dieser nicht mit dem hinterlegten Hash-Wert übereinstimmt, soll versucht werden, auf Grund der Log-Informationen dem Fehler auf den Grund zu kommen. Verläuft dies erfolglos, wird empfohlen, beide nicht mehr authentischen Dokumente weiter zu archivieren mit einem entsprechenden Eintrag in den Metadaten. (Schätzungsweise werden sich die

[2] Siehe dazu mit weiteren Begründungen DOMEA-Konzept, Erweiterungsmodul zum Organisationskonzept 2.0, Technische Aspekte der Archivierung elektronischer Akten (2004), p. 40f. (Online unter http://www.kbst.bund.de/Anlage306107/Technische-Aspekte-der-Archivierung-elektronischer-Akten-pdf-465-kB.pdf.) Das DOMEA-Konzept verlangt die Speicherung dreier identischer Exemplare. Auf Grund der Risikoanalyse sowie der zusätzlichen Verwendung von Hash-Werten im vorliegenden Fall werden zwei Exemplare als angemessen erachtet.
[3] Der PREMIS-Metadatenkatalog sieht dazu ein Element namens *fixity* vor. Siehe oben zum Kapitel Metadaten.
[4] Es sei angemerkt, dass dieses Verfahren zwar auf ähnlichen Technologien basiert wie die digitale Signatur, aber einfacher ist: Es kommt ohne externe Zertifikationsinstanz aus und erreicht die Nachprüfbarkeit durch den Papierausdruck des Hash-Wertes.

Probleme – d.h. nicht übereinstimmender Hash-Wert – im Bereich von Bruchteilen von Promillen bewegen und deshalb mit vertretbarem Aufwand erledigt werden können.)

2.2.2 Exkurs: Kryptografische Hash-Funktionen[5]

Eine Hash-Funktion bildet eine breite, meist unbegrenzte Auswahl von Eingangswerten auf eine begrenzte Auswahl kurzer Ausgangswerte (Hash-Werte) ab. Eine bestens bekannte, allerdings sehr einfache Hash-Funktion ist die Quersumme, die eine beliebige Zahl als Eingangswert nimmt, alle Ziffern addiert und dieses Verfahren solange wiederholt, bis sie eine Zahl von 1 bis 9 als Ausgangswert liefert. Die hier angesprochenen Hash-Funktionen verwenden natürlich weitaus kompliziertere Algorithmen, die aber nach dem gleichen Prinzip funktionieren.
Eine grundsätzliche Eigenschaft von Hash-Funktionen ist, dass aus verschiedenen Hash-Werten zwingend auf verschiedene Eingangswerte geschlossen werden kann. Da die Zahl der Hash-Werte begrenzt ist, gilt das Umgekehrte nicht zwingend: Gleiche Hash-Werte können auch von verschiedenen Eingangswerten stammen. Man spricht dabei von Kollisionen. Je nach Anwendungsgebiet muss mit dieser Einschränkung mehr oder weniger vorsichtig umgegangen werden.
Eine kryptografische Hash-Funktion besitzt zusätzliche Eigenschaften, die sie für sicherheitsrelevante Anwendungen tauglich machen. Dabei geht es besonders um die Möglichkeit von und den Umgang mit Kollisionen. Es muss gezeigt werden können, dass die Funktion zwei wesentliche Eigenschaften hat:

- Preimage resistance: Wenn ein Hash-Wert h bekannt ist, soll es unmöglich sein, einen dazugehörigen Eingangswert zu finden. (Damit verwandt ist die second preimage resistance: Wenn ein Eingangswert bekannt ist, soll es unmöglich sein, einen anderen Eingangswert zu finden, der den gleichen Hash-Wert hat.
- Collision resistance: es soll unmöglich sein, zwei verschiedene (beliebige) Eingangswerte zu finden, die den gleichen Hash-Wert haben.

Keine Hash-Funktion kann dies garantieren. Da die möglichen Hash-Werte begrenzt sind, gibt es immer das Mittel des Ausprobierens, die sogenannte Brute-Force-Attacke. Moderne kryptografische Hash-Funktionen schützen sich dagegen hauptsächlich durch die Länge des Hash-Werts. MD5 erzeugt Hash-Werte mit einer Länge von 128 Bit, SHA1 solche von 160 Bit Länge. Dies macht Brute-Force-Attacken mit heute gängiger Hardware (und auch noch auf viele Jahre hinaus) undenkbar.
Eine Hash-Funktion gilt als geknackt, wenn ein Algorithmus existiert, der mit weniger als den für eine Brute-Force-Attacke benötigten Operationen eine Kollision erzeugen kann. Damit ist eine Hash-Funktion allerdings weiterhin preimage und second preimage resistant, mit ihr signierte Dokumente also noch

[5] Als Quellen für diesen Anhang wurden hauptsächlich die entsprechenden Wikipedia-Einträge benutzt (in der englischen Version qualitativ besser als in der deutschen). Siehe en.wikipedia.org unter „Hash Function“, „Cryptographic Hash Function“, “MD5”, “SHA-1” bzw. de.wikipedia.org unter „Hash-Funktion“, „MD5“, „Secure Hash Algorithm“.

nicht gefährdet. Im allgemeinen löst das Knacken einer Funktion jedoch den Übergang zu weiter fortgeschrittenen Hash-Funktionen aus. Die beiden meistverbreiteten Hash-Funktionen, MD5 und SHA1, wurden in den letzten Jahren geknackt. Dies bedeutet, dass sie für sicherheitsrelevante Anwendungen in nächster Zeit ersetzt werden müssen.
Die Überwindung der Kollisionsresistenz kann besonders dann ausgenutzt werden, wenn ein Attacker den Eingangswert selber bestimmen kann. Man kann sich beispielsweise jemanden vorstellen, der einem Geschäftspartner einen Vertrag zur elektronischen Signatur vorlegt und heimlich eine veränderte Version mit gleichem Hash-Wert produziert hat. Er könnte sich dann auf diese veränderte Version berufen und ihre „Authentizität" mit dem Hash-Wert nachweisen. Dieses Szenario kann für die Anwendung von Hash-Funktionen im Archivkontext ausgeschlossen werden. Es ist deshalb nicht problematisch, MD5 oder SHA1 für die Zwecke des Archivs zu verwenden.
Zur Berechnung der MD5- oder SHA1-Hashwerte stehen einfache, bedienerfreundliche Programme als Freeware zur Verfügung. Diese ermöglichen es auch Laien, schnell den Hashwert einer oder mehrerer Dateien zu berechnen. Beispiele solcher Programme sind HashCalc (http://www.slavasoft.com/hashcalc/index.htm), Jacksum (http://www.jonelo.de/java/jacksum/index_de.html) und viele weitere. Die Automatisierung der Hashwert-Überprüfung, wie oben erwähnt, ist damit auch relativ einfach möglich.

Es wird empfohlen, den SHA-1-Hashwert zu verwenden. Bei diesem Algorithmus stehen die Sicherheit, die Schnelligkeit der Berechnung sowie die Verfügbarkeit entsprechender Software in einem idealen Verhältnis[6].

[6] Siehe dazu auch T.C. Stein, E.A. Guinness, S.H.Slavney, „Establishing a Mechanism for Maintaining File Integrity within the Data Archive", paper presented at PV 2005, Edinburgh (vorläufige Publikation online unter http://www.ukoln.ac.uk/events/pv-2005/pv-2005-final-papers/039.pdf), p.5.